THE TRS-80* MAGAZINE

May 1980 $2.00 U.S., $2.50 Canada

80 microcomputing T.M.

80 Microcomputing 5/80 #5

Word Processors: 2 systems compared. Pg. 56.

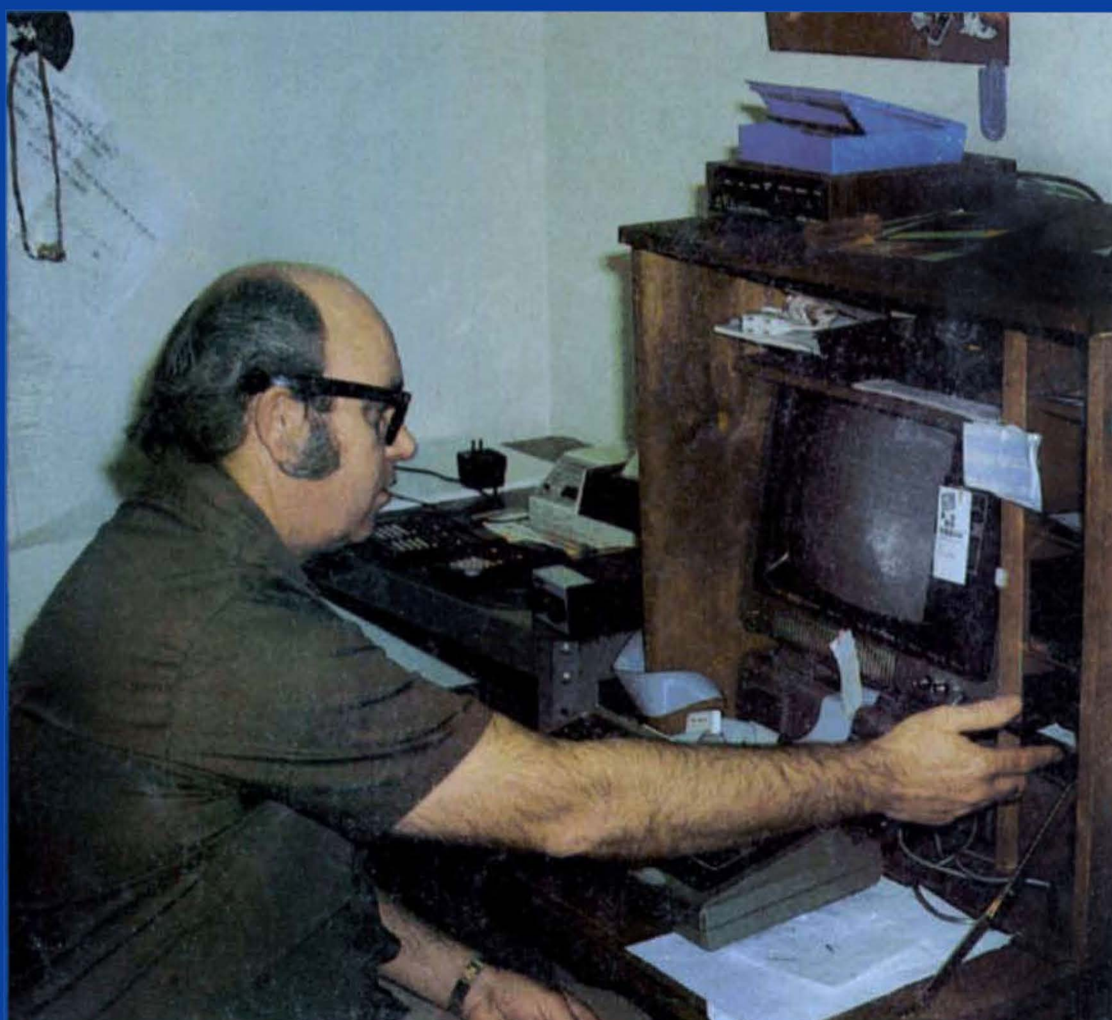
Carpool:
Save gas, travel with a friend. **Pg. 62.**

Tiny Compiler:
A simple compiler in BASIC. **Pg. 44.**

Computer Languages:
Alternatives to Level I & Level II. **Pg. 36.**

Electronic Mail:
Computer bulletin board systems. **Pg. 110.**

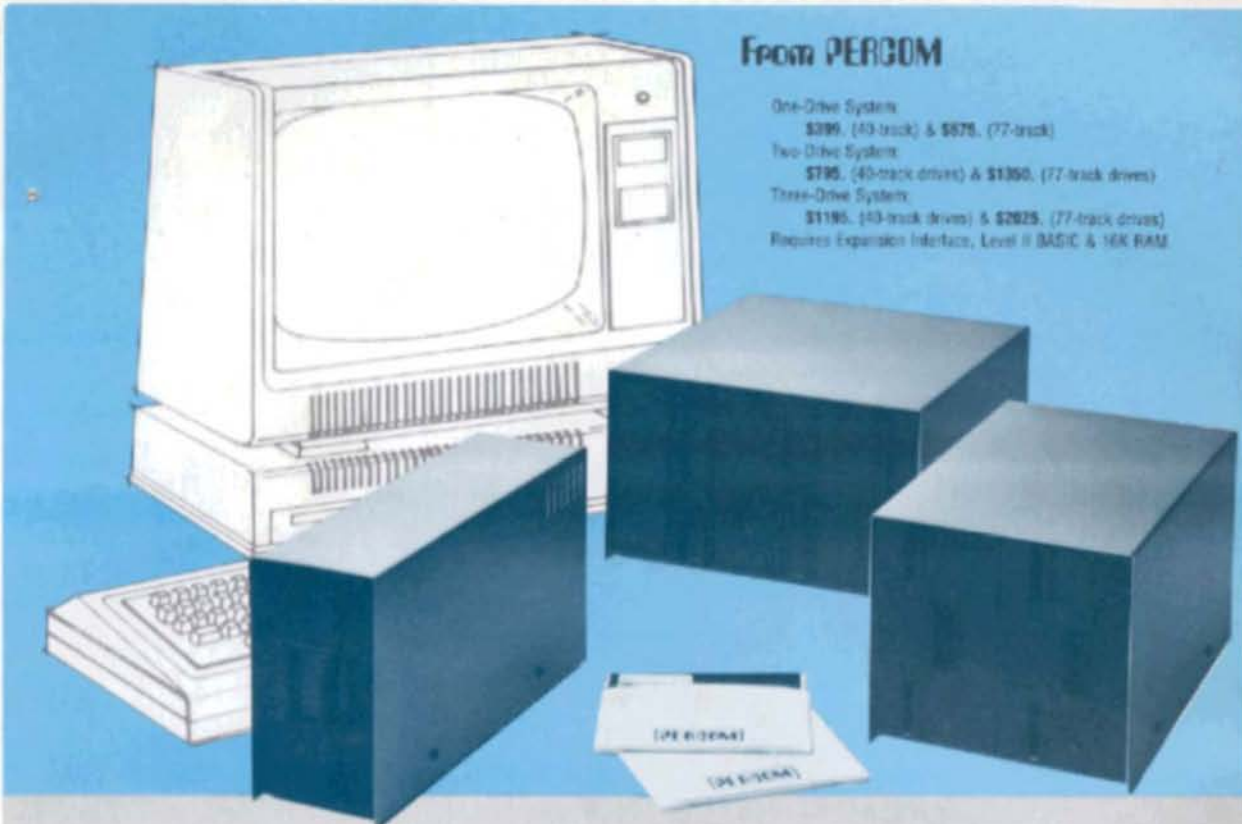
Disk Alternatives:
Cheap, fast and reliable data storage. **Pg. 82.**



0 5
0 74470 65947

Plus:
Software Reviews, Hardware Reviews, Book Reviews.
More than 30 articles and columns.

*a trademark of Tandy Corporation



Low Cost Add-On Storage for Your TRS-80*. In the Size You Want.

When you're ready for add-on disk storage, we're ready for you. Ready with six mini-disk storage systems — 102K bytes to 591K bytes of additional *on-line* storage for your TRS-80*.

- **Choose either 40-track TFD-100™ drives or 77-track TFD-200™ drives.**
- **One-, two- and three-drive systems immediately available.**
- **Systems include Percom PATCH PAK #1™, on disk, at no extra charge. PATCH PAK #1™ de-glitches and upgrades TRSDOS* for 40- and 77-track operation.**
- **TFD-100™ drives accommodate "flippy disks." Store 205K bytes per mini-disk.**
- **Low prices. A single-drive TFD-100™ costs just $399. Price includes PATCH PAK #1™ disk.**
- **Enclosures are finished in system-compatible "Tandy-silver" enamel.**

Whether you need a single, 40-track TFD-100™ add-on or a three-drive add-on with 77-track TFD-200™s, you get more data storage for less money from Percom.

Our TFD-100™ drive, for example, lets you store 102.4K bytes of data on one side of a disk — compared to 80K bytes on a TRS-80* mini-disk drive — and 102.4K bytes on the other side, too. Something you can't do with a TRS-80* drive. That's almost 205K bytes per mini-disk.

And the TFD-200™ drives provide 197K bytes of on-line storage per drive — 197K, 394K and 591K bytes for one-, two and three-drive systems.

PATCH PAK #1™, our upgrade program for your TRSDOS*, not only extends TRSDOS* to accommodate 40- and 77-track drives, it enhances TRSDOS* in other ways as well. PATCH PAK #1™ is supplied with each drive system at no additional charge.

The reason you get more for less from Percom is simple. Peripherals are not a sideline at Percom. Selling disk systems and other peripherals is our main business — the reason you get more engineering, more reliability and more back up support for less money.

In the Product Development Queue . . . a *printer interface* for using your TRS-80* with any serial printer, and . . . the *Electric Crayon*™ to map your computer memory onto your color TV screen — for games, animated shows, business displays, graphs, etc. Coming PDQ!

™ TFD-100, TFD-200, PATCH PAK and Electric Crayon are trademarks of PERCOM DATA COMPANY.
*TRS-80 and TRSDOS are trademarks of Tandy Corporation and Radio Shack which have no relationship to PERCOM DATA COMPANY.

PERCOM

PERCOM DATA COMPANY, INC.
211 N. KIRBY • GARLAND, TX. • 75042

To order add-on mini-disk storage for your TRS-80*, or request additional literature, call Percom's toll-free number: 1-800-527-1592. For detailed Technical information call (214) 272-3421.

Orders may be paid by check or money order, or charged to Visa or Master Charge credit accounts. Texas residents must add 5% sales tax.

Percom *'peripherals for personal computing'*

✓Reader Service—see page 162

80 microcomputing DATA

APPLICATION

GENERAL

HARDWARE

REVIEWS

STYLE

UTILITY

REGULARS

Cover: *TRS-80 Disk & Other Mysteries* author Harvard Pennington.
Photo by Jim Perry.

80 Microcomputing (ISSN # applied for) is published monthly by 1001001 Inc., Pine Street, Peterborough, NH 03458. Application to mail second class postage rate is pending at Peterborough, NH 03458 and at additional mailing offices. Phone: 603-924-3873/4. Subscription rates in the U.S. are $15 for one year and $40 for three years. In Canada $17 for one year and $46 for three years. In Europe please contact Monika Nedela, Markstr. 3, D-7778, Markdorf, W. Germany. In South Africa contact *80 Microcomputing*, P.O. Box 782815, Sandton, S. Africa 2146. All other foreign subscriptions $20 (one year only) surface mail. 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.

META TECHNOLOGIES

Fellow TRS-80 User:

I'd like to thank you for your interest in our products, and, as our president, Bob Fiorelli, promised in our March ad, tell you something about MTC Product Engineering. The term "Engineering" may be misleading. A product offered for sale may involve research and development, design, implementation, test marketing, follow-up evaluation, re-engineering, secondary distribution and exhaustive testing. A number of issues and factors are involved in our product selection and engineering:

- The product should address the needs of a broad class of users.
- The product should free the user from as many non-essential or redundant tasks as possible, so he can spend his time being creative.
- The product should solve technical problems that most users would have difficulty solving themselves.
- Groups of compatible products should exist, so that the effectiveness of each one is multiplied when used with the others.
- The product should be "human-engineered" so that people can understand and use it, without wading through stacks of documentation (many people won't read it anyway).
- The product should provide continuing benefit and lasting value.

Take, for example, our programming tools. The same tools we offer for sale are used in our own development efforts. The SUPERSEDE program is in constant use here at MTC. SUPERSEDE is a good example of a product that is simple in concept, easy to use, and extremely effective in operation, especially when combined with other "programming tool" products.

Another product that embodies many of our philosophies is AIDS-II. This product provides users with an information management system that is simple to understand and use, and yet so powerful and effective that some customers are using it to virtually run their businesses within a week of its purchase. It provides continuing benefit to many diverse users, solutions to difficult technical problems, and compatible products (CALCS, MAILS, REFORGE, MINGLE) are available to multiply its effectiveness. As for human engineering, about 60% of the core of AIDS-II is devoted to making the system as simple and consistent as possible, while providing extremely flexible and powerful data management functions. Information management systems present some of the most demanding technical challenges and MTC excels in this area.

Many users need good sources of educational materials and technical/reference information. We have obtained or created several products to fill these needs, specifically H. C. Pennington's "TRS-80 Disk & Other Mysteries", REMSOFT's REMASSEM-1, and the MTC Technical Bulletin Service, with more to come. The MTC Technical Bulletins are intended for programmers. Their content is not merely comments on or a preview of Radio Shack publications.

Finally, I'd like to clear up some questions regarding diskettes. We sell Verbatim because they have the most widely recognized product and a deservedly fine reputation. Our diskettes are EXACTLY the same as available elsewhere at up to twice the price. Of the tens of thousands of diskettes we have sold nationally, less than one -tenth of one percent have been returned for any reason! We don't make our living selling diskettes. We sell them at our low prices to get your attention. Don't forget, a customer-appreciation month is planned in 1980 (we're not saying when). For one month, and one month only, we will sell a box of diskettes for $19.80. Sorry, one box per customer.

We at MTC look forward to providing you with the quality products and service you deserve and should expect. If we offered you anything less, we wouldn't be Meta Technologies Corporation.

Sincerely,

Alan R. Becker
Product Engineering

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

MTC AIDS-II

Ailing information? Doctor it up with AIDS-II. This Automated Information Directory System is user-defined, features user-specified fields and print/display formats, conditional record selection, updating of fields within records, sorting by any combination of fields, and much more! Unique "windowing" capability allows directories of unlimited size. Window size is typically 200 or more records in 32K. Can be used for mailing lists, client reference reporting, appointment "calendars", inventory records and other information systems. Easy to use. Defining a system takes about a minute. Formatting a report or defining a custom label, less than 30 seconds. Sorting 200 records takes less than 5 seconds. Add "subsystems" for additional capabilities.

MTC AIDS-II $49.95
For Model II $79.95

CALCULATION SUBSYSTEM (CALCS)

Use for report generation involving basic manipulation of numeric data, such as quantity & cost computations, balances carried forward and columnar totals. Expands capabilities with respect to inventory, accounting and other numeric-based information systems.

MTC CALCS $24.95
For Model II $39.95

MAILING/INFORMATION LIST SUBSYSTEM (MAILS)

Use for report & label generation involving formatting of primarily non-numeric data, such as custom, "N-up" and "N-copy" label forms, index-type report formatting, and sub-fielding capabilities for selection & print. Expands AIDS-II with respect to client/product reference systems, mailing lists and other non-numeric information systems.

MTC MAILS $24.95
For Model II $39.95

RECORD/FILE ORGANIZATION EXECUTIVE (REFORGE)

Use for expanding, contracting or reformatting AIDS-II files. Convert random & sequential files to AIDS-II format. Use for converting mailing lists to AIDS-II. Includes MERGE & PURGE capabilities for combining smaller files into larger ones and removing duplicate records.

MTC REFORGE $24.95
For Model II $39.95

PEEK&POKE $14.95

Frustrated because PEEK and POKE have been removed from Model II BASIC?. Satisfy your curiosity with PEEK&POKE from MTC. Included are 8-bit and 16-bit (LSB, MSB) self-relocating machine language routines, instructions, and demo program.

mTc-SHIRTS

HIGH-QUALITY, POLY-COTTON BLEND T-SHIRTS. White with Navy Blue neck and sleeve "ringers". MTC logo on sleeve. Top quality transfers of your choice on front.

Specify size (S,M,L,XL) and Transfer:

- **META TECHNOLOGIES MAKES EVERY BYTE COUNT!**
- **DON'T TOUCH MY BITS!**
- **RAM it!**
- **MICRO LOVERS TAKE SMALLER BYTES!**

mTc-SHIRTS $5.95

FILESORT $29.95
For Model II $49.95

Sort ASCII files by any combination of user-specified fields; ascending or descending, numeric or non-numeric keys. Optionally retains original file.

MAILSORT(Model I) $19.95

MAILSORT replaces existing sort in Radio Shack Disk Mailing List® system. Sort by any combination of fields, such as NAME, within CITY, within STATE. Hi-speed in-memory routine sorts 300 records in approximately 60 seconds. Minimum 32K recommended.

Transfer PROGRAMS and DATA from MODEL I to MODEL II

TRAN-SEND

$49^95 by MTC

Requires MODEL II and MODEL I with disk & RS-232. Simple to use, not a kit - nothing else to buy. Complete with custom cable, 5¼" & 8" floppies, instructions. May be used over phone lines.

Custom Cable only $19.95

Suitable for use with Radio Shack® transfer program(ACT 0131)

✓Reader Service – see page 162

80 REMARKS

by Wayne Green

A nerve wracking two or three months pass with little more being heard.

The problem of computer program theft is complex. One of the questions, which seems to have no answer is magnitude of program theft. So far, despite a repeated offer of $10,000 reward, no one has yet turned in a prosecutable case of an Instant Software program that has been stolen.

Months of Work

Let's suppose you have spent a few months writing a program and getting the bugs out of it. You submit the program to a publisher and hope for the best. In a few weeks you get word that your program is being actively considered for publication.

A nerve-wracking two or three months pass with little more being heard. Have they shelved your hard work? Did it fall between the cracks?

Eventually a big envelope comes in the mail. Your program is alive and well, but comments from associate editors have suggested several changes. One has added nearly two hundred lines of coding to make the program more foolproof and another has changed two of your subroutines to speed operator interaction and sorting.

You get to work again, updating the program and finally submit a finished version to the publisher. A few more weeks elapse and proofs for the documentation arrive. A test master copy of the program dump arrives, and the project is becoming a reality.

You have perhaps several thousands of dollars in time invested and so has the publisher, so you're anxious to see the package in the stores and royalties coming in.

A couple of weeks later you attend the local TRS-80 club meeting. When you arrive several fellows are running their systems and you try a new game. Just to your left you hear something which makes it plain that someone is running *your* program! Wow! They are obviously enjoying it and calling more fellows around to see it. Then, they start pulling cassettes out of their pockets to run off copies.

One way to make this sort of theft difficult is to build obstacles into the programs that prevent them from being easily copied. No system will defeat a determined hobbyist from making a workable copy, but there are some which make it darned hard. This approach cuts the losses to programmers somewhat, and the work involved in committing the theft makes it clear that hobbyists really are hurting someone seriously when they rip off program copies.

Easy Copies

Instant Software programs are relatively easy to copy, but there is a reason for this. I feel that users should be able to make a backup copy of the original in case something goes wrong—like pushing the record button, instead of play, in the heat of excitement. I also like to access a listing to make changes in a program. One of the things you lose when you protect programs is the ability to list it.

I am not naive enough to think that hobbyists won't steal programs now and then. I was hoping that a $10,000 reward might act as an incentive to report a theft that led to a conviction. And, if someone were convicted, that might act as a deterrent. But my offer won't do any good unless someone walks off with $10,000 in his hands—and someone else is convicted of program theft, which is a felony.

The law in this field is new. There are few precedents. I am convinced that program publishing is similar enough to book, magazine and music publishing that it is prosecutable.

It is a shame that some hobbyist will have to suffer the trauma of a jail sentence or fine for the benefit of this new industry, but someone may have to do it before program theft will be taken seriously.

The courts are rather protective of copyrights, so I'm anxious to get a few clearcut cases of program copying. The end result could be a real shot in the arm for the computer programmers, who are the biggest losers in program theft.

Inhibit Growth

Since the growth of the microcomputer business hinges on the availability of good software, unless something is done to eliminate program theft, the field's growth will slow. This benefits no one. A slow growth rate will

Continue to next page

EDIT 80

by Jim Perry

You can please some of the readers all of the time, all of the readers some of the time, but you can't please all of the readers all of the time.

It seems that *80* has displeased some readers in high places by printing the extracts from Harv Pennington's book *TRS-80 Disk & Other Mysteries* (February *80*). According to Radio Shack, no mention was made of the existence of their TRSDOS 2.3.

Suffice it to say that TRSDOS 2.3 was only a rumor when the book was written and had not yet been released when the article was prepared. As a result of this article, Radio Shack has canceled their advertising contract with *80*.

By way of rebuttal I would like to extract a portion of the introduction from Harv's excellent book. It sums up my feelings, beautifully.

On the whole the TRS-80 is a pretty neat machine. In fact, I love my TRS-80. Just a few short years ago, a computer with the power and capability of the TRS-80 would have cost several hundreds of thousands of dollars, required an air conditioned room of considerable proportions and a staff to operate it. Certainly, the Tandy Corporation deserves all the credit possible for the development, production and distribution of this magic machine. Tandy Corporation, I SALUTE YOU!

Like all large corporations, the Tandy Corporation, seems to have continued success in spite of itself. The initial success of the TRS-80 was, I suspect, beyond the wildest dreams of anybody at Tandy. Since there was no way to measure their success against a similar product, at a similar price and with similar distribution, who is to say how successful they really were. It is my contention that they were only about fifty to sixty percent as successful as they could have been!

Very quickly, as a result, an attitude of "don't-tell-us, we'll-tell-you" developed. The general quality of follow-on support, development and software was abysmal. Information about the workings of the system was (and is) a carefully and jealously guarded secret. It's as if "WE," the users, "couldn't possibly know a damn thing or figure it out" and only the High Priests of Fort Worth, when they deem it propitious, will tell us what we need to know.

So, if I tend to excoriate (a fancy word meaning, "give 'em hell") the Tandy Corporation (Radio Shack), it is only because I would like to see them turn around their damn superior attitude and realize that the thousands of you out there are doing more than you are being given credit for and should be listened to. Instead of 'THEM' telling you; 'YOU' tell them. ■

inhibit good hardware production. Stores will be limited, and people will stop writing programs.

My suggestion is that computer stores, including Radio Shack, keep a very close watch on demonstration systems and make sure that youngsters are not left free to dump copies of demo software onto their tapes or disks. Make sure that customers buying systems pay for each and every program taken.

Clubs should set a fixed rule regarding software: No copying at club meetings.

Copyright suits can be very expensive for violators. One of the big map companies makes several hundred-thousand dollars a year in judgements when people and firms make copies of their maps without realizing those maps are copyrighted. Many a club newsletter has taken a photocopy of a gasoline company road map and indicated the site of a meeting only to have the members sued en mass and individually by the map company.

I am quite serious about the $10,000 Instant Software, Inc. is offering and am anxious to pay it. As I write in the ISI booklets, when your best friend asks you to run off a copy of an ISI program you have to be damned sure that he wants the program more than the $10,000.

All of us want the same end result, a healthy, rapidly growing microcomputer industry. I've set the prices for Instant Software as low as possible to discourage ripoffs. But low prices are only possible with massive sales.

A programmer has to make several thousand dollars in royalties if he is going to continue to write more programs. I've set the standard for the field at 20 percent royalty: very high. I did that because I wanted to make program writing so profitable that we have a good supply of top-notch programs. This will be best for the industry, for programmers even program publishers, in the long run.

I'm open to any ideas, suggestions or arguments on this subject, but this should keep things going for a while.

80 Expo

Acting on an ad in a Boston paper, I drove down to see what Radio Shack was doing with their "TRS-80 Expo" at the Park Plaza Hotel. I found quite a crowd gathered in the grand ballroom.

The Radio Shack dealers were there in force with TRS-80 systems in special rooms showing specific programs, plus another 32 systems set up in the main ballroom. They were putting on shows every couple of hours for businessmen. The show started with a film introducing the TRS-80 and Radio Shack, then it went into demonstrations of the software which has so far been completed for the Model II. Most of the demo units were Model II systems.

I talked with some of the Radio Shack people.

The show was well done, and Radio Shack was taking orders at a pretty good clip as a result. A friend of mine accompanied me to the show and after thinking it over for a couple of days, ordered a 64K Model II with which he hopes to run his mailing list. ■

Robert Epstein, a Harvard doctoral candidate, using a TRS-80 in his research

Photo by Reese Fowler

UNLIMITED 80's

by Sherry Smythe

Looking for an unusual application of the TRS-80, took me to the world-renowned laboratory of Dr. B. F. Skinner of the Department of Psychology and Social Relations at prestigious Harvard University in Cambridge, MA. His lab is composed of offices, rooms with caged animals, minicomputer equipment and rooms containing racks of electronic relays whose wires run to a series of experimental chambers. Tucked away in what appears to be a large closet is a TRS-80 Level II with 32K RAM and an expansion interface wired to those same chambers.

Dr. Skinner is performing behavior modification experiments using pigeons, monitored by a TRS-80.

Monitored by Relays

Even at Harvard, grant money is tight. Robert Epstein, a doctoral candidate working with Skinner, needed some research done cheaply and quickly. He thought the TRS-80 was a logical alternative to expensive computer equipment for this sophisticated control application.

He is using interface equipment from Alpha Products Company and has added an 8-bit latching bi-directional port. By adding relays and a silent printer he monitors an experiment which can change 12 to 13 times per second.

Epstein's TRS-80 is running a special machine language program first published in *Behavior Research Methods and Instrumentation* that processes all information fed to it from the relays.

The purpose of Skinner's experiments is to discover how behavior is controlled. Within the chambers—specially designed observation enclosures—his captive birds manipulate some specific devices by pecking, eating, drinking, etc.

Picture an orange crate made of acrylic that houses two pigeons, one on each side of a transparent partition. Each pigeon is visible to the other, but pigeon 1 is the only one that can see a light flash, randomly green, red, or yellow. When the light flashes, pigeon 1 must translate the color to a coded symbol, marked on the wall clearly visible to pigeon 2.

The Experiment

Let's say green color has an X for the symbol. When the green color flashes, pigeon 1 must peck the X from three symbols. Pigeon 2 then pecks a box labeled Thank You and automatically feeds pigeon 1.

Pigeon 2 must then recall the symbol that pigeon 1 pecked and translate it to the proper color, in this case green, which appears in random order with red and yellow. If he pecks correctly on his first try, pigeon 2 gets fed as well.

After watching a few experiments controlled by the TRS-80, Epstein explained that the DEC equipment and all the other minicomputers in the lab, costing many times the price of the TRS-80, were all dedicated machines, but could do nothing except their one program. His TRS-80, when it had completed its function, was a partner for backgammon. Quoting Epstein about the 80, "This is causing a revolution in lab research!"

If you know of any TRS-80 applications that might interest our readers, please drop me a line. ■

80 ACCOUNTANT

by Michael Tannenbaum C.P.A.

The Model II operating system Version (1.2) seemed to work immediately.

At last the soul searching is over, I just received my new Model II. Compared to the painless way I slipped into ownership of the fully loaded Model I, the jolt was considerably greater. It's one thing to spend $899 and another to cough up $4,400.

Unfortunately, I realized that I had to spend $1,100 or more for the additional drive to have a workable system. At least I avoided the start up problems associated with TRSDOS 1.0, 1.1, 2.0 and 2.1. The Model II operating system (version 1.2) seemed to work immediately.

Compare Model II

I am sure that by this time almost every 80 fan has had an opportunity to compare the Model I to the Model II. Ownership, however, highlights some jarring differences. The first thing I noticed was the noise. The Model II has a very busy fan and a disk that runs constantly.

Another difference is the keyboard layout. Where Model I has 63 keys, the Model II has 76. Since the computer powers up in lowercase and the DOS requires uppercase, I had some difficulty entering commands. (Yes, I know it helps to read the manual.)

The next big surprise occurred when I called up BASIC. Compared to the Model I, BASIC takes considerable time to load in the Model II. Once loaded, the Model II executes an automatic "? MEM" and indicates the number of files requested. Surprise—unless you designate files, none will be allocated. There is no default as there is in the Model I.

After reading the manual, I found a BASIC calling sequence very similar to NEWDOS. Both the files and the memory reservation have to be specified when the BASIC interpreter program is requested.

I have an old Line Printer One. Because of the lack of motor cutoff, I got into the habit of leaving the printer off until it was required. For that reason, all of my programs PEEKed at the device control block to check the printer's status.

Since the Model II does not have the ability to PEEK or POKE memory locations, this method of testing printer readiness is not available. If the printer is not ready, you receive an I/O error message. When this occurs, you must execute a SYSTEM "FORMS" command. BASIC will temporarily transfer control to the DOS and you can alter or specify print parameters desired.

Transfer Program

You should plan to keep your Model I for some time to transfer all that good software. Radio Shack has thoughtfully developed a Transfer program for the Model II and a Com-DOS program for the Model I. With these programs, an RS232 board in the Model I and a special cable, it is possible to transfer disk files to the Model II.

Aside from some minor problems in the disk I/O routines, most of the programs that I have selected to transfer were converted without much difficulty. Because of this, I find it surprising that in so many instances there are such price differentials between Model I and Model II versions of a given program. Some programs are highly integrated into the Model I's architecture and will not function on a Model II without a significant revision.

The major problem I had with my new Model II (aside from PEEKs, POKEs, and PRINTs) was careless programming on my part. I had gotten into the habit of leaving out the "THEN" in an "IF-THEN" statement. This syntax violation is not tolerated by the Model II. If I had been consistant with the error, I could have saved myself a lot of time. Unfortunately, sometimes I put it in and sometimes I left it out.

Backup may be another stumbling block. You cannot just duplicate a disk. Backup on the Model II is a two-step procedure. First you must "FORMAT" the blank diskette and then you may execute the "BACKUP" program. Better be patient. A backup procedure involving a 500K diskette is a much slower procedure than on the Model I. Even if you have two drives on your system it will take several minutes. If you have only one drive you may have to insert the backup copy and remove the source up to 10 times.

Enough teething problems. With the big disks and a full speed CPU of the Model II, accounting applications are a natural for almost any size business.

Series One Accounting

A good example of this new wave application software is the Radio Shack, Accounts Receivable-Series One (Cat. #26-4554), developed by Retail Sciences, Inc., for the ill-fated Tandy 10.

Since it is designed for a one drive system, the capacity is somewhat limited. A maximum of 800 customer accounts are permitted. The system uses a single file for both the customer master records and transaction details. Since the file capacity is 1743 records, using the maximum number of customer accounts reduces the capacity of the system.

For example, if you have only 200 accounts, you will be able to accommodate up to 1543 open items or an average of a little over seven per account. For 800 accounts only 943 transactions are available, which is only a little over one open item per account. Since a typical account could have an invoice, a payment and a credit transaction posted to complete a sale, this is an unhealthy situation.

The designers of the system were aware of this problem. For that reason, they allow the user to specify either a balance-forward or open-item method for each account. This feature is rather unusual for receivables systems. Most require the same method for all accounts.

Another unusual feature is a provision for "automatic" billing. It allows you to specify an automatic amount to bill on a regular basis.

The package also includes an invoicing module. The module eliminates the need for a separate billing program and if utilized, can be very useful in reducing invoice preparation time.

The program has several automatic features which include extracting the billing address from the receivable files. This address can be designated as the "Ship to Address" or a separate "Ship to Address" can be specified if desired. Automatic selection of the date and "Net 30 days" terms are also provided. Additional features include automatic extensions, tax calculations and invoice totalling.

The invoicing module only provides input to the accounts receivable ledger. However, because the program documentation is excellent, modification should be possible to generate a data file which could be used for sales analysis or inventory maintenance purposes. This will require modification of the disk file structure supplied by the Shack, since the system disk has no room available for additional files. This lack of memory space is because of the file preallocation facility of TRSDOS which speeds up access to the records in the data file. Reducing file allocation to accommodate invoice details, detracts from the performance of the receivable system.

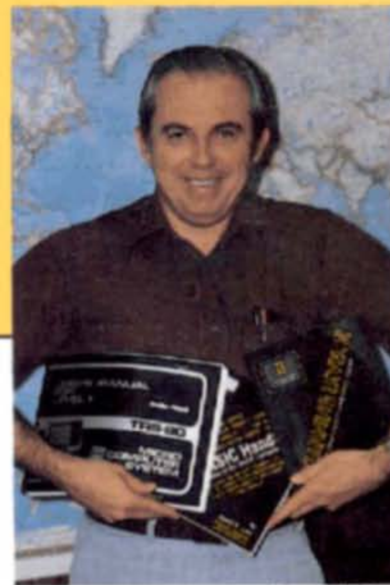
Because the system was originally intended for the two-drive Tandy 10, it is possible to separate the data and systems diskettes. If this is done, an invoice detail file can be created on the system's disk for subsequent analysis.

The package represents a significant improvement over the Radio Shack products reviewed to date. Not only is the user documentation excellent, but the program listings contain remarks which identify each subroutine. File descriptions, indicating variable assignments and field length, are also provided. This data makes modifying the system to fit individual needs considerably easier.

If the Accounts Receivable-Series One is representative of the Model II software that will be forthcoming, I should be very happy that I purchased the machine. ■

✓Reader Service—see page 162

80 REVIEWS



David Lien, author.

Photo by Jim Perry

With all these shortcomings, do I recommend the book? Yes, without hesitation.

Learning Level II
David A. Lien
Compusoft Publishing
San Diego, CA, 1979
352 pages, softcover
$15.95

by Everett Ogden

David Lien's *Learning Level II* is written for people who have mastered Level I as well as for people who have just bought a Level II machine and know nothing about computer programming.

Lien's book does not repeat the material in the Level I manual; instead it includes corrections and a double set of cut-and-paste pages to convert that manual to Level II. It covers the new capabilities of Level II in the style of the Level I manual.

The material is about the same as that in Radio Shack's Level II reference manual, but it's arranged for teaching, not reference. For limited reference use there is a single-page index.

The book doesn't cover everything. The DELETE command is omitted, as are RANDOM, STRING$, CDBL, CINT and CSNG. Except for USR, the rest are covered to my satisfaction.

Lien rightly avoids discussion of machine language as much as possible, but I'd like more on communicating between machine language and BASIC for those who have some familiarity with both.

Particularly well covered are the editor (two chapters), PRINT USING (two chapters, including an exponential format my older manual mentions only in its statement summary), strings (six chapters and parts of others) and error messages (one page each).

Real Complaint

My only real complaint about the book is the number of typographical errors in it. Most are in the text. These are obvious and won't cause any trouble. But many of the typos are in the programs or sample runs where they might confuse beginners.

I had planned to come down hard on typos in this review, but a few days after I got the book I received an errata sheet from Compusoft with over 40 corrections. It listed all the important errors I had caught and many more I'd missed.

There are some other problems as well. In the section on converting Level I tapes to Level II, Lien says that the lack of abbreviations in Level II will lengthen programs and may make individual lines too long. This is not true.

Each Level I abbreviation takes at least two bytes. Its Level II counterpart is stored in one byte and is only spread out to the full word when listed on the screen or printer.

Also, since all Level I words are less than three times as long as their abbreviations, a converted line's listing would be less than three times as long as the original. Since Level I permits 70 characters per line, and Level II allows 255, there is ample space for conversion.

Later Lien suggests that the data conversion tape is so complicated that it's easier to write a Level I program to list the data on the screen, then write it down and re-enter it into your converted Level II program. Few people own both Level I-Level II systems, so it's not a very practical suggestion.

Recommended Reading

With all these shortcomings, do I recommend the book? Yes, without hesitation. If you are upgrading from Level I, you will get your money's worth. If you are starting on Level II with no previous computer experience, this book is almost mandatory.

Make sure you get a copy of the errata sheet. This should be automatic if you buy direct from the publisher, which may be the only way you can buy it. I haven't seen Compusoft books in any store.

Lien is an excellent teacher and writer. What Compusoft needs now is an excellent proofreader. ■

TRS-80 Micro Computer Technical Reference Handbook
Radio Shack, 1978
Fort Worth, Texas
$9.95

by James R. Fatz

Have you ever wondered exactly what was going on inside that sealed keyboard of your TRS-80? I certainly have!

The *TRS-80 Handbook* has put an end to my guessing. Written clearly and simply, anyone who has an understanding of hardware nomenclature will be able to keep pace with the book's technical data. Although the book is intended for a technical audience, it offers the inquisitive 80 owner information on testing and analyzing.

The book begins by displaying a dual page foldout of a system block diagram. This diagram puts the interactive circuits in perspective before the reader gets into a great amount of detail. The next page is a memory map of the Level I system identical to that contained in the Users Manual.

The first section, "Theory of Operation," contains descriptive paragraphs on: CPU Address Lines, CPU Data Bus, CPU Control Group, System RAM, Video Divider Chain, Video RAMs, Video Processing, Keyboard, I/O Ports, System Power Supply and Level II ROMs.

All these sections contain a detailed analysis of exactly what occurs, pin by pin, in the appropriate circuitry during system operation. For example I was always mystified by the appearance of double-width characters on the screen. I discovered that this is normal when the system is placed in thirty-two character format.

The second section, "Adjustments and Troubleshooting," is technical but interesting. I was surprised to find that the intermittent problems that I was having, were most likely caused by a solder ball rolling around. I eliminated them by placing the keyboard unit on end and tapping it.

In the "Outside World" section a short outline is given on how to interface the TRS-80 with analog devices. A sample program shows how to control power to a coffeepot.

Although short, the description of the Expansion Port Edge Card pin connections is adequate and essential for "outside world" expansion.

A complete Parts Listing makes up the next section. Divided into item and function sections these listings include the card symbols, electrical descriptions and part numbers. (An invaluable aid to the do-it-yourselfer.)

The final section contains schematics for the entire system. These include a Level II BASIC schematic and a complete TRS-80 schematic on two six-page foldouts. One thing that I find particularly useful is an exact copy of circuit board layout, which, unfortunately, is not included in the users manual. ■

Problem Solving and
Structured Programming in BASIC
Elliot B. Koffman and Frank L. Friedman
Addison-Wesley, 1979
$11.50

By Mikel Aickin

Intertwined with a vacation last summer I spent some time combing bookstore shelves, in search of a book to make learning BASIC painless. My search revealed four general categories: those which assume that you are a moron, unable to find distal portions of your own anatomy; those ground out by the Fellowship of Dusty Textbook Writers; those which are terminology bound; and *Problem Solving and Structured Programming in BASIC* by Koffman and Friedman.

Very Readable

I put this book in a class by itself because that is where it belongs. An introduction to both computing and BASIC, it is a reference worth having on your shelf. It is also readable. I began by skimming the earlier sections, but before long I was reading intently.

First of all, the book is well printed. The text is sharp, the diagrams, figures and programs copious. Full and half-tone shading emphasize logical relationships. However, learning a computer language without a computer is rather like learning a foreign language with only a Berlitz record as a companion, that's why the book's graphic displays and examples are extremely important.

Each chapter follows the same format. After an introductory section establishes the purpose and general viewpoint, a small number of concepts and language structures are discussed in great detail. Each structure is shown enclosed in a box with its syntax and a precise description of its purpose.

A section of common programming errors—unbelievably helpful—a summary of the topics covered in the chapter, and some exercises (solutions are provided at the end of the book) round out the chapter.

A Collection of Dialects

Since BASIC is not so much a language as a collection of dialects, you may be wondering which BASIC is presented. There are two answers.

By treating the general operations that any BASIC is designed to perform, the authors show how BASIC works, regardless of whether your particular version differs in certain respects. The emphasized topics, top-down and structured programming, are language independent.

Secondly, the authors discuss three dialects: minimal BASIC; Dartmouth BASIC; and BASIC-PLUS (the Digital Equipment Corporation variety).

Although this leaves out some features available on personal computers (PEEK and POKE, for example) and includes some not found in small BASICs (functions and subroutines, for example), once you have read this book you will be more prepared to read your system manual.

I have a few complaints with some parts of the book. The IF-THEN and IF-THEN-ELSE statements, for example, are presented as identical in all three versions of BASIC, but, they are not. This is only minor.

With *Problem Solving* by Koffman and Friedman, my BASIC is becoming fluent. ■

Pascal: An Introduction to
Methodical Programming
by William Findlay and David Watt
Computer Science Press
Potomac, MD
306 pages, paperback
$10.95

by Chris Brown
***80* Staff**

Don't be misled by the title of this book. It is not strictly a Pascal handbook. The main purpose of the book is to teach the reader the techniques of methodical programming. The achievement of fluency in Pascal is actually a secondary objective.

What, you ask, is methodical programming? A good question. As far as I can tell, it is synonymous with logical programming. Because it is cross-cultural, this confusion in terms is the essential problem I had with the book.

The Common Tongue

That comment is not necessarily a condemnation, but it should be taken as a warning. The edition, intended for use in an elementary programming course, was authored by two gentlemen from Scotland. Messers Findlay and Watt are lecturers in computing science at the University of Glasgow, and their approach to computing is more European than American. For instance, the authors make the assumption that Pascal is the *lingua franca* (the common tongue) of computing. This claim reveals the author's computing bias. In American computing, if a common tongue exists at all, it is probably BASIC.

Aside from furthering the annoying practice of including foreign buzzwords in English language texts, the book assumes the reader has no programming experience or mathematical expertise. However, most students of Pascal in this country will have already had some experience with computers. Starting from square one, may actually inhibit their progress.

And remember this book is intended for classroom use. The style is businesslike, but for readers who learn their programming at the School Of Hard Knocks rather than the local university, it may not be the most efficient format.

Non-essential Topics

The text is divided into six sections. Each contains chapters that include exercises (with answers). Topics that are not essential to understanding are marked and can be skipped on first reading.

This technique is symptomatic of the time constraints within which most college courses operate. However, if material is only tangential to the topic under discussion, why include it in the first place (especially in an introductory discussion).

The book does include five appendixes dealing with such Pascal idiosyncrasies as syntax diagrams, reserved words, predeclared entities, input and output, etc. These are most useful.

The book was produced by computer and as result the typeface and page layouts are rather primitive. Texts generated by computer are, in general, hard to read. While computers certainly have their place in our society, it is not yet in book production.

The $10.95 price is reasonable in light of the outrageous sums routinely charged for limited edition, short lived college texts. The book's cost effectiveness for the independent hobbyist is another question. I think one can do better in an introductory Pascal edition than this Scottish college text. ■

57 Practical Programs & Games in BASIC
Ken Tracton
Radio Shack, 1978
Catalog #62-2008
$3.95

by James R. Fatz

Finally! After so much time, the TRS-80 system that you saw in the store window is on your desk waiting to do something useful. You already found out that it is awfully expensive buying pre-programmed tapes. Now what?

You might try *57 Practical Programs & Games in BASIC*, but don't hope for much, even if it is from Radio Shack.

Although the book contains a wide variety of programs from Moments, Skewness and Kurtosis to Space Wars, many of the programs simply do not work. Whether this is due to program logic or misprintings the result is the same —*frustration*.

The errors contained in the programs range from the obvious, like a missing parenthesis, to very obscure ones such as reversed variables in assignment lines.

One good point is that the programs are flowcharted adequately. This can help you learn the relationship between program logic and flowcharting principles. It's too bad that more time was not spent proofreading than flowcharting.

So beware. There are many special interest programs in this book, but it may take some debugging to get them to work on the 80. ■

BASIC-1P
Richard H. Shubert
Small Systems Software
Newbury Park, CA
$19.95

by Fred Blechman

Many TRS-80 owners start out with a Level I machine, as I did, to learn programming and familiarize themselves with a microcomputer. However, even after buying so many Level I tapes and creating a number of Level I programs of my own, I couldn't resist the lure of Level II. After all, according to what I'd heard, a conversion tape supplied by Radio Shack with Level II would still allow me to use all my Level I programs.

So I converted to Level II, only to find that the conversion tape supplied only converted Level I BASIC tapes, not machine language. Furthermore, some programs required considerable changes before they'd RUN properly in Level II!

Finally, the conversion tape expanded almost all the Level I abbreviations, because Level II doesn't allow them, so my listings and printouts from my newly acquired printer were no longer in Level I abbreviated language! While this might not bother some people, it made it difficult for me to author articles on Level I programs without resorting to my typewriter for an abbreviated Level I listing.

Enter BASIC-1P

I tolerated this until recently, when Richard H. Shubert created his BASIC-1P tape from Small Systems Software. This program provides full TRS-80 Level I BASIC capability for Level II 16K (and up) and also adds two Level I commands, LLIST and LPRINT, so you can now list your programs and control your printer from Level I BASIC. Schubert's program contains two additional commands that even Level II BASIC doesn't have, LPRINTON and LPRINTOFF, that allow you to print characters displayed on the screen!

BASIC-1P is a machine language tape that loads into the top 4K of RAM with a SYSTEM command. If your machine has a 32 or 48K memory, BASIC-1P loads at the same location (decimal 28327) about 11K up from the bottom of RAM. This allows you 11264 bytes of usable RAM when in the Level I mode.

BASIC-1P is compatible with most any printer and includes driver software.

BASIC-1P documentation is excellent. The program executes automatically and asks you to select a printer interface option: RS-232-C, TRS 232, Centronics or no printer. Depending on which option you select, you are asked for a few simple inputs (line feeds required?, nulls required?, baud rate?) that are appropriate for your interface.

Now a READY appears on the screen, and the computer acts like a Level I 11K machine in almost all respects. In addition, you can LLIST the program in Level I "language" which even Level II can't do!

You can add LPRINT statements to your programs to have them appear on the printer, which Level I can't do. If you enter LPRINTON at the keyboard or in the program, all PRINT statements will appear on both the screen and printer until the computer encounters an LPRINTOFF command from either the keyboard, or in the program.

To return to Level II BASIC, type CMD"S". (Your Level I program will be lost.) The screen responds with MEMORY SIZE?. If you want to save BASIC-1P, type 28327 and ENTER. This leaves you with 11130 usable bytes of RAM in Level II. To return to BASIC-1P without reloading, type SYSTEM and press ENTER, followed by /28327 and ENTER.

BASIC-1P comes with disk instructions as well.

BASIC-1P is a marvelous, useful tape but it has a couple of limitations. Since it was designed as a resident interpreter in high RAM to act like a Level I BASIC ROM, it does not accomodate most Level I machine language programs. Some will load and run. If in doubt, CLOAD the tape and try.

Other Level I machine language programs will "freeze" the keyboard. If this happens, push the keyboard reset button (left rear of keyboard unit) and you'll find yourself back in BASIC-1P. The very worst is that the machine language tape will bomb-out BASIC-1P, and you'll be back in Level II with the MEMORY SIZE? prompt.

The other limitation is that Level I programs exceeding 11264 bytes do not fit into the memory available when BASIC-1P is resident. The entry point of BASIC-1P in RAM (decimal 28327) is not controlled by the user, but is programmed with the machine language. ■

The Programmer's Book of Rules
by George Ledin Jr. and Victor Ledin
Lifetime Learning Publications
Div. Wadsworth Publishing Company, Inc.
Belmont, CA
September, 1979. 256 pages
$7.95 Paperback

by Chris Brown
***80* Staff**

English language writers and editors, wrestling with syntax, turn, in times of desperation, to a slim book of usage called *The Elements of Style*.

Written in 1918 by an odd little man named William Strunk, this book of rules states, in no uncertain terms, exactly what is and is not allowed when constructing the English language (according to Strunk). Called affectionately, "Strunk," by its users, the book has become legendary in composition classes and editorial offices.

Composing Is Composing

Since composing computer language is similar to composing human language, it is not surprising that *The Programmer's Book of Rules*, by George & Victor Ledin, is the computerese equivalent of Strunk's 1918 edition.

The *Programmer's Book of Rules* is a list of 272 programming do's and don'ts. Co-authors George and Victor Ledin are California-based educators. George Ledin is the head of the Department of Computer Science at the University of San Francisco. His brother, Victor, has taught computer programming at U.C. Berkley, and for Intel in Santa Clara.

Despite their academic backgrounds, the authors have maintained a real-world outlook on programming as a sociological, not just technical, exercise. Their concern for client, as well as programmer, is laudable.

The *Book of Rules* states simply and concisely what the authors feel is essential to the art/science of programming. In terms reminiscent of old William Strunk, the authors exhort the reader to "Make sure all variables have different names."

"Avoid multiple, unconditional, jumps," they write. "Code carefully . . . write a reference manual, . . . choose the right language for the job, . . . edit your program."

Some of these canons border the obvious, others, seem trivial, but the solid footing of the author's opinions is undeniable.

The layout of the book facilitates its use. Pairs of facing pages work together presenting rules of programming on the left, while right-hand pages provide examples and detailed discussion.

Each chapter contains a summary of reference material ranging from the academic (*Software Engineering—An Advanced Course*), to the esoteric (*SIGPLAN Notices*, vol. 9, no. 11). A bibliography and complete subject index are supplied, and the thoughtful inclusion of an Author Index further simplifies its organization.

Shirt-Sleeve Reference

The book is well-produced. In this day of quick and dirty computer books, it is encouraging to see a carefully prepared edition that sells for a fair and reasonable price.

The *Programmers Book of Rules* is a shirt-sleeve reference manual. Its appeal is universal and its authors pull no punches, mince no words in their presentation. If sound advice and definitive opinion is what you are looking for, you'll find it in *The Programmers Book of Rules*. And while your at it, pick up a copy of Strunk's book. You'll need it when you write your documentation. ■

Maxi-Disk and Shuffleboard
Parasitic Engineering
Berkeley, CA
Maxi-Disk: $995
Shuffleboard: $249

by Kevin Cohan

Two products now available from Parasitic Engineering, Berkely, CA, add storage and CP/M capability to your 80.

Adding Parasitic's Maxi-Disk gives the TRS-80 owner full size floppy drives that more than triple your storage capacity per drive. Furthermore, installing Parasitic's Shuffleboard in a Maxi-Disk system converts your TRS-80 into a true Z-80 CP/M with access to either system through a simple command.

Using eight-inch floppies with a TRS-80 requires both hardware and software modifications. Parasitic's $995 Maxi-Disk package has everything you'll need to make the alteration: one full-sized floppy drive, a small circuit board, an eight inch Maxi-DOS disk, containing patches to enable TRSDOS 2.1, 2.2, 2.3 or NEWDOS to run with the new hardware.

Installing the Maxi-Disk circuit board requires opening the expansion interface. This voids the official Radio Shack warranty, but it is worth it when you consider that after the installation, you will have 256K of disk space on line with one Maxi-Disk drive, and that up to four may be installed.

After opening the expansion interface, you need only unplug the disk controller chip. The Maxi-Disk board plugs directly into the socket in which the controller resided. Photographs included with the instructions show you the correct installation. That's it!

With TRSDOS you have the option of combining 5¼,and 8-inch drives in any order. For example, you may choose to have a mini drive as drive #0 to store programs and a Maxi drive as drive #1 to store associated data bases on full-sized floppies. For CPM all drives must be 8-inch.

Software Modification

Software modification is next. Here you must patch the disk operating system (any TRSDOS or NEWDOS) so that it "knows" about the new hardware.

To do this, Parasitic provides a disk program called DOSPATCH. Though the final configuration of drives is up to the user, to initialize the system, drive #0 must be a Maxi and drive #1 must be a Mini.

Running the DOSPATCH program generates the Maxi-DOS system (drive #0) disk from your original system disk. After this is done, the drives may be reconnected in the chosen order and backups made of the new system disk.

For an additional $249, you can obtain Parasitic's Shuffleboard. This device turns a Maxi-Disk TRS-80 system into a dual DOS computer incorporating TRSDOS and Z-80 CP/M. CP/M is a disk operating system, used with eight-inch floppies. It is universally applied and your programs will run on any other CP/M system. Consequently, more software has been written for the CP/M than any other system.

To support CP/M, you need to install another small keyboard circuit, the Shuffleboard. Unplug the Z-80 CPU chip, plug the Shuffleboard into the CPU socket and plug the Z-80 chip into the empty socket on the Shuffleboard.

From now on, when the computer is used as a CP/M system, the CPU ignores the first 16K of memory space (except for the video RAM), regards the second 16K as the first, the third as the second and so on.

You now have a true CP/M system, with a maximum memory space of 48K. True CP/M requires 256 bytes of RAM starting at location 0000H, though the bulk of the DOS resides at the top of the memory space.

These locations in low RAM contain critical I/O device information, as well as system entry vectors. Since the ROM in a TRS-80 occupies the first 12K of memory, those pseudo CP/M systems must locate this crucial system information somewhere else. Consequently, any attempt to run software which looks directly for the vectors in low RAM will cause a major crash. The Shuffleboard eliminates this problem.

Two modes are jumper selectable on the Shuffleboard. Mode A always boots the system into TRSDOS. CP/M is accessed by typing in the command CPM and inserting the CP/M system disk supplied with the Shuffleboard.

Mode B is exactly the opposite. The system boots into CP/M, and TRSDOS is accessed by the command TRSDOS. Unfortunately, if a fatal crash occurs while in the DOS not selected by the jumper, the system disk must be removed and a reset performed with the other system disk.

Your total investment to turn an ordinary TRS-80 into a Maxi-Disk TRS-80/ Z-80 CP/M system is $1244. The Maxi-Disk installation may be used alone, but it is the addition of Shuffleboard that makes this set-up valuable.

If you are hesitant about spending this kind of money on your system consider that by adding this hardware to the TRS-80, not only is the disk storage capacity per drive tripled, but you also have access to an unrivaled new software realm. ■

Percom Disk Drives
Percom Data, Inc.
Garland, TX
Disk Drive: $795

by Walter C. June

Most of you reading the microcomputer trade magazines are familiar with the name Percom Data Company, Inc. Their advertisement probably appears in this issue.

I talked to other companies who offered drives at about the same price, but power supplies, enclosures and cables weren't included. Furthermore, none of the companies committed themselves to delivery time.

I called Percom twice before ordering my system. I couldn't believe how simply they packaged the system and called back to make sure I had not overlooked anything. Yes, they assured me, their systems included the drive(s), power supplies, enclosure, connecting cable and shipping. (Percom is now charging for cable.) They promised delivery in six to eight weeks so I ordered a dual drive TFD-100 system.

On the seventh week the cases arrived. For my system they shipped two WANGCO model 82 drives, although you might also receive Pertec FD-200 drives. Both drives support 40-track operation and have dual sense lamps so you can record on both sides of your diskettes without having to punch extra holes in the jackets.

The drives are packed separately in their own protective cartons and some assembly is required. The instructions make assembly a snap. First you determine which drive you will use as DRIVE 0 and remove the terminating resistor from the other drives. Then you mount the drives into the enclosure with the machine screws provided and plug in the lead from the power supply. Plug in the connecting cable and replace the top of the enclosure. In fifteen minutes you're done.

From my ignorance of floppy disks and how they worked I had this fear in the back of my mind that buying a 40 track drive while Radio Shack used 35 track drives would make my system incompatible. Percom assured me there was no problem but I didn't believe them until I placed my Radio Shack DOS 2.1 diskette into my new DRIVE 0 and booted the system. It came up the first time without a hitch and I felt much better. The five extra tracks on my Percom are additional ones that Radio Shack drives are not physically able to reach. The standard DOS doesn't even know they exist.

But why give up that extra fourteen percent capacity on each drive?

You don't.

Percom sends their PATCHPAK diskette with each system which tells DOS how to handle 40 track drives. This PATCHPAK requires two drives to run, but once updated DOS handles 40 track drives just as well as it did the 35 track drives.

Though Percom now charges for their connection cable, for $24.95-$39.95—depending on the number of drives—I recommend *it* to complement, as near as I can tell, the perfect system. ■

THE ASSEMBLY LINE

by William Barden, Jr.

Relocatability is a main ingredient for constructing assembly language routines . . . in BASIC programs.

In this month's column we're going to discuss two interesting topics: "relocatability" of Z-80 instructions and programs; and a way to read the TRS-80 Model I keyboard.

Relocatability is a main ingredient for constructing assembly language routines that can be embedded in BASIC programs. The second subject will let us read every key with a fairly simple assembly language routine.

Are You Free to Relocate?

Relocatability is one of those mysterious Z-80 buzz words that everybody uses. When used to describe assembly language code, relocatability simply means a given instruction will execute properly anywhere in memory. To examine this idea, you'll need to know something about instruction formats. Pick up that copy of *TRS-80 Assembly-Language Programming* and do some rereading. I'll wait . . . .

Z-80 instructions have many formats. Some are only one-byte long specifying a simple action, such as setting the carry flag (SCF) or complementing the A register (CPL). These instructions contain no address fields or other operands. They execute anywhere in memory.

Another type of instruction uses a field, or fields, within the instruction to define a CPU register or register pair. An example of this type of register addressing is a LD A,C which uses a field for the source register (C) and destination register (A). This type of instruction also executes anywhere in memory.

Another common type of addressing is immediate addressing, in which the data is immediate within the instruction itself. Since the data is constant data, the instruction appears the same anywhere in ROM or RAM. The instructions LD A,23 and LD HL,3C00H are examples of immediate addressing.

There is no problem when any of the register pairs or index registers are used to point to a memory location. An ADD A,(HL) or LD A,(IX + 5) has the same appearance whether it is loaded at location 25 or 2500.

But some instructions do change when loaded into different areas of memory. In general, any instruction that specifies a memory address where the operand for the instruction is located does change.

If we were to load a variable into the A register, for example, we might have the instruction LD A,(40EEH) or LD A,(VARP). The second and third bytes of the LD would specify a memory address. When this memory address remains constant, there is no problem.

For example, we could load variables into the A register by LD A,(40EEH) instructions anywhere in memory.

A Variable within the Program

A problem develops, however, when the variable to be used was located within our own program. Suppose that we use the code shown in Program Listing 1.

```
       ORG    8000H      ;ORIGIN
START  LD     A,(VAR1)   ;LOAD VARIABLE
       CALL   SUBS       ;CALL CONVERSION
       JP     NEXT       ;GO ON
VAR1   DEFB   23         ;HOLDS CURRENT # of
                          TRS-80s IN YUMA
```

Program Listing 1.

When we assemble this code (which is incomplete), the LD A,(VAR1) is assembled as 3AH (operation code for LD A) in the first byte and the address value for VAR1 in bytes 2 and 3. In this case, the address of VAR1 would be at location 8009H as the LD, CALL, and JP instructions each take three bytes.

This code runs only at the 8000H area. If it were relocated anywhere else, the LD A,(VAR1) would attempt to load VAR1 from 8009H, instead of nine bytes from the start of the new area!

An equivalent condition would exist for JP (Jump) instructions, where the jump address is in bytes 2 and 3 of the instruction, and for CALLs. The same problem would also be present for an immediate instruction, such as LD HL,TABLE, specifying an address of a variable (rather than a constant).

Any time that an instruction specifies a direct memory address that refers to a location not fixed in memory, the instruction cannot be relocated and executed. Instructions that are not relocatable are: conditional and unconditional JPs; CALLs; loads that use a direct address; and immediate loads that specify a variable address.

JR (Jump Relative) jumps are relocatable because they do not specify a direct memory address, but a displacement value that is referenced to the current location of the instruction. This means that any subroutine that contains nonrelocatable instructions cannot be used as embedded assembly code in BASIC programs.

The reason for this, of course, is that we generally don't know in advance where the dummy string for BASIC will reside. If we did, we could ORG (Origin) the assembly code for that spot. But, the string may be in various places dependent upon other strings and BASIC statements.

Without first reassembling a new Origin, we cannot move assembly code ORGed at one area into another if the code contains instructions with direct addresses.

It is possible to make a completely relocatable assembly language subroutine for use in BASIC dummy strings or SYSTEM type operations, as long as JRs are used in place of JPs. No CALLs to areas within the subroutine can be used, nor variables within the program. An example is shown in Program Listing 2 in a subroutine that whites out the screen. It is embedded in Level II BASIC as a dummy string by the use of CHR$.

```
0000 21003C    00100        LD     HL,3C00H    ;START OF SCREEN
0003 3EBF      00110 LOOP   LD     A,0BFH      ;GRAPHICS ALL-ON
0005 77        00120        LD     (HL),A      ;WHITE OUT ONE CHARACTER
0006 23        00130        INC    HL          ;POINT TO NEXT
0007 7C        00140        LD     A,H         ;GET MS BYTE OF ADDRESS
0008 FE40      00150        CP     40H         ;TEST FOR END
000A 20F7      00160        JR     NZ,LOOP     ;GO IF NOT END
000C C9        00170        RET                ;RETURN TO BASIC
0000           00180        END
00000 TOTAL ERRORS

100 REM THIS IS A SAMPLE CALL TO THE WHITE OUT SUBROUTINE.
110 REM THE ASSEMPLY-LANGUAGE SUBROUTINE IS RELOCATABLE AND
120 REM IN DUMMY STRING A$.
130 A$=CHR$(33)+CHR$(0)+CHR$(60)+CHR$(62)+CHR$(191)+CHR$(119)+CHR$(35)
    +CHR$(124)+CHR$(254)+CHR$(64)+CHR$(32)+CHR$(247)+CHR$(201)
140 B=VARPTR(A$)
150 POKE 16526,PEEK(B+1)
160 POKE 16527,PEEK(B+2)
170 C=USR(0)
180 GOTO 180
```

Program Listing 2.

Automated Relocation

It's possible to relocate code automatically.

Continue to next page

This technique, used on larger systems, assembles a machine program and the object code is output to disk. The object code in this case contains not only the machine language form of the instructions, but also information relating to the origin, relocatability or absoluteness of the instructions and data and other load items.

The object code by means of a special loader examines each load item and loads the data associated with it. If the data is specified as relocatable, such as a JP or CALL instruction, a load bias is added to the direct address portion of the instruction. This produces the proper address in memory where the instructions are to be loaded. The job of relocating the machine code, then, is done by the loader. The assembly program does not have to be reassembled for each new memory area where it is to execute.

Can we relocate existing machine language code that has no source listing? Yes, but the process is tedious.

It is possible to make a completely relocatable assembly language subroutine for use in BASIC Dummy strings . . .

First, the code must be disassembled by a disassembler such as Small System Software's RSM series or Apparat's DISASSEM. Then, all JPs, CALLs and loads with references to variables in locations that are not fixed must be changed. These references must be given addresses corresponding to the locations for the new area. It's easy to make a mistake while doing this. Also, it is not easy to determine which references to variables are to fixed locations and which are to locations that will be relocated to new areas.

An alternative method is to reenter the source code, based on the disassembled listing, with the original ORG to see if it assembles to the original machine code. Then, reassemble with a new origin.

Project Number 2

Learning to play the TRS-80 Model I keyboard is a great deal less demanding than playing Bach or Garner on 88 keys. First of all, we have only 53 keys (count 'em), not including the numeric keypad on some of your systems. All of these keys are active keys and generate a signal. In fact, the keyboard forms a matrix (array) as shown in Fig. 1. The matrix has eight rows and eight columns. Associated with each row is a row address of hexadecimal: 3801, 3802, 3804, 3808, 3810, 3820, 3840, and 3880.

To address a row, a Z-80 LD instruction with a row address is executed. The LD instruction

Continue to next page

ROW	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0	ADDRESS
0	G	F	E	D	C	B	A	@	3801H
1	O	N	M	L	K	J	I	H	3802H
2	W	V	U	T	S	R	Q	P	3804H
3	/	/	/	/	/	Z	Y	X	3808H
4	' 7	& 6	% 5	$ 4	# 3	" 2	! 1	0	3810H
5	? /	> .	= -	< ,	+ ;	* :	) 9	(8	3820H
6	SPACE	→	←	↓	↑	BREAK	CLEAR	ENTER	3840H
7	/	/	/	/	/	/	/	SHIFT	3880H

Fig. 1. Keyboard Matrix

```
F000          00140          ORG    0F000H       ;CHANGE THIS FOR YOUR SYSTEM
F000 CD0BF0   00150 CALLKB   CALL   READKB       ;READ KEYBOARD
F003 B7       00160          OR     A            ;TEST FOR NULL
F004 28FA     00170          JR     Z,CALLKB     ;LOOP IF NULL
F006 32203E   00180          LD     (3E20H),A    ;DISPLAY ON VIDEO
F009 18F5     00190          JR     CALLKB       ;LOOP FOR NEXT CHAR
              00195 ;
              00200 ;SCAN 7 ROWS
F00B 210138   00210 READKB   LD     HL,3801H     ;ROW 0 ADDRESS
F00E 7E       00220 READ10   LD     A,(HL)       ;GET ROW VALUE
F00F B7       00230          OR     A            ;TEST VALUE
F010 2005     00240          JR     NZ,READ20    ;GO IF KEY PRESS
F012 CB25     00250          SLA    L            ;SHIFT ROW ADDRESS
F014 F8       00260          RET    M            ;RETURN IF NO PRESS
F015 18F7     00270          JR     READ10       ;MORE ROWS TO GO
              00280 ;CONVERT ROW, COLUMN TO INDEX
F017 4F       00290 READ20   LD     C,A          ;ROW VALUE
F018 AF       00300          XOR    A            ;ZERO A
F019 CB3D     00310 READ25   SRL    L            ;SHIFT ADDRESS
F01B 3804     00320          JR     C,READ35     ;GO IF DONE
F01D C608     00330 READ30   ADD    A,8          ;ROW*8
F01F 18F8     00340          JR     READ25       ;CONTINUE
F021 06FF     00350 READ35   LD     B,0FFH       ;COLUMN COUNT
F023 04       00360 READ40   INC    B            ;BUMP COUNT
F024 CB39     00370          SRL    C            ;SHIFT ROW VALUE
F026 30FB     00380          JR     NC,READ40    ;CONTINUE 'TIL ONE OUT
F028 80       00390          ADD    A,B          ;NOW ROW*8+COL IN A
F029 4F       00400          LD     C,A          ;TRANSFER TO C
              00410 ;FIND TABLE ENTRY
F02A 3A8038   00420          LD     A,(3880H)    ;SHIFT ROW ADDRESS
F02D 0F       00430          RRCA                ;ALIGN TO BIT 6
F02E 0F       00435          RRCA
F02F 81       00440          ADD    A,C          ;ADD IN INDEX
F030 4F       00450          LD     C,A          ;BACK TO C
F031 0600     00460          LD     B,0          ;NOW IN BC
F033 2142F0   00470          LD     HL,KBTAB     ;ADDRESS OF LOOK UP TABLE
F036 09       00480          ADD    HL,BC        ;POINT TO VALUE
F037 7E       00490          LD     A,(HL)       ;GET VALUE
              00500 ;DEBOUNCE DELAY
F038 21BD0B   00510          LD     HL,2957      ;35 MILLISECOND DELAY
F03B 01FFFF   00520          LD     BC,-1        ;-1 TO BC
F03E 09       00530 READ50   ADD    HL,BC        ;DECREMENT COUNT
F03F 38FD     00540          JR     C,READ50     ;GO IF NOT DONE
F041 C9       00550          RET                 ;RETURN WITH CHAR IN A
              00560 ;KB LOOKUP TABLE HERE.
0008          00570 KBTAB    DEFS   8            ;ROW 0 LC
0008          00580          DEFS   8            ;ROW 1 LC
0008          00590          DEFS   8            ;ROW 2 LC
0008          00600          DEFS   8            ;ROW 3 LC
0008          00610          DEFS   8            ;ROW 4 LC
0008          00620          DEFS   8            ;ROW 5 LC
0008          00630          DEFS   8            ;ROW 6 LC
0008          00640          DEFS   8            ;GAP OF 8
0008          00650          DEFS   8            ;ROW 0 UC
0008          00660          DEFS   8            ;ROW 1 UC
0008          00670          DEFS   8            ;ROW 2 UC
0008          00680          DEFS   8            ;ROW 3 UC
0008          00690          DEFS   8            ;ROW 4 UC
0008          00700          DEFS   8            ;ROW 5 UC
0008          00710          DEFS   8            ;ROW 6 UC
F000          00720          END    CALLKB
00000 TOTAL ERRORS
```

Program Listing 3.

will return a single byte into the A register in the Z-80. In this case the keyboard is treated as a series of memory locations.

Only one row at a time can be addressed. One LD instruction is required for each row.

If no key is pressed, the byte contains all zeros; if one or more keys are pressed, there is a one-bit in every bit position corresponding to the column for the keypress.

As an example, if the #/3 key is pressed, a 00001000 is read into the A register when an LD A,(3810H) instruction is executed.

Sounds simple, doesn't it? To detect a key press, all one must do is to execute eight LDs, one for each row, and look for a zero bit. If none is found, the eight LDs are executed again. If a zero bit is found, a conversion is made to the proper ASCII character for the key has been pushed, a zero is returned in A.

READKB is divided into five parts, scanning, conversion, table lookup, debounce, and the lookup table itself.

Scanning is done by reading in the first seven row values and looking for a zero in any bit. (CPL changes all ones to zeros and all zeros to ones in a "one's complement.") The keyboard address of 3801H, 3802H and so forth, is held in the HL register pair. A new address is obtained by shifting the L register left one bit at a time. For example, 02 shifted left yields 04. The contents of H remains constant at 38H.

When the L register contains 80H (address 3880H), all rows have been scanned and a 0 is returned in A. The M(inus) flag will be set directly after the SLA to 3880H.

The next section converts the row value and row and column.

Learning to play the TRS-80 Model I keyboard is a great deal less demanding than playing Bach or Garner on 88 keys. First of all, we have only 53 keys . . . .

As some of us know from early battles with the keyboard, there are other subtleties involved here. First, there's the conversion to ASCII. The Level II BASIC interpreter uses a somewhat convoluted approach to this conversion by checking ranges, subtracting tricky factors and doing table lookups for special characters. We will take the more straightforward approach of doing a direct table reference, though it requires more memory. That way we can redefine the keys as required from ASCII to any other functions we desire.

Secondly, there's that infamous debounce problem. Level II has n-key rollover, which allows another key to be pressed while the first is being held. We'll use 0-key rollover and eliminate some of the debounce problem.

The process of debouncing simply means that the scan is performed long enough to catch the key press but not rapidly enough to detect an erroneous bounce on either make or break. As soon as a zero is detected, a time delay is performed that provides a reasonable amount of time for the key to be held and released. Typically this would be 20 to 40 milliseconds (20 to 40 thousandths of a second).

Now it appears we have all the elements for reading the keyboard. Let's put them together in a subroutine called, reasonably enough, READKB. The subroutine appears in Program Listing 3.

The subroutine will return an ASCII value (or whatever you put in table KBTAB) for the corresponding key. The subroutine does not wait until a key is pushed before returning. The ASCII value is returned in the A register. If no column value to a value of 0-55 by finding ROW•8 + COL. This index value can then be used to look up a unique value for each row, column address. We have two problems here: How do we multiply ROW•8; and how do we convert the input byte in A to a column value of 0-7?

Finding ROW•8 is done by taking the partial row address in L (01H, 02H, 04H, etc.) and shifting it right until the C(arry) bit is set. For each shift, 8 is added to the A register. The result is ROW•8. The top row, of course, is row 0. We now have a value of 0, 8, 16, 24, 32, 40, or 48 in the A register. Adding the bit position of the column bit will give us a value of 0-55.

First, though, we must convert the column bit position to a value of 0-7. We do this by shifting out the input byte until a one appears to set the carry out of the left hand side of the C register. For each shift a count in the B register is incremented. At the end, the column value is added to ROW•8 in C to give us ROW•8 + COL, a value of 0-55.

We did not scan the SHIFT row in the scanning section for a good reason. We now have a unique (row, column) value in C. Since each key may be upper or lowercase, however, we must convert the 0-55 value to 0-110, representing codes for both upper and lowercase, based on the SHIFT key.

First, we read in the SHIFT key by LD A,(3880H). Complementing this value and shifting right two bit positions gives 0X000000, where X is a zero, if the SHIFT key is not pressed and a one, if the SHIFT key is pressed. This value is then added to the row, column value in C to give a value of 0-119. There are numerous "gaps" in this range representing impossible combinations, but memory is cheap!

The next part of READKB finds the table entry corresponding to the (row, column, SHIFT key) value. The hash value is added to the address of the lookup table KBTAB to point to the unique value for the key. The value is then loaded into the A register.

Next, a debounce delay of 35 milliseconds is performed. The carry flag is reset after ADD HL,BC, if the result is minus. We use this technique to decrement a delay count in HL. The delay count can be changed by the reader for experiments in various debounce delays.

Upper and Lowercase Keys

The lookup table can provide a unique value for upper and lowercase keys, including: SHIFT, up arrow; SHIFT, right arrow; and others. The values for Level II and Disk BASIC compatibility are given in Table 1. These can be added to the source code by DEFBs that replace the 15 DEFS pseudo-ops. (Each DEFS reserves 8 bytes of storage.) Any other values may be defined for the keys. For example, special communications or word processing control characters can replace selected key values.

There is an easier way to obtain a keyboard value. We can utilize the built-in subroutine in Level II ROM at location 03E3H. This is the keyboard driver routine. A word of warning, however. This driver is replaced by another driver in TRSDOS or other programs to eliminate debounce problems.

To use any of the drivers, find which one is in use by examining the keyboard DCB (data control block). The address of the keyboard driver in use is at locations 4016H and 4017H (16406 and 16407) in reverse Z-80 address format (least significant byte of address followed by most significant). Use this address and perform a CALL to the driver. On return, the A register will contain a zero if no key has been pushed, or the ASCII-like value if a key has been pushed. If a key has been pushed, n-key rollover and debounce are also performed.

Next month we'll examine some techniques of assembly language high speed graphics. In the meantime, try as much assembly language coding as possible, and remember, Radio Shack's computer stores are now stocking tranquilizers for assembly language programmers (part #Z80-OHNO). ■

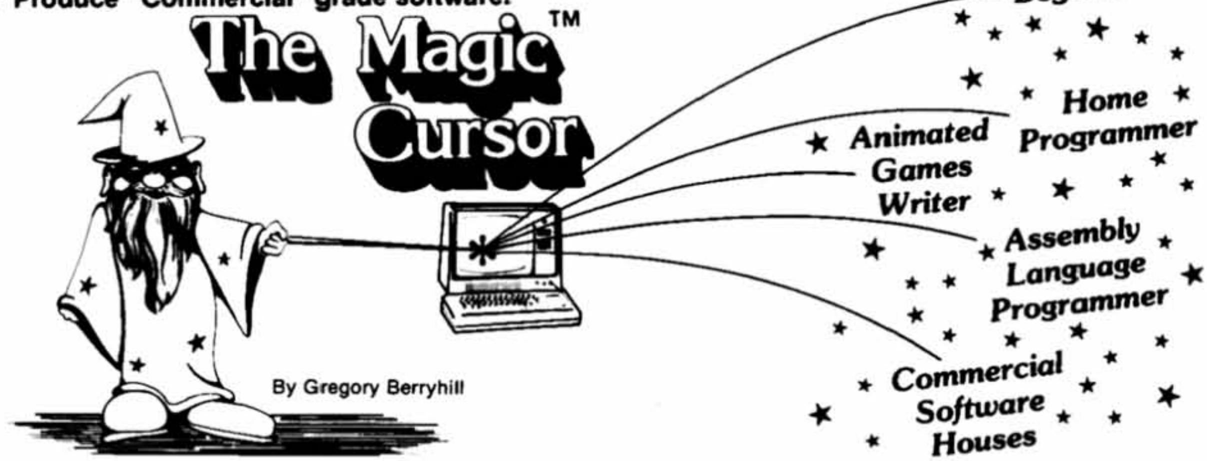
```
All values hexadecimal

40,41,42,43,44,45,46,47     Row 0 LC
48,49,4A,4B,4C,4D,4E,4F     Row 1 LC
50,51,52,53,54,55,56,57     Row 2 LC
58,59,5A,0,0,0,0,0          Row 3 LC
30,31,32,33,34,35,36,37     Row 4 LC
38,39,3A,3B,2C,2D,2E,2F     Row 5 LC
0D,2F,1,5B,5C,5D,5E,20      Row 6 LC
0,0,0,0,0,0,0,0             Gap of 8
20,61,62,63,64,65,66,67     Row 0 UC
68,69,6A,6B,6C,6D,6E,6F     Row 1 UC
70,71,72,73,74,75,76,77     Row 2 UC
78,79,7A,0,0,0,0,0          Row 3 UC
20,21,22,23,24,25,26,27     Row 4 UC
28,29,2A,2B,3C,3D,3E,3F     Row 5 UC
0D,2F,1,5B,5C,5D,5E,20      Row 6 UC
```

Table 1

TRS-80* OWNERS:

- Let the computer write your "Basic" programs for you!
- Draw Pictures, animated figures, data forms!
- Create a library of displays!
- Produce "Commercial" grade software!



The **Magic Cursor** is a Revolutionary Family of Products which provides a dramatic new method of reproducing drawings and displays that you create on your screen. It makes both simple displays and complex interactive data input forms. It stores a "BASIC PROGRAM" on disk (or tape) ready for you to execute alone or as a subroutine. It produces screens in both standard and wide screen.

It is available for any level 2, 16K or larger system with tape or disk. An optional version is now available which creates an assembly language program.

Be sure to pick out the system that fits your present needs and order it today. You may upgrade your original copy by paying the difference and a moderate service charge.

MAGIC CURSOR PROGRAMS

THE BABY CURSOR allows you to easily create screens (including graphics) on your video. A powerful command then generates the BASIC instructions to recreate the screen. For the first time, a program for automatic generation of video display forms. (16K Tape or 16K Disk) **$24.95**

THE MAGIC CURSOR I additionally makes sophisticated Data Entry and Display easy. With Magic Cursor I you define the Data Entry or Display fields directly on your screen. The definition commands generate the BASIC instructions to implement the Data Entry and Display. The Magic Cursor I has commands which move, center and duplicate blocks of graphical or alpha/numeric displays. You can even justify text. (16K Tape Only) **$79.95**

THE MAGIC CURSOR II adds the power to write animated games easily in BASIC. The Magic Cursor II allows you to reload previous screens either from memory or from Disk. You can then modify them and store either the modified screen or only the changes. (32K Disk Only) **$99.95**

THE MAGIC CURSOR III will be available soon for the new Model II Computer (32K One or more Disks) **$149.95**

THE MAGIC CURSOR IV provides the features of Magic Cursor II but stores an assembly language program. (32K Disk Only) **$99.95**

Call 713/474-2428 or order by mail. Master Charge, Visa, Certified Check or Money Order accepted. Personal Checks require 14 days to clear. C.O.D. or collect calls not accepted. Software Guaranteed for replacement only. Prices subject to change without notice. Programs supplied on cassette tape. For disk versions, the cassette supplied will automatically create a disk file.

*Trademark of Radio Shack, a Tandy Co.

OTHER PROGRAMS WRITTEN BY CCC

GENERAL LEDGER, Accounts Payable, Accounts Receivable, etc. - We are an official Osborne/McGraw Hill conversion co.

RESTAURANT MANAGEMENT - Cost analysis, inventory control, recipe design, labor scheduling, etc. for both fast food and traditional restaurants.

INSURANCE RATING - Instant pricing for policy premiums based on age, marital status, etc. for both Texas and National.

SMALL BUSINESS MANAGEMENT - From our library of custom design for DIAMOND & JEWELRY MERCHANTS, DONUT SHOPS, SAILBOAT SAIL SHOPS, CONSTRUCTION FIRMS, ENGINEERING FIRMS, MANUFACTURING FIRMS - in the areas of cost analysis, product design, inventory control, product pricing sheets, purchase order, invoicing, etc.

PROGRAMMER UTILITIES for supporting assembly language programming and disk modification.

MAGIC MEMORY - personal inventory programs for stamp collection, coin collectors, personal libraries, etc.

WRITE FOR OUR COMPLETE SOFTWARE CATALOG!!

Send Check or Money Order payable to:

CUSTOM COMPUTER CENTER, INC.

ATTN: Jim Martens
P.O. BOX 58042
HOUSTON, TEXAS 77058
or call: 713/474-2428

CUSTOM COMPUTER CENTER, INC. P.O. BOX 58042 / DEPT. 8-E / HOUSTON, TX 77058

SEND INFORMATION ONLY.

QUAN.	DESCRIPTION	SYSTEM (Size/Tape or Disk)	PRICE

Handling Charge: $1.50 Texas Res. add 6% Sales Tax.
TAX & HANDLING
TOTAL
MASTER CHARGE / VISA
CARD # ______ EXPIRES ______
NAME
ADDRESS
CITY
STATE ZIP

80 APPLICATIONS

by Dennis Kitsz

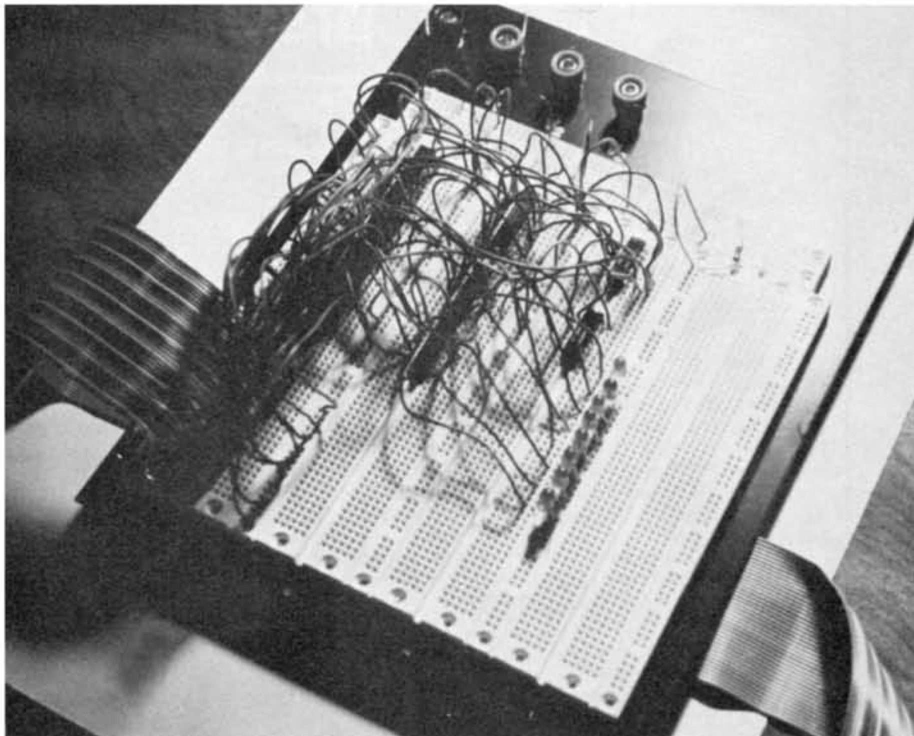


Photo 1. *Prototype board version of I/O latch. LEDs to the right are part of my testing routine.*

Photo 2. *Before and after versions of the I/O latch project.*

Photos by Dennis Kitsz

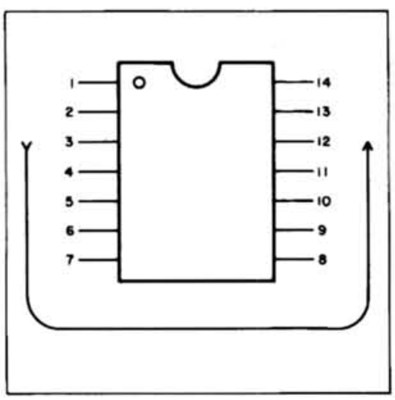
The most difficult part is deciding to begin. Really. And, I must reveal, the impetus for me was strictly economic.

In January's column, I spoke in less than rhapsodic terms of my music synthesizer. It has in fact served me well, for if it hadn't been for the shock of a $100 repair bill, electronics would have remained for me an immutably undecipherable code. But, there came a next time, and another synthesizer failure, so, with frugality aroused, I vivisected my music machine—and stopped face-to-chip with integrated circuits. Eventually, I learned that although electronics is made up of equal parts skill and knowledge, the greatest requisite is a fool's courage.

So join me.

Radio Shack Boards

First, we recall the arrangement of pin numbers of a standard dual-inline package (DIP):



The circuits suggested last month have from 14 to 20 pins, but the numbering always begins at the upper left near a notch or dot (or both), proceeding down one side and up the next.

There are two ways to construct this circuit. The first is to obtain a prototyping board, available by mail or from Radio Shack (Cat. No. 276-174). You'll need two if you use this board. The other method is to assemble the final, soldered version. Again, there's a Radio Shack board (276-170).

Being impatient, I invariably use this approach when building published and debugged circuits.

Notice in Photo 2 that I have soldered the IC's to the foil side of the board, an unusual

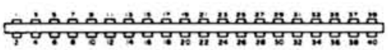
Continue to next page

practice. With many wires on the breadboard, though, this method is easier for the eyes to follow, and has no effect on the operation of the device.

Fit each part onto the board, soldering it carefully in place. Insert short lengths of wire connecting the IC power pins to the supply voltage (the vertical bus to the right is convenient) and ground (the left bus). Notice that the power pin arrangement is similar for all circuits but the 74LS75. Also connect any other pins to ground as indicated in the schematic. Cut and gently strip the ends of wire lengths, and run them as shown in the circuit diagram.

Work slowly and always keep the board oriented, so you know the order of the IC pins. Particularly if this is your first experience with digital hardware, work for short periods and double-check similar groups of wires. Note that some input, for example, is arranged in groups (74LS30), while others alternate (81LS95). Be sure also to connect together the three points marked "A."

The most tedious job is to separate, strip and solder the wires from the 40-contact edge connector. Use a razor blade or Exact-o knife to separate the leads for about 1½ inches; strip a small amount of insulation from each. Carefully examine the TRS-80 edge card diagram below:



The connector is a mirror image of this. Identify which lead matches which connection point, as the leftmost wire may be #2! Cut small bits of mailing labels or white tape, number 1 to 40, and wrap one around the end of each wire. Refer again to the schematic diagram, and solder each wire in place. Solder the remaining leads to separate, unused terminals on your board, or snip off the bare ends.

On the schematic, the outputs of Z2 and the inputs of Z7 remain unconnected until we determine which output port we wish to use. Since last month we used the example 178 (binary 1011 0010), temporarily (but securely) run connectons between the following pins:

Z2	Z7
2	1
5	2
6	3
8	4
11	5
13	6
16	11
17	12

Mount the boards on standoffs or wooden

Continue to next page

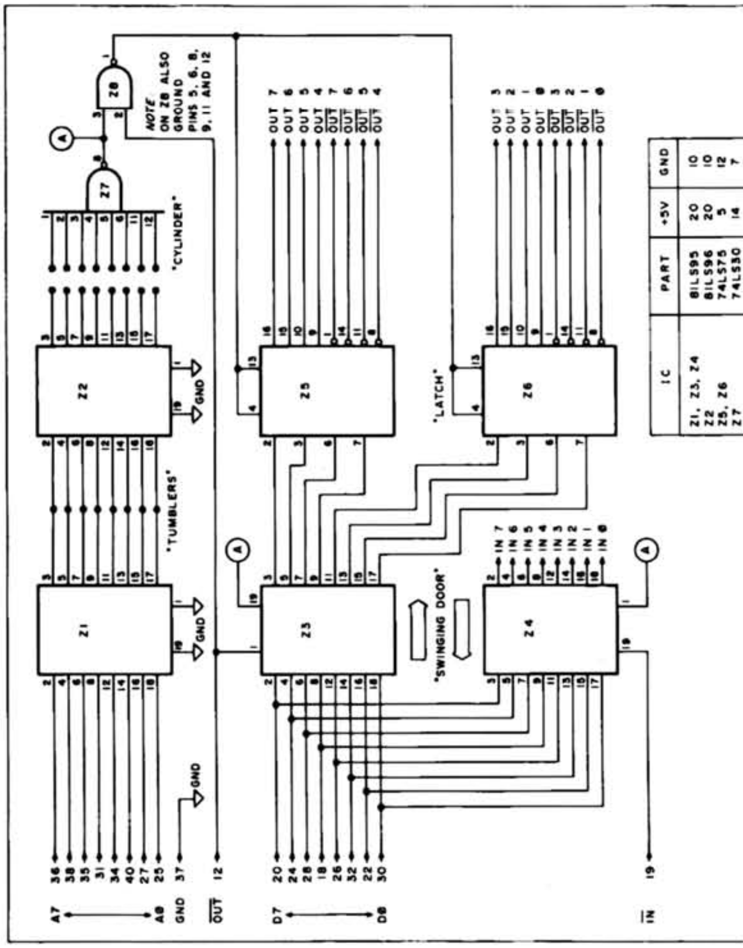


Figure 1. Complete Circuit Diagram

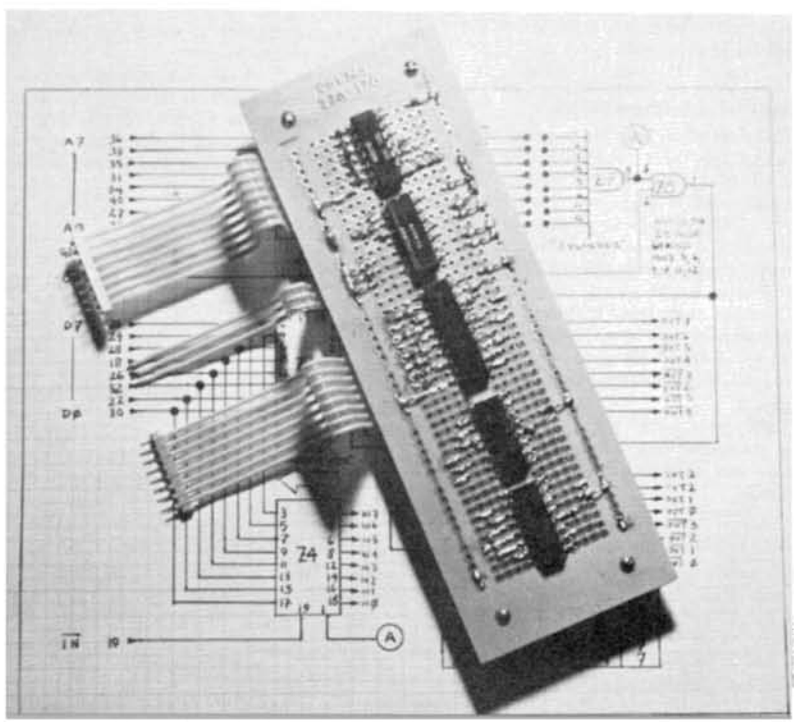


Photo 3. One section of soldered breadboard version, containing (from top) Z7, Z8, Z3, Z5, Z6. ICs are placed on the foil side for ease in routing wires.

CAPTAIN 80

by Bob Liddil

The real culprit in the demise of the small software vendor is none other than the end user.

Here's the Captain, kicking back after a hard day's work in the old salt mine. My new disguise, that of Joe Independent Software Producer lies in disarray on the couch.

I thought it was easy to run a software company. It seemed simple enough. Acquire the rights to a couple of good programs, get a few stores to sell and promote your software and POOF! Enter The Captain Eighty Software Exchange.

Wrong Number

Boy, did I get a wrong number. The most common reason (Those who've been there will tell you.) for the failure of software companies is not poor programs, but poor judgement. More than one distributor that I have dealt with over the last 90 days has teetered on the brink of extinction because of the thoughtlessness of computer stores who do not pay their bills on time. The software may still be sitting on the shelf or not. Thirty days pass and, still, no check arrives.

Meanwhile, the bills at the software company pile up, setting up yet another chain reaction in which the poor author does not see a check for months at a time.

The real culprit in the demise of the small software vendor is none other than the end user. No, not the guy who shelled out $14.95 for a copy of Scott Adams Adventure, but the 15 others who accepted bootleg copies, then proceeded to enjoy the program, as if they'd paid the price to play!

Who Me? A Software Pirate?

Who among us hasn't at one time accepted a backup copy of a favorite program from a friend? It is second nature to want to slide by on as little dollar investment as possible. But multiply 15 bottleg copies by 1,000 purchasers of NEWDOS at a conservative $50 price and suddenly the loss is a staggering: $750,000! That is the loss to one vendor and on one program.

Multiply that figure by the thousands and thousands of backed-up bootlegged programs and you'll need a computer and a good math program, if you can find one, to calculate the damage.

Sadly, most people see no wrong in duplicating programs, modifying them to suit their taste and distributing copies among their friends.

Case in Point

I recently helped a friend market a program. It was an adventure, written in BASIC. Luckily, for me and my author, there was a chain of computer stores in Boston with a buyer that liked the cut of the program.

A couple of weeks went by, and I was anxious to learn how the program had been received. I discovered that Danny Debug, a local computerist, had bought the adventure at his friendly neighborhood computer store. He did not particularly like the opening display (title page) or the tape and disk save routines (you know, "SAVE GAME"). So he deleted them from the program.

Now this, alone was no big deal. Danny Debug has the right to rip apart any program that he buys. But the story didn't end there. Once the program was altered to suit its new owner, it was returned to the computer store where the virtues of the new improved copy were extolled.

No, Mr. Debug didn't offer to SELL the new version. He GAVE IT AWAY. First, he gave a copy to the manager of the computer store, then, who knows? Without a doubt the program, in its new format would travel to a users' group where copies would be enthusiastically scarfed up by software hungry people who wouldn't think of stealing.

Out of Games

Noted computer author Lance Miklus, whose Mean Checkers Machine and Star Trek programs have given me many countless hours of enjoyment, said that he was forced out of the game market by pirates. Mr. and Mrs. Joe Average who duplicate tapes for their friends and accept the same have robbed the marketplace of one of its most skilled and imaginative game writers.

I know a guy, named Paul, who staunchly refuses to accept anything but a labeled, original copy of software. I have seen him turn down some fine programs because of his principals. I have even joined in the ribbing he took each time he stuck up for what he believes in.

Suddenly, it isn't funny anymore. The shoe is on the other foot, now that I've tried to help a friend. Paul has had the right idea all along. It'll just take the rest of us a little while to catch up.

I asked the question in a recent column, "Where are all the high quality game and educational programs?" Now I know. ■

80 APPLICATIONS

From page 23

strips, and run wires to your 5-volt power supply. These circuits demand ±5 percent accuracy, so don't substitute a spare 6-volt battery eliminator! Without attaching the cable to the TRS-80, apply power to the board. If you have a meter, check for a current draw of about 130 mA; IC temperature should rise just above room level.

Connect to 80

Remove power from both the computer and your input-output (I/O) board, orient the cable carefully, and connect it to the TRS-80. Restore power to the computer, quickly run through MEMORY SIZE? to READY, and try a few commands.

If at any point the TRS-80 does not perform in the usual manner, turn it off!

Remove the cable and recheck your work, particularly the cable connections themselves. Do it with a friend so you don't overlook familiar errors.

If all is well, apply power to the computer and to the I/O latch, and again assure the proper operation of the computer. At this stage, be especially quick and careful.

The device is now complete and ready for testing. Set up these programs:

```
10 PRINT INP(178);:GOTO 10
20 OUT 178,255 : OUT 178,0 : GOTO 20
30 FOR X = 0 TO 255 : out 178,X : NEXT : GOTO 30
```

Run 10. The screen should repeatedly display 255. Now ground each input of Z4 in turns (pins 2, 4, 6, 8, 12, 14, 16, 18). The values displayed should change. Record them, convert them to hexadecimal and binary, and observe their relationship to the grounded inputs.

Run 20. Briefly connect an earphone, in series with a 100 to 500 ohm resistor, between ground and each output of Z5 and Z6. You should hear a buzz of identical pitch (the level may vary) from each output.

Run 30. Reconnect the earphone, but notice that the pitches vary from a buzz to a series of very slow clicks.

When your device passes all these tests, it is ready for use. If you have the background, put the input-output latch right to work. Otherwise, next month: how it works, why it works, and what to do with it.

• • • • • •

Problems? Send a complete descriptions of symptoms and an SASE to me at Roxbury, Vermont 05669. Calls accepted (not collect!), if your TRS-80 and telephone share the same desk. ■

TBS-80 SYSTEM UTILITIES.

If you thought your TRS-80™ microcomputer was just a toy, think again. Enhance your system with these powerful **TBS-80 system utilities.**

TERMINAL CONTROL by F. Barry Mulligan is a machine language utility that enables you to use all the potentials of RS-232 tele-communications without hassle. Requiring 16K or more, it can interface to any Level II BASIC or assembly language program, or may be used as a stand-alone system to send and receive entire programs or data. The beauty of this program is that it turns your computer into a truly smart terminal. All RS-232 features can be set from the keyboard and the current values can be displayed or changed at any time. Basic programs can be sent in Level II compressed format for high-speed exchange. Whether you want to send or receive data from a BASIC program, or talk with the computer networks and bulletin boards or with any other terminal or computer or try any of the possibilities that computer communications has opened up, **TERMINAL CONTROL** is your answer. Only briefly described here, this remarkable program sells for only $19.80 on tape and $29.80 on disk.

SYSTEM DOCTOR does a thorough diagnostic check of your entire computer system. It lets you know if something is wrong before you spend time programming or entering data. The program checks the ROM to ensure that every bit is functional and checks the RAM six different ways. The disk drives are tested in a variety of ways to ensure reliability. The cassette recorder is also tested for speed, volume and distortion with the help of a calibration tape provided with the program. The video memory and display are also checked as well as the line printer. **SYSTEM DOCTOR** also does a 12-hour check of the entire system and records the results on tape, disk or the screen. As a bonus, this program also includes the **DISK DRIVE HEAD CLEANER.** The card insert that cleans the head can be obtained free by mailing in the coupon provided. For $28.50, **SYSTEM DOCTOR** is the first complete diagnostic program for the TRS-80. A disk version is available for $38.50.

LINE PRINTER by Dosse Segbeaya is a machine language program that accelerates printing on Centronics printers by making it a background task. Requiring 32K and a disk drive, this program enables the user to set aside up to 16K of memory as buffer which when filled is sent to the line printer while your Basic program continues to run. Any Basic program that uses LPRINT's will run significantly faster with this program. Also included is the ability to set the number of characters per line, the number of lines per page, the spaces between lines, and the left, top and bottom mar-ins. Page numbers can be placed anywhere on the first line starting at any given number. Printouts of anything that is on the screen can also be made by hitting shift/break. If you do programming and you use multistatement lines, **LINE PRINTER** enables you to LLIST your program with single statement lines. This rather amazing program is resident in high memory as it interfaces with almost any Basic program. It sells on disk for $24.50.

BASIC TOOLKIT by F. Barry Mulligan is a basic programmer's dream come true. Requiring 16K or more, this program has the following features. Variables Map-Gives an alphabetical listing of each variable used, a list of the lines the variables appear on, and shows the number of times the variable appears on the line. Goto X Ref-Lists in numerical sequence the destination of each GOTO and GOSUB statement and the line number that it appears on. Recall-Allows you to recall a program after you have hit reset, accidentally typed NEW or have booted back to DOS. Merge-Enables you to merge tape or disk programs. Test Memory-Does a thorough check of memory to be sure every location is operable. Search Memory-Search for every occurrence of a two-byte combination and list the location where it occurs. **BASIC TOOLKIT** is resident in memory while programming and is accessed by hitting shift/break. A must for basic programmers, this utility sells for $19.80 on tape, $29.80 on disk.

TBS has other incredible software for Tandy's microcomputer. Intent on making it a powerful tool, we have large scale **business accounting systems, general accounting systems, data processing systems** and the **Library 100.** We have the only **DISK HEAD CLEANER** (for APPLE too!) and **GRAN MASTER DISKETTES,** the best on the market.

TBS is **YOUR COMPANY,** and we build systems, not just software. The above products are available now, nationwide. Visit your local Computer Dealer or Associate Radio Shack Store and demand the best, demand TBS. For more information contact us through the numbers below.

™ TRS-80 is a trademark of the Tandy Corporation.

TBS™ **THE BOTTOM SHELF, INC.**
(404)939-6031 • P.O. Box 49014 • Atlanta, GA 30359

r Service – see page 162

80 INPUT

In your article "Screen Editor" the author uses expressions like . . . "all we have to do is stick a zero at . . . . " Well, I can tell you where to stick it!

Fully Functional Fix

I purchased a Level II, 16K, TRS-80 last January and a 32K expansion interface last July; for the most part I am a satisfied TRS-80 owner. However, a problem arose within the memory of the expansion interface. Using the default option by responding with ENTER to the MEMORY SIZE? request, not all 48K of RAM was available in the machine; by not using the default option all the memory could be accessed.

The Radio Shack Computer Center in Buffalo charged me $55 to fix this problem. They installed a new interface board that eliminated the buffered cable. Regretfully, when I tested the computer at home the same problem remained. I returned the TRS-80 to the service center where it was discovered that a memory chip, "though fully functional," was not responding to the default option. I was informed that the problem was solved by merely rearranging the memory chips.

One of the principle reasons I purchased a TRS-80 was anticipated high quality follow-up service. I am not pleased that (1) Radio Shack "fixes" things that are fully functional and charges me for this service, and (2) so much of my time has been wasted on this problem.

Dr. Grant Ian Thrall
Buffalo, NY

Findisk-II

In the January issue you reviewed the Mumford disk indexing program, which I have used and found to be a fine program. However, your readers should also know about Documan Software's Findisk-II which I now use. It is the same price, does all the work of the Mumford program and has some very useful added features.

Most useful of these is an Update routine that revises the Master Index by eliminating killed files and adding any new files—automatically—just be reading the revised disk.

Another feature that I find helpful in Findisk-II is the ability to add a description to each file name so that those terse file names of yesteryear can be deciphered. And we all have several versions of most programs, so this feature permits their specific differences or sizes to be recorded.

Since NEWDOS and VTOS allow diskettes to be numbered during formatting, another good feature of Findisk-II is its ability to read those numbers automatically if they are used.

No matter which program you use, I am convinced anyone with disks needs a program of this type to keep his sanity.

George Kelly
Tempe, AZ

Single/Double Density

I just received my March issue of *80 Microcomputing* and am reading it from cover to cover. I am glad to see your company publish a magazine dedicated to the TRS-80. There are so many things about the TRS-80 and associated hardware that the companies won't tell you about. I would like to see your magazine talk about disk drives, single vs. double density, 35 vs. 40 vs. 77 track.

When I bought my 35/40 track drives I was led to believe that they were 40 track single density disk drives and that the expansion interface could not process double density disks. Well, to my surprise and without the help of the sellers, I found out my disk drives were capable of handling double density.

In addition I see one company now coming out with a black box that converts the expansion interface to double density. This could increase my drives from 102K of available space to 204K. In addition, if I had all four drives, I could go all the way to 816K space.

Now I am looking closely at the worm gear drive which moves the recording heads on the drive to see if they could be changed to give me 77 tracks vs. 40 tracks. I tried my 77-track patch pack to see if the DOS, computer and interface could handle 77 tracks. Well the DOS and computer clicked off all 77 tracks without any problem. But of course the disk drive stopped at 40 tracks, even though it continued to try and advance beyond the stop.

My observations on this indicate that by just changing the worm gear and possibly the record head I could increase my disk drives capacity from 35/40 tracks to 77 tracks. Think of this, I could possibly increase my disk drives from 102K to 1,576,960. That's 1.5 Meg disk storage for four drives. I would like to know what your readers think of this idea.

Donald H. Bennett, Jr.
Woodbridge, VA

Print the Directory

The lack of a printout routine in TRSDOS for printing the disk directory has been a problem for me. When I had accumulated several disks with several versions of a number of programs, each having a unique file name, I found it hard to keep track of the location of a file and correct spelling of a file name.

Recently Radio Shack published a BASIC language routine that routes the information normally destined for the screen to the printer. I found that running the statements under BASIC and then reverting to DOS by CMD"S" allowed me to print the directory listing under the DIR command.

I incorporated this into a program which, after printing, I could return to BASIC using the BASIC* command and run the program using a GOTO 100 command. This program saves me the trouble of remembering the address and values required to set and reset the VIDEO/PRINT operation.

```
10 ' PROGRAM TO PRINT VIDEO INFO
20 '
30 POKE 16414,141:POKE 16415,5
40 CMD"S"
50 END
60 '
100 ' PROGRAM TO RESET VIDEO
110 '
120 POKE16141,88:POKE 16415,4
130 END
```

D. E. Fitchhorn
Albuquerque, NM

Beat the Cheat

In response to Mike Tollerton's dilemma (Microchess cheating), I suggest he continue the game:

MOVE	WHITE	BLACK'S RESPONSE
13	F8-F7	G8-E7
14	F7-E7	C7-B8
15	E7-D6	CHECKMATE!!

The above not only provides a happy solution to Mike's problem, but also proves that "Crime Does Not Pay".

I would like to add my plea to those of Mark S. Lucas and Christopher M. G. Buttery for machine and assembly language discussions in plain simple English with step-by-step instructions and a lot of "for examples".

In your article, "Screen Editor", the author uses expressions like: "This turns out to be quite simple on the TRS-80"; "This is easy to implement"; "All we have to do is stick a zero at . . . " Well, I can tell you where to stick it!

Articles like this are frustrating. The program listing looks impressive, but how do I get it into my machine and how do I use it?

Another frustrating example: "EDTASM on Disk" in your first issue. The heading above the title teasingly states, "A step-by-step procedure . . .," etc. Fine! Until you get to Step 6. Stone wall! How do you type Program A into that little square inch of space in the lower left of your screen?

You must realize that when Radio Shack came on the scene a vast new breed of amateur computerists was spawned. People like myself who, with no previous knowledge, struggled through and mastered Level I BASIC (thanks to David Lien) and with that foothold went on to grapple Level II.

Now we have this big chasm, this no-man's-land, between BASIC and seemingly unreachable machine and assembly language. Please, somebody, show us the way.

Robert L. Boelke
Alta Loma, CA

Warranty Clarified?

I found your first copy of *80 Microcomputing*, like *Kilobaud*, excellent. Congratulations on this new publication.

After reading articles and columns dealing with TRS-80 modifications, readers may be tempted to pop the covers off of their own equipment. Before they break the seals they should be aware of Tandy's little publicized service policy. The Tandy policy is to refuse to service and/or impose a substantially higher repair charge. I, for one, will continue fighting this policy. (See letter below.) I feel that if all TRS-80 owners and your publication support my protest of this policy, Tandy will have to change it.

I do know that the service policy from most major computer manufacturers allows owners to touch their hardware beyond the keyboard, especially after the warranty period. With Tandy's introduction of the TRS-80 Model II, they now are entering the "major leagues" of computer manufacturers. They are comparing Model II with DEC, IBM and HP in their latest catalog. As far as service policies are concerned, however, the comparison stops here.

Ernest Kirschner
Durham, CT

Dear Mr. Kirschner:

Mr. Kornfeld has referred his copy of your letter of September 14, 1979 addressed to the Federal Trade Commission to me for response. As I view your letter, you make two related complaints—the first is directed at our practice of making a higher charge for repairs where a customer has modified or tampered with computer equipment and the second is our failure to give notice of this policy.

We feel our policy of refusing service and/or imposing a higher repair charge on computer equipment that has been modified or tampered with is not only legal but is totally justified.

Servicing and repairing computers is highly technical and in those instances where repairs are attempted by an unskilled technician or modifications are made to a system, subsequent repair work and servicing becomes extremely difficult. Not only must a technician find and fix the real problem, but he must also test the entire unit to make certain that in attempting to fix the unit the person tampering with it did not create additional problems. When modifications are involved, the technician has the additional problem of trying to ascertain the nature of the modification and whether it is the problem.

The truth of the matter is that we simply do not want the responsibility of guaranteeing our work on equipment that has been modified or tampered with by unskilled technicians.

I feel your second complaint has merit. While I know of no law, rule or regulation requiring a seller to advise consumers that if they tamper with or modify a product the repair charges at the seller's facility for such product will be greater than those charged for other products, I do feel this information should be made known to consumers.

We are considering this problem and in all likelihood we will insert this information in our owner's manual or on the warranty card the next time it is reprinted. In this manner, we can hopefully avoid future instances of consumer dissatisfaction.

Louis G. Neumann
Staff Attorney
Tandy Corporation

DOS to BASIC

Here's how to AUTO LOAD a BASIC program from Radio Shack's DOS. Enter TRSDOS 'DEBUG' utility. Type D6B00 (enter); to display memory locations to screen. Type M6B00 (space); and enter the following machine code exactly as shown:

```
6B00: 21 0F 6B 22 16 40 21 2C 6B 22 2A 6B C3 2D
      40 E5
6B10: 2A 2A 6B 7E FE 0A CA 1F 6B 23 22 2A 6B E1
      C9 21
6B20: E3 03 22 16 40 3E 0D C3 1D 6B 20 20 1F 42
      41 53
6B30: 49 43 0D 0D 0D 52 55 4E 22 4D 45 4E 55 45
      22 0A
```

Remember to use the space bar to increment to the next location. After the 0A at location 6B3F is typed, press (enter) and then type G402D (enter) to return to DOS READY. Type TAPEDISK (enter) and save the following program:

```
? F MENUE/CMD:0 6B00 6B3F 6B00
? E (enter) to return to DOS READY.
```

You now have a CMD file program called MENUE/CMD which when executed via DOS's 'AUTO' command will load 'BASIC', set number of files to 3 (default value), set memory size to maximum (default value), and RUN a BASIC program called "MENUE".

The BASIC program "MENUE" may be any program previously saved using "MENUE" as its filespec or it may be an actual diskette --MENUE-- with program choices listed. If you have a dedicated diskette and wish to auto execute some other filespec, then modify memory locations 6B39 to 6B3D to contain your own five character filespec. For example HEX 4D = M, 45 = E, 4E = N, 55 = U, and 45 = E. INVEN would be 49 4E 56 45 4E.

Gary Alcorn
Tempe, AZ

Avert Disaster

I work full-time as a programmer/analyst at a large firm. I have never written to a magazine before, but I noticed a program in your magazine which could cause financial disaster, even if used as written. This is the program in the article "Passing the Plate", which appeared in the February issue.

I have two suggestions for anyone who is thinking of using this program:

1. Instead of erasing last week's data, keep a separate tape for each week, and save the data twice on each tape. In the unlikely event that both sets of last week's data are bad, you can at least pick up from the week before.

2. Unless you have a printer, and have adjusted this program to print all detail information), don't throw away the books! Keep them up as usual, just in case . . .

Finally, if you are going to computerize anything beyond the household budget, it is a good idea to show your programs to a professional accountant or business consultant.

Jim Slack
White Cloud, MI

Mis'Guide'd

First of all, I very much appreciate the positive review of my publication, "*Guide to TRS-80 Information*," under Rival Publications II, *80 Microcomputing*, February 1980, p. 38.

At the same time, however, you can understand my distress that this review quoted the wrong price for the "Guide." With a low price booklet such as this I'm sure you can see the importance of avoiding extra processing steps, time and costs.

The "Guide" in fact sells for:

1 copy $2.80 plus .50 postage and handling
($1.50 Airmail, if overseas)
2 to 9 copies $2.75 each
10 to 50 copies $1.90 each
(plus 20% postage, if overseas)

By the way, no one has ever returned a "Guide" for a refund. Thank you for your cooperation.

F. E. Huebner
Oak Park MI

New Column?

I am writing to ask that you consider a new column for your new *80 Microcomputing* magazine.

Continue to next page

While your magazine quite rightly caters to the many individuals who own a Model I, a growing number of other persons like myself are investing in Radio Shack's new TRS-80 Model II, and need some way to regularly express their findings.

I feel that *80 Microcomputing* has a responsibility to these new TRS-80 owners, who are most likely Model I owners too, to provide a column in which software can be evaluated and programming routines can be shared.

John I. Spady
Seattle, WA

We are currently considering a regular Model II column.—Eds.

Paid for Itself

I just cannot let another day go by without writing to you about your new microcomputing magazine. I can say, without a doubt, that your magazine is the best around for TRS-80 users.

Your first issue answered two questions I have had for a year. Your second issue really blew my mind. It cleared up three problems I've been having with my disk system.

In my mind, your magazine has already paid for itself with two issues, and I have ten more to come!

David R. Laplante
Worcester, MA

80 Skywatch

Fairborn University is a very modest affair devoted to photoelectric observations of variable stars, lunar occultations, etc. A TRS-80 is interfaced to the photometer for realtime data collection and for data analysis such as calculating extinctions, local sidereal time, etc.

Perhaps other TRS-80 owners also use their TRS-80 in astronomically related activities? If so, I would very much like to hear from you.

Russ Genet
Fairborn University
1247 Folk Road
Fairborn, OH

80 DEBUg

KWICer Index

I have discovered that there are two errors in the Disk BASIC listing of my KWIC Index program published in the March 1980 issue of *80 Microcomputing*. Line 1110 should read:

```
1110 IF W(I)=TW THEN 1190
```

Line 1160 should read:

```
1160 TC=TC+" "+TW
```

I am sorry to have inconvenienced you and your readers.

I have also had several questions about the bracket in the program listing for Investment Analysis. The printer on which I made the listing does not print the up arrow (↑) but prints the bracket ([) instead. This is fairly common for many printers and shows up in many program listings.

Leslie E. Sparks
Durham, NC

Galactic Update

There may have been some confusion about my article in the Feb. issue of *80 Microcomputing* concerning the speed of the Galactic Software, Inc. Mail/File program. As stated in the article the actual printing of labels by this program is not as fast as some of the other programs on the market.

However, if you include the time it takes to sort the names before you can print your labels, the Galactic Software Inc. Mail/File is much faster than any of the other programs. There is also a patch to the program available from Galactic Software that will now allow you to print up to six labels across.

This effectively increases the throughput greatly. This, when coupled to the fact that the program is always ready to print both an alphabetical or zip code list, makes it the fastest mailing label program for the TRS-80 Model I Computer.

Galactic Software has also released a version of the Mail/File program for the TRS-80 Model II. Like the Model I version this program is also always ready to print labels in both alphabetical and zip code order. However, the Model II version will handle 2,500 names and addresses.

Reese C. Fowler
Peterborough, NH

Re-video

In "Reverse Video" (April), my mind transformed a type 75452 integrated circuit into Z52.

Please make these corrections: After the subheading "Reverse Video" at the bottom of page 54, change all references from Z52 to *Z41*.

The plated through hole in the wiring layout also goes Z41, as does the arrow labeled "Video Mixing" in the schematic. If you followed the trace, you should have connected the wire correctly in spite of the labeling error.

My apologies to readers who may have been confused by this, and let me remind readers that they are welcome to call me at home with questions or clarifications on any of my articles.

Dennis Kitsz
Roxbury, VT

DIR Overwrite

I ran into a problem following Mssrs. Butlers' instructions in their EDTASM on Disk article. Step 19 says, "Type Tapedisk," then gives instructions to respond to the "?" prompt.

Upon loading Tapedisk, however, my machine crashed into Debug, which was still in memory even though the Debug (off) command had been executed. The TRSDOS manual recommends typing DIR after Debug (off) to avoid this problem. Unfortunately the EDTASM has been moved by Program A to the overlay area of memory and is therefore overwritten by the DIR command.

Assuming the user's machine has 32K or more the fix is to move the EDTASM to memory starting at 8000H and retrieve it from there.

Only four changes are required:

```
In Program A line 00130 becomes 6583 110080 00130
    LD DE,8000H ;Dest. Address
In Program B line 00120 becomes 7FF0 210080 00120
    LD HL,8000H ;Source Address
Add step 18a. "Type DIR (enter)"
In step 20 the instructions to Tapedisk should be
   F EDTASM/CMD:0 7FF0 9A40 7FF0 (enter)
```

Good luck with the new magazine, the first issue was excellent.

Wendell Kapustiak
Downers Grove, IL

THERE IS A DIFFERENCE IN TRS-80 DISK DRIVES CAPACITY

LOBO DRIVES' new family of disk memory products provides you with a choice of memory capacities you need to effectively execute the complex business software you've developed for your TRS-80*. LOBO DRIVES' selection of readily available, software compatible drives permits you to expand your inventory, payroll, customer list, and accounts receivable files as your business grows.

And LOBO DRIVES brings you more . . . a new plug-in expansion interface that provides an easy way to add hardware enhancements, communications capability, and programmable features . . . and it comes with the LOBO DRIVES famous 1 year, 100% parts/labor warranty.

Call or write for the complete LOBO DRIVES story. Find out just how competitively priced a family of high capacity drives can be . . .

LOBO drives INTERNATIONAL ✓15

935 Camino Del Sur Goleta, California 93017. (805) 685-4546

"CAN YOU REALLY AFFORD TO PAY LESS?"

Quantity discounts available – Dealer inquiries invited

TR-2

Yes, I want to know more about LOBO Drives and what they can do for my TRS-80. Send me information on:

- ☐ 5 1/4-in. Floppy drive
- ☐ 8-in. Winchester hard disk, 10 Mbyte drive
- ☐ 8-in. Floppy drive Single sided Double sided
- ☐ Double density expansion interface

Name ____

Company ____

Address ____

City ____ State ____ Zip ____

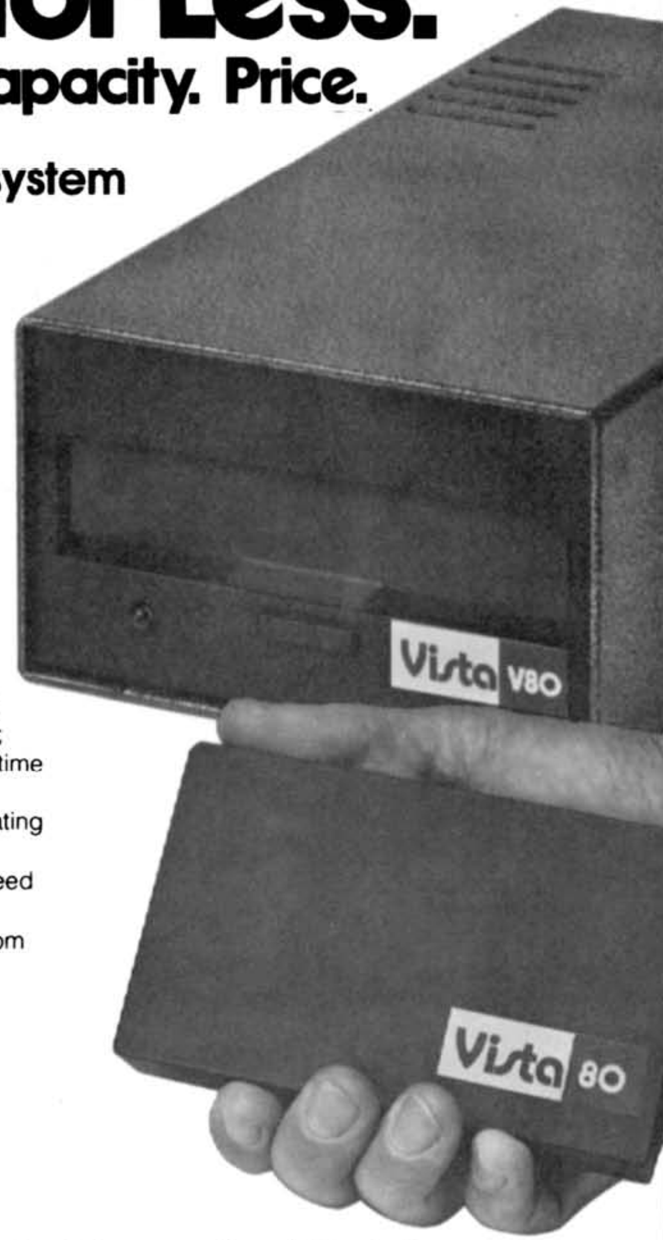
Phone No. ____

If dealer, provide resale no. ____

*TRS-80 is a registered trademark of Radio Shack, a Tandy Company

More for Less.

Speed. Capacity. Price.



The Vista V-80 mini disk system is 8 times faster than the TRS-80, 23% more storage capacity, and costs only $395.

Compare our performance to Radio Shack's TRS-80*. Then match our price with theirs. Then decide which one is for you.

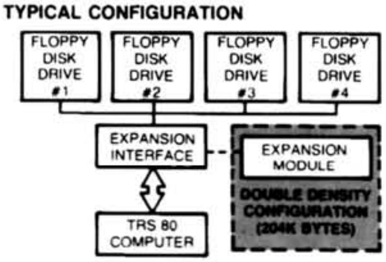
Features

- Vista offers 102K bytes to Radio Shack's 89K. That's 13K more bytes per drive for Vista.
- The V-80 operates at 12ms versus 40ms for TRS-80. Our drive can operate at 5ms, but only 50% of TRS-80 will operate at that speed; therefore, Vista has purposely set the access time at 12ms.
- Totally compatible with all available disk operating systems.
- Upgraded system. Increased storage and speed patch supplied at no charge by Vista.
- Drives are interchangeable for any location from Drive 0—thru Drive 3.
- Immediate Delivery.
- **120 Day Warranty**

Prices:
Single Drive System $395
Two Drive System $770
Four Drive System $1450

*TRS-80 is a registered trademark of Radio Shack, a Tandy Company

TYPICAL CONFIGURATION



Single Density Configuration (102K Bytes)

Vista Expansion Module

The expansion Module provides a double density modification to your current Radio Shack interface that allows you to format diskettes in either single or double density. In double density format, your Vista Drive increases your storage capacity up to 204K bytes on a single 40-track drive.

To insure the highest performance possible, without compromise, we recommend that you use Vista disk drives in conjunction with our Expansion Module.* For a demonstration on your system call **TOLL-FREE 800-854-8017**

Price $239.00 (includes all hardware and software)

Vista cannot guarantee Radio Shack drives to operate 100% in double density.

Vista

The Vista Computer Company 1401 Borchard Street • Santa Ana, California 92705 • 714/953-0523

80 NEWS

by Nancy Robertson

VPI Hands-on Workshop

The chemistry department of the Virginia Polytechnic Institute and State University, Blacksburg, VA will be offering a workshop to teach users how to interface and use the TRS-80 in instrumentation and automation systems. The workshop will be held June 23-27.

The program will emphasize participation by students in designing interfacing and other hands-on learning.

For further information contact Dr. Linda Leffel at the Institute (703) 961-5241.

News for VTOS Users

If you are currently running, or interested in running under the VTOS Operating System, Virtual Technology, Inc. Dallas, TX would like to bring up two patches:

1) In case you have not received Maintenance Release #2—(for sequential files), it is:
PATCH SYS0/SYS.RVEJARAJ using—
X'4744' = 00

2) In case you have not received Maintenance Release #3—(for variable length records), it is:
PATCH SYS0/SYS.RVEFARAJ using—
X'4864' = CD F0 4A
X'4AF0' = DD 71 08 DD 75 0C DD 74 0D C9

In addition Virtual has three new products for the Radio Shack TRS-80 Model I level II microcomputer system: the VTOS 3.1 Operating System, the VTOS 3.1 Reference Manual, and the VTOS 3.1 System Kernel.

VTOS 3.1 contains modifications to the FORMAT, BACKUP, and CHAIN utilities that provide better service for single drive users and improvements to the PATCH utility. New MEMORY and ALLOC commands yield faster memory and disk resource management.

Combined with these are keyboard and printer speed buffers which enable you to type-ahead of the system without missing keystrokes.

The VTOS 3.1 Reference Manual contains the detailed functioning of each command and also provides an insight into the theory of operation.

The VTOS 3.1 System Kernel contains all of the file maintenance facilities required by an assembly language program without operating system utilities. The VTOS 3.1 System Kernel is available to software vendors, under a license agreement, to be used as the distribution media for the vendor's own software.

Reader Service ✓ 179.

Waltham MA

Anyone interested in learning more about the Exatron Stringy Floppy is invited to the first East Coast ESFOA workshop, Saturday, May 31 at Micro Communications, 80 Bacon St., Waltham MA. Units available for hands-on experience; refreshments available. Workshop will run from 10 to 6. Phone (617) 899-8111 for more details or directions.

Centronics Walk-in Service

Centronics Data Computer Corporation of Hudson, NH will offer an option by September 1981 of 21 Centronics service centers in the U.S. where Centronics distributors, dealers and printer owners can return their machines to the local service center for repair or routine maintenance, rather than calling for a service "house-call".

Seven U.S. locations now have this service. They are located in Woburn, MA; Philadelphia, PA; Chicago, IL; Dallas, TX; Sunnyvale, CA; Costa Mesa, CA; and Hudson, NH.

The company's walk-in service will repair or replace a machine within five working days, with priority service available if needed. The walk-in service will save customers 30 percent on the regular Centronics service visit or contract.

Data Base System

V. R. Data Corporation, Folcroft, PA, has announced a new Data Base System for the TRS-80. Customized records can contain up to 25 user-defined variable length fields and up to 250 characters per record.

The system maintains a dictionary of the fields and their characteristics and manipulates any record by its field name.

Field changes can be applied on individual fields or generically to the file. Records can also be linked into a logical "family" of records regardless of their physical location on the file or order of creation.

Reports can be printed in either horizontal or vertical formats.

This four program BASIC system requires a Model I with 48K, a minimum of two disk drives and a line printer. Programs are also available for Model II at 950 records per diskette.

Reader Service ✓ 161.

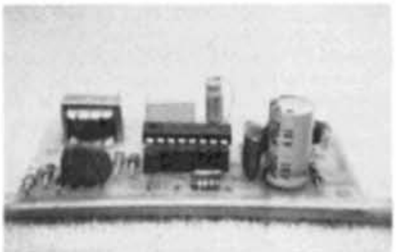
Children's Computer Camp

Dr. Michael Zabinski, Professor at Fairfield University, will be directing an overnight summer camp in Moodus, CN for ages 10–17 where the main activity will be computers.

June 29 to July 4 small group instruction will be offered to campers on TRS-80 computers.

The recreational facilities of the Grand View Lodge include swimming and tennis.

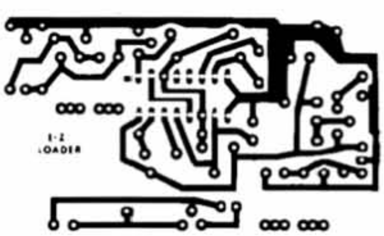
Contact Michael Zabinski, PhD. at 203-795-9069 or write Computer Camp, Grand View Lodge, Box 22, Moodus, CN 06469.



E-Z Loader Assembly

Tape for Easy Loading

The E-Z Loader tape digitizer developed by Dave Miller and Paul Goelz accomplishes three tasks critical to reliable tape loading. It outputs constant level pulses, even if the input pulses are varied, facilitating playback. It eliminates double triggering by the computer on the same pulse, which facilitates loading outside tapes. It only "looks" for a pulse when one is supposed to be present, which helps prevent false triggering of the computer on noise pulses thay may be on the tape.



E-Z Loader Circuit

Circuit boards and complete assembly instructions are available for $6.00.

Reader Service ✓ 182.

General Ledger System

The General Ledger System for the TRS-80 Model II is now available from Taranto and Associates, San Rafael, CA.

This system features unlimited inherent files, a year-to-year comparison on the income statement and the balance sheet, account transaction summary reports up to a full year and the automatic posting of retained earnings to user-defined accounts.

The company asks customers to fill out a questionnaire about their needs so that the system will fit specific business requirements.

The General Ledger Sytem is available for $249.95.

Reader Service ✓ 163

Package Integrates Model II Applications

For the first time, a system is available for the TRS-80 Model II user that completely integrates application development all the way from a professional operating system through database management and documentation. The package, called the Advanced Application Development System, is being offered by the Software Firm, Inc., Denver, CO.

In use for several years on other Z-80 based systems, this development system has reduced average development time by over 50%. With the new package no modifications are required to the Model II.

Device independence is included in the design, which greatly expands the hardware options available for the Model II by providing for straightforward incorporation of devices that Radio Shack does not support, such as spinwriters, additional terminals, special keyboards, line printers, plotters, etc.

The package costs $595 for the Operating System, Datamaster, Monitor, Basic and Script. Other components are individually priced.

Reader Service ✓ 171

Reference for Educators

The *TRS-80 Microcomputer Sourcebook for Educators*, a comprehensive guide to the use of microcomputers as a medium and an object of instruction in the classroom, and as a tool for the school administration, is available free from Radio Shack.

The booklet briefly describes the use of the microcomputer in the classroom and provides guidelines for selecting a system based on potential applications, costs, service, reliability and courseware.

Reader Service ✓ 172

From CP/M to CP/M 2

Lifeboat Associates, New York, NY, is now offering CP/M 2.0 for TRS-80 Model II computers. It makes it possible to run the existing CP/M programs for business accounting, word processing, scientific and special application programming on the new TRS-80 Model II.

The entire system with complete documentation costs $170.

Reader Service ✓ 167

Model II Compilers Developed by Microsoft

Microsoft, Bellevue, WA, is announcing TRSDOS-compatible versions of their COBOL and BASIC compilers for the TRS-80 Model II. The company is the author of Radio Shack's BASICs, and complete line of system software for 8080, Z80 and the new 16-bit microprocessors.

The COBOL-80 compiler is an ANSI-74 implementation of COBOL, featuring complete interactive screen handling capability using ACCEPT and DISPLAY and has indexed and relative files. The BASIC compiler produces highly optimized object code that runs 3-10 times faster than interpreted BASIC programs and is also available in a version for the TRS-80 Model I computer.

The BASIC compiler is priced at $395; the COBOL-80 at $750.

Reader Service ✓ 165

TRS-80 Leasing Available

Radio Shack has announced that it is now possible to lease a TRS-80 Model II. A&A Financial Corporation is offering a 36-month lease, preceded by a 90-day warranty period.

According to Vice President Richard W. Cree, of A&A Financial, "The true lease allows the customer to pay only for the use of the equipment. Payments do not apply towards ownership of the system, and customers have no obligation to buy at the end of the lease term.

Further information on leasing a TRS-80 Model II is available from participating Radio Shack stores and dealers, and Radio Shack Computer Centers, and from A&A Financial Corporation, Fort Worth, TX 76102.

New Numeric Key Pad

Microcomputer Technology Inc., Westminster, CA, has announced a 16-key numerical key pad kit for the TRS-80. Keys include 0 thru 9, (-), (/), (.), backspace and enter key.

Reader Service ✓ 184

Numerical Key Pad Kit

TRS-80 Converts to Duplex ASCII Terminal

Percom Data Company, Garland, TX, has a program available which will allow a TRS-80 computer to be used as a full duplex ASCII terminal.

The program, called TERM80 is written in Level II BASIC, and loads a machine language program for terminal operation. It is available on minidiskette with user instructions for $24.95.

Reader Service ✓ 166

Model II Word Processor

WordMagic II, by CalData Systems, San Diego, CA, is a word processor designed specifically for the TRS-80 Model II. It has total TRS-80 file compatibility, full cursor control, full edit capability, paging, printing, centering, automatic generation of a table of contents, etc.

CalData Systems is asking $100 for the word processor with its manual. The manual is available separately for $100.

Reader Service ✓ 178

Video Screen Filters

An optical filter that eliminates the glare on video screens and extends the life of CRT tubes is available from Sun-Flex Co. Inc., San Rafael, CA. There is no loss of resolution and the entire video display remains in view once the filter is in place.

The price of the filter is $22.50.

Reader Service ✓ 181

Sun-Flex Video Filter

Spikes & Hash Reduced

Electronic Specialists' Inc., Natick, MA, announced Super Isolator to curb electrical problems such as power line spikes, surges, noises and hash. The device is intended to prevent problems with crashes, memory loss and glitches.

The Super Isolator Model ISO-3 sells for $85.95.

Reader Service ✓ 183

Electronic Isolator that Cuts Hash

10 Additional Megabytes

Lobo Drives, Int., Goleta, CA, has three new 10 Megabyte Winchester technology hard disk memory systems; one each for the TRS-80, Apple and the S-100.

The Lobo Model 7710 T, A, S provides a way to add 10 Megabytes of high speed mass storage.

Quantity prices begin at $4995 with volume discounts available to dealers.

Reader Service ✓ 176

Program to Correct Cassette Volume Levels

The Verifier, a program available from Wolf and Associates, La Crescenta, CA, is designed to correct volume level for loading a prerecorded cassette, to indicate the correct volume for playback on cassettes you tape yourself and to test the quality of a blank cassette.

Wolf and Associates are selling this package for $14.95.

Reader Service ✓ 173

Keyed Access and Spooling Facility

Two new packages by Automated Resource Management Inc., Irvine, CA, are on the market.

KFS-80 is a subroutine which can be incorporated in an application program to provide keyed and sequential access to multiple files within the TRSDOS. The package also includes a utility program for file maintenance and a demonstration program. This disk package costs $49.95.

LPSPOOL, a two-disk package, is a line printer spooling facility for the TRS-80 Model I. It permits concurrent printing in the foreground while normal TRSDOS operation continues in the background. Automated Resource Management Inc. is charging $39.95 for LPSPOOL in either the 32K or 48K version.

Reader Service ✓ 175

Aids for Data Entry

The Universal Data Entry System, by the Software Store, Marquette, MI, aids operators of CP/M systems in efficient and accurate data entry. Features include validation procedures to improve data quality and duplication of field items to eliminate repetitive entries.

The two program system is priced at $195.

Reader Service ✓ 177

MOD II Software

Small Business Systems Group has converted its most successful TRS-80 MOD I software programs for use on the TRS-80 MOD II system.

Accounts Payable, Accounts Receivable, General Ledger, Payroll and Dental Billing are available for the one disk user. For the two disk user, we have A/R coordinated to GL, A/P coordinated to GL, and Payroll coordinated to GL, and an expanded Dental Billing.

Reader Service ✓ 160

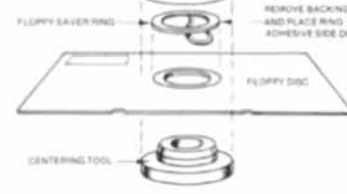
Floppy Saver

Floppy Saver, a reinforcing ring designed to lengthen the life of mini-disks, prevents damage to the center hole by the clamping hub and rotating spindle. Such damage can cause loss of the data and the disk.

Floppy Saver is made a 7-mil (.007) mylar with paper-protected super adhesive backing. The rings are punched on a special steel die with a tolerance not exceeding .0005 inches. The Floppy Saver is installed with the special tool provided and in many cases can resurrect damaged mini-disks.

Floppy Saver is available from Tri-Star Corporation, Grand Junction, CO in a kit of 25 rings for $14.95.

Reader Service ✓ 169.



Rings for Floppy Saving

MICROSOFT AND TRS-80™ MODEL II. RUNNING THE SHOW.

TRS-80 Model II is designed for professional business applications. Your ultimate goal for your Model II is probably a fast, turnkey system that's easy to use and easy to expand. To get there you need dependable, flexible system software to write the programs that run the whole show.

Microsoft's COBOL-80 and BASIC compilers are now available in versions fully compatible with Model II TRSDOS. You can have either of these universally popular programming languages plus all the advantages of a compiler: faster execution times, compact executable code, security for your programs.

With Microsoft's compiled languages you get a complete program development system, including our standard MACRO-80 Assembler and LINK-80 Linking Loader. Your compiled programs are relocatable modules that can be linked to each other or to Z80 assembly language subroutines.

COBOL-80 Compiler

Microsoft's COBOL-80 is an ANSI-74 standard COBOL that supports such advanced data manipulation verbs as COMPUTE, INSPECT, STRING, UNSTRING, and SEARCH. Plus three-dimension arrays, full COPY facility, compound and abbreviated conditions, and an optional packed decimal format that saves on mass storage by as much as 40%. In addition to Sequential and Relative files, COBOL provides Indexed files, allowing records to be retrieved with a user-specified key instead of a record number.

Interactive Screen Handling Most business applications require machine interaction, formatted screen displays, and the ability to insert and delete information as the machine prompts the user. The COBOL ACCEPT/DISPLAY verbs are implemented for this purpose—to DISPLAY formatted screens and ACCEPT operator input.

CHAIN and Segmentation Ideal for menu-driven application programs is COBOL-80's CHAIN feature. With

COBOL-80 formatted screen display

CHAIN, control is transferred from the menu program to any executable module as specified at runtime. COBOL-80 also supports Segmentation to make maximum use of memory when large programs are executing.

BASIC Compiler

The Microsoft BASIC Compiler has the fastest execution times of any BASIC available. It is language compatible with the Model II's interpreter so you can write and test your programs using the interpreter, then compile them for secure storage and efficient execution.

New BASIC Features In addition to those language features provided with the interpreter, the BASIC compiler supports all the latest features of Microsoft BASIC 5.1: WHILE conditional statement, CALL, long variable names, and ANSI compatibility. Plus the compiler has double precision transcendental functions (SIN, COS, TAN, ATN, LOG, EXP, SQR).

Secure Besides being an indispensable system software tool for developing your own utilities and application programs, the BASIC compiler is ideal for producing programs for resale. The machine code for any application program may be placed on a diskette, ROM, or other media, but the BASIC source program need not be distributed. Thus the original application program is protected from unauthorized alteration.

```
 110  I = 4*J: K = (4*J-1) AND &H0FF0
**0025' L00110:    LD      HL,(J%)
**0028'            ADD     HL,HL
**0029'            ADD     HL,HL
**002A'            LD      (I%),HL
**002D'            DEC     HL
**002E'            LD      A,L
**002F'            AND     F0
**0031'            LD      L,A
**0032'            LD      A,H
**0033'            AND     0F
**0035'            LD      H,A
**0036'            LD      (K%),HL
```

BASIC compiler object code listing

Optimized Machine Code Compiled BASIC programs are fast and compact due to extensive optimizations performed during compilation: **1.** Expressions are reordered to minimize temporary storage and eliminate common subexpressions **2.** Constants are folded wherever possible **3.** Peephole optimizations are performed **4.** The code generator is template-driven, allowing optimal sequences to be generated for the most commonly used operations **5.** String operations and garbage collection are extremely fast

Get Microsoft BASIC Compiler or Microsoft COBOL-80 and get serious with your Model II. After all, who's running this show?

Prices for COBOL-80 and BASIC Compiler include the MACRO-80 Assembler and LINK-80 Linking Loader and all documentation. Documentation purchased separately, $20.

COBOL-80	$750.
BASIC Compiler	$395.

For TRS-80 Model I software, contact Microsoft Consumer Products.

MICROSOFT

10800 NE 8th
Suite 819
Bellevue, WA 98004
(206) 455-8080
Telex 328945

242

We set the standard.

Parlez-vous *Fortran? Pascal? Forth? Confused?*

Languages

by Jim Perry
80 Staff

New languages are appearing every month for the TRS-80. Why? Very simply because the built-in BASIC is not really the best language for anything—except learning about computers!

BASIC was developed in the early sixties at Dartmouth College. The name is an acronym for Beginners All Purpose Symbolic Instruction Code. It was designed to be a simple language, easy to understand and learn, but was never intended to be a working programmer's tool.

Because virtually every microcomputer designed in the past five years has used BASIC of some sort or another, it has achieved amazing popularity. Probably 90 percent of the people introduced to computers have started with BASIC. But as one programmer was heard to say, after finishing an elaborate software package in BASIC, "Programming in BASIC is like hitting your head against a brick wall, it's great . . . when you stop."

So, what other languages are available, and why are they better? Last things first. What have the other languages got going for them that BASIC lacks? Read on and discover the amazing world of compilers.

Sugar and Spice

It's not what computer languages are made from; they are, in fact, special computer programs, written in the most elemental computer language, machine language. The purpose of any computer language is to act as an interface between a B.P.U. (Bionic Processing Unit) and a C.P.U. (Central Processing Unit). Most people have a B.P.U. that is capable of expressing 12,000 words, and some have been known to handle up to 35,000 (Mr. Einstein was in that category).

Most C.P.U.s are only able to deal with a hundred or so words. The language program acts as a filter between these two extremes.

Obviously, the more words that a computer language program can understand, the easier it is to talk to it.

BASIC has repertoire of about 100 words, all of which are recognized by the BASIC interpreter. In operation the interpreter examines all the words you care to feed it line by line and word by word. As it recognizes a word, it selects an appropriate set of machine code instructions that do what the word says to do, and then tells the C.P.U. to follow the instructions.

After the C.P.U. hs finished its task, it returns to the interpreter for the next set of instructions. The interpreter then looks at the next word. The entire process takes a very long time. The moral of the story: If you need to get results in a hurry, don't use an interpreter.

Languages such as Fortran (from Formula Transactor, originally developed by I.B.M. in the fifties), are not interpreted in the manner just described, but are compiled.

A compiler program is not as easy to use an interpreter, but it is usually 10 to 20 times faster. After the program is entered into the computer, the compiler is run, converting all the words it recognizes into machine code routines. The result is an object code. The object code is the final program. It is very compressed, and there is no stopping and starting while things are checked out by an interpreter.

The main hassle with this technique is getting all the bugs out. More obvious errors are picked up by the compiler, but if the object code is faulty you have to fix the source and compile again. Compilers also require a lot more memory than interpreters.

The structure of Fortran is still similar to BASIC; to some extent, BASIC was designed with advancement to Fortran in mind. Fortran shares a serious drawback—it is very difficult to read a long program and understand it. With a few GOTO23456s and GOSUB4563s to hide the trail, the logical flow of a program becomes about as clear as mud. Both languages were designed before the concept of structured programming had been fully developed.

Other Options

Languages such as C and Pascal have been designed with structured programming very much in mind. With these languages, you call routines by name and define your own words where needed. With power like this you are encouraged to write readable programs: IF DOOROPEN THEN CLOSEDOOR is an example of a self-explanatory program line.

If you can't face the thought of learning assembly language but still want extra speed in your BASIC programs, then Infinite BASIC from RACET computes may be useful. With this system you have a selection of efficient subroutines from which to choose. Infinite BASIC modules are loaded as subroutines that can be called from a BASIC program. For a graphics program you can select various subroutines and draw lines or scroll the screen at will. The modules are available to perform a hundred separate functions, all in assembly language. They are extremely fast. With powerful add-ons such as this, the usefulness of BASIC can be greatly extended.

In this issue of *80 Microcomputing*, there are reviews of tiny Pascal, Level III and a program for creating your own compiler (written in BASIC). In future issues we will have reviews and explanations of all the other TRS-80 compatible languages that are on the market. Remember, BASIC is just the beginning! ■

IF YOU CAN READ, YOU CAN WRITE !

80 is mainly written by people like you, not an elite group of intellectuals. A lot of the readers think that it is difficult to write, but it is remarkably easy if you sit down and try. Other budding authors think that rejection slips are the rule, at 80 we accept about 70% of all submissions. There is a lot to be said for creative writing, including FAME & a small FORTUNE. There is even a helpful leaflet, "How to write for 80," available on request. So why not drop us a line and ask if we would like to see your article idea written up, it can't hurt to ask.

Send all correspondence to Jim Perry, Managing Editor, 80 Microcomputing, 80 Magazine Street, Peterborough, NH 03458.

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 $69.95.*

The **TRS-80*** is undoubtedly one of the best small computers around. But its cassette recording system can be very frustrating, particularly if you can't read an important cassette. And getting the volume control set just right is really a pain. Waiting 45 seconds to load "Blackjack" is no fun either.

JPC Products Company has developed an improved cassette system that uses your present cassette recorder but loads programs 5 TIMES faster, with much better reliability.

The original JPC cassette interface is the TC-3, which was developed 2 years ago for the M6800 computers. It has been very successful, as you can see from the magazine and owner comments below. Now we have a similar unit for the TRS-80*. At $69.95 plus shipping the TC-8 is a real value.

PRAISED IN REVIEWS

"The JPC Products Model TC-3 cassette interface provides a VERY RELIABLE means of saving programs and data to tape . . . . both fast and reliable. The TC-3 is very convenient to use. The hardware is excellent . . . . The TC-3 is rated AAA because it is an excellent low cost alternative to a floppy disk system for the person on a tight budget."

Product Review
68' Micro Journal
June 1979

OWNERS LOVE IT

"I love your interface. I have recommended it to all." *D.O. (Florida)*

"Excellent! A poor man's floppy. Thanks." *F.B. (Quebec)*

"It is great. I can't believe it's so fast." *E.T. (Arkansas)*

"Great! Fast! Reliable! Worked first time!" *C.Z. (New York)*

"It is the answer to my prayers!" *S.S. (New York)*

"The best through-the-mail device I have ever purchased." *J.P. (Florida)*

"I'm glad I bought this kit. Worked first time. Never one bad load." *R.M. (New York)*

"The first kit I have ever built that worked the first time without any adjustment or trouble-shooting." *F.L. (Colorado)*

INEXPENSIVE MEDIA

The TC-8 and your present cassette recorder will allow you to store 50,000 Bytes on a standard 10 minute cassette. Or 300,000 Bytes on a standard 60 minute cassette. At a cost of $1 to $3. Our C-10 data cassettes sell for $1.39 with a money-back guarantee.

PLUG IT IN

The TC-8 plugs into the expansion connector on the back of the keyboard and does its thing 5 TIMES FASTER! Less than ONE BAD LOAD in a MILLION BYTES! With the VOLUME CONTROL ANYWHERE BETWEEN 1 AND 8.

If you prefer you can leave the existing recorder connected normally and add a second one for the TC-8.

THERE'S A CATCH

The TC-8 magic is mostly done in software. So you have to load a small program into the upper seven hundred bytes of memory. It is usually out of the way there. We provide the software on a cassette that comes with the TC-8. Just load it in.

*TRADEMARK OF TANDY CORPORATION

YOU CAN BUILD IT

The TC-8 is available as a kit for $69.95 plus shipping. It is very easy to build. It should only take you an hour or so. Even if you have never built a kit before, you can build the TC-8.

If you can get toothpaste on a toothbrush, you can learn to solder. Our instruction manual will show you how.

"CAN'T FAIL" GUARANTEE

If you build the TC-8 and for any reason it doesn't work, we will make it work at NO COST. All you have to pay is the shipping. We guarantee it.

WE WILL BUILD IT

The TC-8 is available fully assembled for $99.95 plus shipping. We ship assembled units 3 weeks after we receive your order. But truthfully, with our "CAN'T FAIL" guarantee, we don't understand why you wouldn't rather do it yourself.

ORDER NOW

To order your TC-8 kit, send your check or money order for $69.95 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.

At the present time, the only version of the TC-8 available is for 16K LEVEL II SYSTEMS.

✓190

JPC PRODUCTS CO.
Phone (505) 294-4623
12021 Paisano Ct.
Albuquerque, N.M. 87112

Two versions of a "stripped down" Pascal for your 80.

Pascal I & II

Fred Monsour
309 Camellia Drive
Charlottesville, VA 22903

Has programming in BASIC lost its challenge? Are you ready to try something new, but without the headaches of assembly language?

If you haven't considered Pascal, now is the time. There are at least three Pascal software packages available. The first release from the FMG Corporation is a powerful tool, but its price, $150 for the complete system—$100 without its assembler, linker and library—is steep for those who are merely curious.

Even more restrictive is FMG's 48K RAM requirement and at least two disk drives.

The other two systems are People's Pascal I and People's Pascal II, both marketed through the Computer Information Exchange, Inc. (Box 158, San Luis Rey, CA 92068), and somewhat more affordable at $15 and $23. Though only a subset of full Pascal (these systems are to Pascal what Tiny BASIC is to Level II), there's enough to be interesting, fun and perhaps even useful.

To use People's Pascal: load the editor/compiler and either type in or load from tape a program; compile P-code; optionally run and debug the program using the interpreter; translate P-code to Z80 code; load run-time system; optionally load T-BUG (not supplied) and save Z80 code and run-time routines on tape; run program.

Editor/Compiler

Depending on line length, the editor/compiler can handle 50 to 200 lines of Pascal source text at a time. Larger files can be created, edited and compiled by saving portions of the large file on tape as the text buffer is filled. Source text of indefinite length can be handled in this way since the compiler alternately reads from the source tape and writes to the object (P-code) tape.

All actions required of the user are prompted. The compiler is set up for use with one cassette recorder, but it can be easily modified to work with two (see Example 1). This eliminates having to switch source and object tapes back and forth manually during compilation. Editor/compiler commands are:

F(ree): Give number of bytes left in text buffer.
L(ist): List line(s) of text on screen.
P(rint): List line(s) of text on printer.
D(elete): Delete line(s) of text from buffer.
N(umber): Renumber lines of text.
R(ead): Read block(s) of text from tape.
W(rite): Write line(s) of text to tape.
E(of): Write end-of-file mark to tape.
C(ompile): Compile program in text buffer.

As the compiler reads the source code from the text buffer and/or cassette files, it produces an intermediate code, called P-code, which is later translated into Z80 machine code. During compilation the source and object code are displayed on the video monitor, and the object code is written to tape.

Compiler options include output to line printer and compilation without generation of an object file. The latter is used to quickly check for syntax errors before the object code is written to tape. If an error is found the compiler not only gives a message, but points to the faulty section.

```
Line 193: Change PRINT #-1 to PRINT #-2
Line 234: Change "WRITE CAS1"
  to "WRITE CAS2"
Line 235: Change PRINT #-1 to PRINT #-2
```

Example 1. Modifications to editor/compiler to allow using two cassette recorders.

P-code Interpreter

Once the compiled P-code is on tape, it can be either translated into Z80 code or executed via the interpreter which, like the editor/compiler and the translator, is written in BASIC. While the interpreter is extremely slow because of its double level of interpretation (P-code to BASIC to Z80 code), this effect is useful for debugging a program.

In addition, commands are provided for controlling and analyzing program execution. These are:

R: Run program.
S: Single-step execution of program.
G: Go to and execute next program instruction.
X: Display counters, registers and pointers.
T: Trace through last few P-codes executed.
K: Display specified stack locations.
B: Set breakpoint(s).
C: Clear all breakpoints.
Y: Display all breakpoint locations.
E: Examine specified P-code locations.
N: Examine next P-code location.
U: Examine previous P-code location.
Q: Quit interpreter.

While understanding the workings of the Pascal P-machine is necessary to use some of these commands properly, the interpreter can still be of value without knowledge of the P-machine's complexities.

Once you're satisfied that the P-code is correct, with or without debugging on the interpreter, the translator then gives your program its final form—Z80 code. After a short initialization dialogue, the translator reads the P-code from tape and begins the conversion. The P-codes and corresponding Z80 codes will be optionally listed on the video monitor and/or the printer.

At the end of translation, the start and end addresses of the Z80 code and the Z80/P-code ratio are displayed. One nice feature of the translator is that conversion can be customized for speed or for size. A program at optimal speed runs about twice as fast and requires

UPI

PASCAL

FORTRAN

APL

STOCK QUOTES

TRS-80™ Super Version #1

Why limit yourself to Radio Shack® BASIC or spend hundreds of dollars on new programs and languages?

Through the phone lines, your TRS-80™ becomes a sophisticated and powerful computer. Access new languages, make vital decisions based on stock quote, earn money selling those top notch programs you write.

Two services, MicroNet™ and The Source™ open the world to the 80 user. Pick the group for your needs or both!!

MicroNet
- Hookup $9.00
- Per hour $5.00

The Source
- Hookup $100.00
- Per hour .... $ 2.75

These charges are billed to your Master Charge or VISA.

Houston Micro Computer Technologies, Inc. offers you three packages to choose from:

Package I (ST80III, Accoustic Modem, RS232C) with:
- MicroNet .... $460.
- The Source .... $540.
- Both .... $545.

Package II (ST80I, Accoustic Modem, RS232C) with:
- MicroNet .... $400.
- The Source .... $480.
- Both .... $485.

Package III for Mod II (HMCTERM, Accoustic Modem, Cable)
- MicroNet .... $500.
- The Source .... $580.
- Both .... $585.

ST-80 I ($25.00)
Set Serial Board from Keyboard
Auto Line Feed
Line Printer on

ST-80 II ($85.00)
Dump Memory to Disk
Allow Lower Case Letters
Transmit Auto Logon
Transmit Buffer to Serial: Out Port
Serial: Into Memory

ST-80 III ($150.00)
Help Command
Auto Answer Auto Originate
Switch Status
Warm Program Restart
Job Log

HMCTERM ($200.00)
Converts TRS-80 Model II to Complete Intelligent Terminal for Processing with Major Mainframes

---

To order by phone or for local dealer information call: **713/661-2005**
Texas residents add 6% sales tax • MasterCharge • Visa

HOUSTON MICRO-COMPUTER TECHNOLOGIES, INC.

Home and Business Computer Specialists

5313 BISSONNET • BELLAIRE • TEXAS • 77401 • 713/661-2005

13

Dealer inquiries invited.

```
(• UNLIMITED PRECISION DIVISION IN PASCAL •)
(• AFTER A BASIC PROGRAM BY JEF RASKIN, BYTE, FEB '79 •)
CONST DECIMALPOINT = 46 (• ASCII VALUE •);
VAR DIVDEND, DIVISOR, QUOTIENT, PRECISION: INTEGER;
BEGIN;
  READ (DIVDEND#, DIVISOR#, PRECISION#);
  QUOTIENT := DIVDEND DIV DIVISOR;
  WRITE (QUOTIENT=, DECIMALPOINT);
  REPEAT;
    DIVDEND := (DIVDEND--QUOTIENT•DIVISOR) • 10;
    QUOTIENT := DIVDEND DIV DIVISOR;
    WRITE (QUOTIENT#);
    PRECISION := PRECISION - 1;
  UNTIL PRECISION = 0;
END.
```

Program Listing 1: Pascal program demonstrating unlimited precision division.

about twice as much memory as one designed for size. In a 16K system, roughly 9.5K bytes of Z80 code can be accommodated.

Run-time Routines

Okay, now you have the final product, right? Not exactly.

Have you included keyboard scanning, video output, 16 bit arithmetic routines, etc., in your Pascal program? No? Well don't despair. The authors of People's Pascal I have taken care of them for you. All you have to do is load these run-time routines from tape via the Level II SYSTEM command. They, too, are written in assembly language for easy modification.

When the run-time routines (about 1.4K bytes) are loaded the program is ready to go. Before you run it, however, it's a good idea to save a copy of the program/run-time system on tape. This is easily done using T-BUG or any similar monitor. The final product can then be loaded and run without any other Pascal I software present.

In fact, by modifying two calls to Level II character input and output routines, the program should be compatible with any other Z80 microcomputer (assuming RAM is available in the correct locations).

People's Pascal II

People's Pascal II consists of three sections: the Monitor, which includes the run-time routines and P-code interpreter; the editor, which is used to enter and modify Tiny Pascal source code; and the compiler, which creates P-code from source.

The three are loaded and used as one program via the Level II SYSTEM command, making Pascal II quicker and easier to use. There are actually two versions provided—one for 16K TRS-80s and one for 32K or more. Except for a potentially larger symbol table in the second version, they are essentially the same.

After loading the 16K or 32K version with the SYSTEM command, type "/ (Enter)"; the monitor will respond with a sign-on message and a period, which is its command prompt. Monitor commands are:

E: Enter new source file or edit old one.
C: Compile source file into P-code.
C/-P: Compile but don't generate P-code (for syntax check).
C/-S: Compile; OK to overwrite source if object size requires.
R: Run compiled program.
R/-C: Run program; overwrite Editor & Compiler if necessary.
LS<filename>: Load source from cassette.
LP<filename>: Load P-code from cassette.
WS<filename>: Write source file to cassette.
WP<filename>: Write P-code file to cassette.

The editor does not use line numbers, but maintains a pointer to the current line on which the editing commands operate. These commands list to video monitor; scroll up or down; delete; insert; replace and extend one or more lines. A maximum of 600 lines of text is permitted, each no longer than 130 characters.

Once the Pascal source code is complete, return to the monitor, type "C" and watch your video screen. The text is listed as it is compiled—about one line per second. Syntax errors generate a code number and an automatic return to the monitor.

Before running the program, you may want to save the source file on tape as insurance against software explosions. After it runs to your satisfaction you can save the P-code, which will load and execute without compilation.

If you have at least 36K of RAM you can compile or modify the Pascal source code.

Tiny Pascal Language

Tiny Pascal handles only two date types—integers and one-dimensional integer arrays. This

Continue to 42

is its greatest limitation. However, these constraints can be overcome, as users of Tiny BASIC and integer FORTH may have found. One very simple example is given in Program Listing 1.

Valid Tiny Pascal statements include: assignment of variables and constants; definition of procedures and functions; conditionals (IF . . . THEN . . . ELSE, CASE); repetitives (WHILE . . . DO, REPEAT . . . UNTIL, FOR . . . TO . . . DO, FOR . . . DOWNTO . . . DO); READ; WRITE; compound statements of any of the above.

Not provided are label definition, GOTO, READLN and WRITELN. Procedure and function parameters are passed only by value. There is no record or file-handling capability.

Additional features access memory (like BASIC's PEEK and POKE), use machine language code (like USR or CALL) and allow hexadecimal constants, "else" branches on CASE statements, logical operations and printing (but not manipulation) of character strings.

Pascal I permits one compiler directive, $INCL. When encountered in a Pascal source text, it instructs the compiler to read a source file from tape and insert it into the text at that point. A library of procedures and functions can be developed and reused in this way. In fact, the authors of Pascal I provide the foundation for such a library: PASLIB. It contains the routines SET (for graphics), AT (for cursor positioning) and RND (for random number generation).

Pascal II also has several extra features, all built into its compiler:

ABS (X): Absolute value of x.
SQR (X): Square of X.
INP (X): Value of input port X.
OUTP (X,A): Outputs A to port X.
INKEY: As used in Level II BASIC.
PLOT (X,Y,A): Turns graphics point at X,Y on if A odd, off if A even.
POINT (X,Y): Returns 1 if point at X,Y on, 0 if off.
MOVE (B,A,N): Moves memory block of N bytes from A to B.

Conclusion

Pascal I and II each has its own merits and shortcomings. Pascsl I offers greater versatility when creating source code. It has error messages and a pointer rather than code numbers, output to printer, an interpreter and the $INCL directive.

On the other hand, Pascal I requires the loading of at least three fairly long programs from cassette and runs slowly since it's in BASIC. It can't be put on disk because the PEEKS and POKES conflict with DOS BASIC.

Pascal II, however, can be, via NEWDOS' LMOFFSET or something similar (TRSDOS users see Allan J. Domuret's articles in *Microcomputing* (Oct. & Nov. 1979).

Pascal I has the RND and AT functions; Pascal II has ABS, SQR, INP, OUTP, INKEY, POINT, and MOVE. Each of these can be created in Pascal or through a machine language call if necessary, but it takes a bit of work.

Since each system has certain advantages, it would be nice to use both for programming. Unfortunately, the source and P-code tape format of each system is incompatible with that of the other.

The two systems also differ in their final products. Programs written on the Pascal I system require only a small run-time unit, are more portable and are transferable to other Z80 computers with little effort. Pascal II programs run only on the Pascal II system.

Finally, Pascal I produces Z80 code, which runs significantly faster than the P-code and interpreter combination of Pascal II (see Program Listing 2).

Whichever you choose, one of these systems is definitely worth trying, if only to see first hand why Pascal has created such a stir. Tiny Pascal probably won't displace BASIC as your 80's native language, but it might satisfy your appetite for something new and set you thinking about that Big Pascal yet to come.■

```
10 DEFINT I, X, Y, Z
20 FOR I = 1 TO 10000
30  X = 500: Y = 7: Z = X*Y
40 NEXT I
50 PRINT Z
60 END
```

```
VAR I, X, Y, Z: INTEGER;
BEGIN
 FOR I := 1 TO 10000 DO BEGIN;
 X:= 500; Y:= 7; Z:= X*Y;
 END;
 WRITE (Z#);
END.
```

Level II BASIC: Programming time = 1 min.; running time = 2 min 20 sec.
Pascal I: Programming time = 14.5 min.; running time = 9 sec.
Pascal II: Programming time = 1.5 min.; running time = 23 sec.

Program Listing 2: BASIC and Pascal benchmark programs and comparison of program development and running times.

Faster than a speeding interpreter, this BASIC program compiles your efforts into assembly language.

TINYCOMP

David Bohlke
Coggon, IA 52218

When I first began using some machine language routines, their speed really impressed me. But because I'm mostly interested in BASIC, I thought it might be good experience to write a BASIC compiler.

My system is a Level II TRS-80 with 16K; and TINYCOMP 1.0 is just a beginning.

The actual TINYCOMP program extends from line 800 to line 3800 in the listing. The program to be compiled should be numbered between 1 and 500. Since an END statement is required, it is a good idea just to record it at line 500 with the compiler program.

Machine code is POKEd beginning at location 30000. The only table used begins at 32650 and is used to store the variables (A-Z, scratch) which TINYCOMP processes.

There are only five statements that can be compiled, with several variations (see Table 1).

In general, TINYCOMP PEEKs through the BASIC program one line at a time and branches to various subroutines that select the machine code tokens to be POKEd (see Table 3).

```
10 A = 1:B = 0:C = 191:D = 1024
20 PRINT@B,CHR$(C);
25 GOSUB 300
30 B = B + A
40 IF B = D THEN 99
50 GOTO 20
99 END
300 X = 0:Y = 30
310 X = X + A
320 IF X = Y THEN 340
330 GOTO 310
340 RETURN
```

Sample Program 1

A,B,C	represent any of the legal variables, A-Z
nnn	must be a positive integer
LET A = nnn	LET is optional
LET A = B + C	subtraction also supported
LET A = B	
PRINT@A,CHR$(B);	semicolon optional
GOTO nnn	
IF A = B THEN nnn	= is only test
END	must have
GOSUB nnn	
RETURN	

NOTE: only LET (assignment) statements should be used on multiple statement lines.

Table 1. TINYCOMP Syntax

TINYCOMP Listing

```
10 A=1:C=65:D=1024:E=91
20 B=0
30 PRINT@B,CHR$(C);
40 B=B+A
50 IFB=DTHEN70
60 GOTO30
70 C=C+A
80 IFC=ETHEN99
90 GOTO20
99 END
500 END
800 POKEM,P:PRINTP;:M=M+1:RETURN
805 P=PEEK(Q):Q=Q+1:IFP=32THEN805ELSEIFP=0THENC=2
807 RETURN
810 IFP<65ORP>90THEN815ELSERETURN
815 PRINT:PRINT"ERROR LINE #";L1(L-1):END
820 IFP<48ORP>57THENRETURN
822 C$=C$+CHR$(P):GOSUB805:GOTO820
830 C$="":GOSUB820:IFC$=""THENC1=-1:RETURN
832 C1=VAL(C$)
834 D1=C1/256:E1=C1-D1*256:RETURN
850 PRINT@0,"TINYCOMP 1.0     DAVID BOHLKE     COGGON, IA":RETURN
900 P=42:GOSUB800:GOSUB990:GOSUB800:P=P1:GOSUB800:RETURN
905 P=22:GOSUB800:P=D1:GOSUB800:P=30:GOSUB800:P=E1:GOSUB800:RETURN
910 P=25:GOSUB800:RETURN
915 P=115:GOSUB800:RETURN
920 P=235:GOSUB800:RETURN
925 P=34:GOSUB800:GOSUB990:GOSUB800:P=P1:GOSUB800:RETURN
930 P=195
932 GOSUB800:P=E1:GOSUB800:P=D1:GOSUB800:RETURN
935 P=183:GOSUB800:P=237:GOSUB800:P=82:GOSUB800:RETURN
945 P=32:GOSUB800:P=JR:GOSUB800:RETURN
950 P=201:GOSUB800:RETURN
960 P=225:GOSUB800:P=233:GOSUB800:RETURN
965 P=213:GOSUB800:RETURN
990 P=VT+V1*2-INT((VT+V1*2)/256)*256:P1=(VT+V1*2)/256:RETURN
1000 DEFINTA-Z:DIM L1(50),L2(50)
1010 Q=17129:L=1:VT=32650:M=30000:CLS
1020 M1=PEEK(Q)+PEEK(Q+1)*256:L1(L)=PEEK(Q+2)+PEEK(Q+3)*256
1030 GOSUB850
1040 PRINT:PRINT@960,L1(L);M;":";:L2(L)=M:L=L+1:Q=Q+4
1045 IFL1(L-1)>500THEN1220
1050 C=0:GOSUB805:IFC=2THEN1050
1090 IFP>64ANDP<91THENQ=Q-1:GOSUB2000
1100 IFP=140GOSUB2000ELSEIFP=178GOSUB2500
1110 IFP=141GOSUB3500ELSEIFP=143GOSUB3000
1120 IFP=145GOSUB3700ELSEIFP=146GOSUB3800
1190 IFP=128GOSUB950
```

```
1195 IFPEEK(Q-1)=58THEN1050
1200 Q=M1:PRINT:GOTO1020
1220 GOSUB850:PRINT@960,"ADJUSTING JUMP'S . . .";
1250 FORI=30000TOM:IP=PEEK(I):IFIP<>195THEN1290
1260 DN=PEEK(I+1)+256*PEEK(I+2):DH=0
1262 FORJ=1TOL:IFDN=L1(J)THENDH=L2(J):PRINTL1(J);
1264 NEXT:IFDH=0PRINT"  L N F =";DN:GOTO1290
1270 MB=DH/256:LB=DH-MB*256:POKE(I+1),LB:POKE(I+2),MB
1290 NEXT:PRINT
1300 POKE16526,48:POKE16527,117
1302 GOSUB850:PRINT@960,"";
1305 INPUT"<ENTER> TO  =RUN=  MACHINE CODE . . .";A$:CLS
1310 X=USR(0)
1333 GOTO1333
2000 GOSUB805:GOSUB810:V1=P-64:GOSUB805:IFP<>213THEN815
2010 GOSUB805:GOSUB830:IFC1=-1THEN2100
2020 GOSUB905:GOSUB920:GOTO2150
2100 V2=P-64:GOSUB805
2110 IFP=205ORP=206THEN2300
2120 V4=V1:V1=V2:GOSUB900:V1=V4
2150 GOSUB925:RETURN
2300 SG=P:GOSUB805:GOSUB810:V3=P-64:V4=V1
2310 V1=V3:GOSUB900:GOSUB920:V1=V2:GOSUB900:V1=V4
2320 IFSG=206GOSUB935:GOTO2150
2330 GOSUB910:GOTO2150
2500 GOSUB805:GOSUB805:GOSUB810:V1=P-64:GOSUB805:GOSUB805
2510 IFP<>247THEN815ELSEGOSUB805:GOSUB805:GOSUB810:V2=P-64
2520 GOSUB900:D1=60:E1=0:GOSUB905:GOSUB910:V1=V2
2530 GOSUB920:GOSUB900:GOSUB920:GOSUB915:RETURN
3000 GOSUB805:GOSUB810:V1=P-64:GOSUB805:IFP<>213THEN815
3010 GOSUB805:GOSUB810:V2=P-64:GOSUB805:IFP<>202THEN815
3020 GOSUB805:GOSUB830:IFC1<0ORC1>500THEN815
3030 V4=V1:V1=V2:GOSUB900:V1=V4:GOSUB920:GOSUB900
3040 GOSUB935:JR=3:GOSUB945:GOSUB930:RETURN
3500 GOSUB805:GOSUB830:IFC1<0ORC1>500THEN815
3510 GOSUB930:RETURN
3700 GOSUB805:GOSUB830:IFC1<1ORC1>500THEN815
3710 D2=D1:E2=E1:C1=M+8:GOSUB834:GOSUB905:GOSUB965
3720 D1=D2:E1=E2:GOSUB930:RETURN
3800 GOSUB960:RETURN
```

I have to apologize to devoted T-BUG users for the inefficient codes used, but I have never actually written a machine language program. Rather, I just used the codes I felt I could control.

Execution

To begin, protect high memory for the machine code and variable table by answering MEMORY SIZE? with 30000. This permits about 2600 bytes for the compiled machine language program.

Q	beginning of BASIC line to be compiled
L	line being compiled
VT	beginning address of variable table
M	current location (30000 +) to POKE machine code
M1	beginning of next BASIC line to be compiled
L1(L)	decimal number of compiled line
L2(L)	M location corresponding to Level I(L)
C,C1,SG	flags
P	PEEK value for corresponding M
V1,V2	CHR$ values of legal variables in line being compiled
D1,E1	absolute values for loads, jumps

Table 2. Variable List

Line numbers 10-99 contain a sample program to be compiled. You can key them in with TINYCOMP. After the program is entered, you should still have over 10000 bytes remaining (for TINYCOMP 2.0?).

Be sure to issue a CSAVE before you attempt to run TINYCOMP.

When you type RUN, the sample program is executed in BASIC. This enables you to check for any syntax errors. To compile the sample program, type RUN 1000.

During execution the decimal line number, the beginning memory location of the machine code line to be compiled, and the POKEd machine code tokens are printed. This takes only a few seconds for the sample program. You can now execute the compiled program by pressing ENTER when the prompt appears. The sample program will fill the screen with all A's, B's, C's through Z's. What took about 520 seconds for the BASIC program, takes only 3.2 seconds in the compiled machine code! That's roughly 160 times faster. They use about the same memory, a little over 111 bytes.

I provided another sample program to demonstrate the GOSUB command (see Sample Program 1). The coding paints the screen white, and employs a timing delay in the routine at line 300. By altering the variable Y in line 300 you can change the execution speed.

To exit the machine code, just press BREAK.(If you get stuck in an endless loop, press the RESET button.) To execute the machine language program without having to re-compile, type RUN 1300.

Current Maximum

Currently, the maximum number of lines which can be compiled is 50. This can be adjusted by changing the DIM statement (L I and L II) in line 1000.

Any TINYCOMP syntax errors are displayed as ERROR LINE # nn (see Table 1). There is no provision for CSAVEing or CLOADing the machine code, but the combined BASIC programs (TINYCOMP and the sample program) can be saved and loaded as usual.

Although there aren't a lot of practical applications for a program that permits only five statements, I hope TINYCOMP 1.0 can be a learning experience for you as it has been for me.

Some of the statements I would like to add are PRINT"STRING", PRINT a variable, INPUT, RND, and the SET and RESET. With over 10000 free bytes remaining, there should be sufficient memory to make the TINYCOMP statement set at least as large as Level I. I would appreciate any helpful suggestions. ■

800-832	Frequently used subroutines
800	POKEs machine code in 30000 +
805	PEEKs Basic program
810	check for variable type
815	prints ERROR message
820	finds integer value of constant
830	check for constant type
900-990	Assembler codes to be POKEd
900	LD HL(nnnn)
905	LD D nn LD E nn
910	ADD HL,DE
915	LD(HL),E
920	EXC HL,DE
925	LD(nnnn),HL
930	JP nn
935	OR A SBC HL,DE
945	JRNZ
950	RETURN (for END)
960	POP HL JP(HL)
965	PUSH DE
990	computes (nnnn) for 900,925
1000-1333	Main control loop, steps through BASIC program
1000-1010	sets parameters
1020-1045	save decimal and POKE line numbers
1050	gets statement code
1090-1200	branch on statement codes
1240-1290	POKEs absolute addresses (GOTO,THEN,RETURN)
1300-1333	executes machine language program
2000-2330	compiles LET
2500-2530	compiles PRINT@
3000-3040	compiles IF . . . THEN
3500-3510	compiles GOTO
3700-3800	compiles GOSUB . . . RETURN

Table 3. Line Description

An outline of the commands for a new language.

PUT-N

William O'Brien
11 Dongan Place
New York, NY 10040

Higher level structured languages for the TRS-80 are in short supply for the serious programmer just now. Yes, of course, there is Microsoft's FORTRAN, but one must know the language thoroughly in order to use it. The need arises, therefore, for a high level language with simple instructions that allow a programmer to implement it immediately on his system.

It is for this reason that I am proposing the use of the following structured statements, if anyone would care to compile the language along these guidelines.

PUT-N

A High Level Command Language

USER	SYSTEM RESPONSE
HELLO	OK . . .
MEM?	YOU'VE GOT ROOM (indicates a large area of user RAM).
	ITS A LITTLE CROWDED IN HERE (half of the available RAM is no longer available).
	NOT REALLY (very little RAM is available. The CPU will also respond with this when there is so little RAM available that it doesn't remember the question.).
FOR . . . WHEN . . . DO	for all the possibilities suggested, when the value or conditions are as you wish them to be, do whatever it is you want done. This is excellent for analog inputs. If, for instance, you see that you will be busy for the rest of the year and you want to be reminded when Christmas arrives.
IF . . . WHAT	a conditional statement such that should the conditional occur, the CPU will advise you how to proceed.
ON ERR, WHAT?	if an error occurs, print just what kind of an error it is and how to correct it.
ON ERR,FIX!	for appliance operators who do not wish to be bothered with statement errors. The CPU is instructed to correct the error on its own. The importance of this instruction can not be over emphasized.
HUH?	should you receive an error message that you do not understand, the CPU will try to explain it more clearly.
HOLD	should the programmer find it necessary to leave for a while, this instruction allows the CPU to shut most of itself off in the interim.
BACK	cancels HOLD.
VARLST	lists all variables not in use as of its implementation.
IF FULL NO	when printing to screen this forces a stop should the screen become filled and start to scroll. Excellent for preventing eye strain.
IDLE	allows the processor to do whatever it wants to for a while if the programmer wants to take a break.
BACTOBUS	cancels IDLE.
READY?	similar to HELLO, CPU responds with OK, notifying the programmer that it will accept instructions.
AIM	for interfacing the TRS-80 with a Rockwell Computer, it provides the necessary output data.
FIRE?	for emergency use only, should the programmer see smoke near the CPU or any of the peripherals.
WHATDO	for the novice programmer, this instruction allows the CPU to explain its various capabilities.
CLOAD	for the experienced programmer and novice alike. This loads a program from tape and causes an automatic Checksum error.
RCDR	causes the CPU to print what model tape recorder is on line, since this is an abstract choice.
INTERERR	implements an intermittent error

I made the TRS-80* into a serious computer. Now I've made the Model II into a spectacular one.

I'm Irwin Taranto, and I've helped almost a thousand businesses get their first computers up and running.

I've done it primarily with the TRS-80, because it's a really elegant piece of hardware. Given the right programs, it can do substantially the same work as the traditional minicomputers that cost four times as much.

I proved it with four on-line, interactive programs adapted from the genuine Osborne & Associates systems, originally designed for the $30,000 Wang computer. Then I added two of my own and made them all work on a $4000 TRS-80.

Now I've done the same thing for the new TRS-80 Model II. It's an $8000 computer that works twice as fast and has four times the memory—up to two million characters.

My new systems are fully documented, and because I'm working with a much more powerful computer, they're a night-and-day advance over the Model I programs. They'll turn your Model II into a complete business computer, set up and ready to go.

THE TRS-80 MODEL II PROGRAMS

General Ledger/Cash Journal: handles up to 7000 transactions on 500 different user-defined accounts. It keeps track of them by month, quarter and year, makes comparisons to the prior year, and does departmentalization.

Accounts Payable/Purchase Order: generates the purchase order and posts the item to payables when the goods are received. Invoice-linked, it calculates and prints checks and aged ledger reports and links fully to the general ledger.

Accounts Receivable/Invoicing: keeps track of billed and unbilled invoices, open and closed items, aging and service charge calculation. It prints statements, links to the general ledger, and can work within either an invoice-linked or balance-forward accounting system.

Payroll/Job Costing: computes regular, overtime and piecework pay, keeps employee files, figures taxes and deductions, prints checks, journal, 941-A and W-2 forms, and breaks out individual job costs.

*A trademark of the Tandy Corporation

When I say set up and ready to go, I mean just that. If you're not quite sure on that point, call the number below and we'll give you the names of some of the people who've already bought all over the world. Call them up and hear what they have to say.

These Model II programs are completely custom-tailored, which explains their $249.95 price. Before we'll send you a disk, you have to fill out a detailed questionnaire that tells us your precise business requirements. Then we send you the disk, all the instructions you need, and my phone number. If you call, we answer all your questions. If your questions are tough enough, I'll talk to you personally.

Because that way I'll make sure that Model II of yours turns into a spectacular computer, just like I promised.

☐ Please send me the custom questionnaires for the following $249.95 Model II programs:

- ☐ General Ledger/Cash Journal
- ☐ Accounts Payable/Purchase Order
- ☐ Accounts Receivable/Invoicing
- ☐ Payroll/Job Costing

☐ Please send me information on the TRS-80 Model I programs at $99.95 each

☐ Please send me information on other Taranto business programs

Your name ____________

Company name ____________

Address ____________

City/State/Zip ____________

✓45

Taranto
& ASSOCIATES, INC.

P.O. Box 6073, 4136 Redwood Hwy., San Rafael CA 94903 • (415) 472-2670

✓Reader Service—see page 162

USER	SYSTEM RESPONSE
	that disappears when the power is shut off (as is the case when taking the CPU for repair of same).
WAIT . . .	this instruction should be used just after ordering a Disk Drive from the manufacturer. It disables all Disk Operating features for five months thus eliminating any chance of an L3 error.
MNUL	forces an output from the CPU, causing it to print what the instruction manual really means or should say, rather than what it does say, since they are not always the same.
EDIT	allows the CPU to scan the job ads in *The New York Times* and list all journalism offerings.
BACKUP!	forces the CPU into an error routine when brought in for service to uphold your claim that it isn't working right.
SORTMTPL	CPU inputs the various answers given by various Radio Shack personnel and sorts through them to check for a possible valid answer among them.
FORMAT	when writing a program for your brother Matthew, it allows the CPU to insert his name in the first REM statement (this will not work with other names).
APPEN	outputs instructions as to how an unprintable symbol might be written.
MERGE	allows the CPU to notify you when you've spent so much on Radio Shack peripherals that you should take out a loan.
BYTECOUNT	outputs how big a chunk of your income is going toward the electric bill.
ACTWAIT	after inputting the waiting time for a peripheral from Tandy, this function allows the CPU to calculate the actual time you should allot.
HANGUP	utility that permits the CPU to get hung up on a problem whenever the programmer invites a friend over to see the wonderful thing.
LLIST	a typo, ignore it.

ARITHMETIC OPERATORS

.ME. might be equal to	.PE. is probably equal to
.GL. greater or less than	.NE. not equal or equal to
.SC. similar or congruent to	.AE. actually equal to
.PS. perhaps the same	.AD. addition operation
.SU. integer subtraction	.LD. long division
.SD. short division	.DC. divide and conquer
.MU. an ancient island	.SA. sequential addition (multiplication)
.FI. forget it	.WE. whichever is easier
.LA. lose the answer	.MA. make up an answer
.NY. new york	.IC. impossible to compute
.WC. inside and to the left	.CT. compared to

As you can see, a command language utilizing the above instruction set would prove invaluable to the TRS-80 user, whether a neophyte or an experienced programmer. It explains the more mysterious aspects of the machine and the complicated nature of its creators. If anyone has additional statements they would like to add, feel free, this language is not protected by copyright. I hope you'll enjoy PUT-N.■

Cheap and simple word processing.

BASIC Word Processor

Delmer D. Hinrichs
2116 S.E. 377th Ave.
Washougal WA 98671

You *do* always think of better phrasing after you see your thoughts in print, don't you?

A word processor for your computer lets you write letters and articles without typing a series of rough drafts. All of your typing revisions and editing are done on the video display. You produce a hard copy only after you are satisfied with the text.

You can also save the text on cassette so that you can make additional corrections and/or hard copies at a later time.

This BASIC word processor has the following features:

1. It accepts normal upper/lower case typing on an unmodified 16K Level II TRS-80.

2. It is line-oriented. Every line of text has a line number, for reference. This number need not be printed.

3. Editing is similar to Level II BASIC editing, except that the entire line is always visible under a transparent cursor.

4. You can change any of the ten variables to format for print or display.

5. Words can be automatically moved between lines to make all lines the correct format length.

6. Lines of text can be deleted, inserted, replaced, or moved.

7. Lines of text can be automatically right-justified.

8. Up to 120 lines of text, nearly 5 pages, double-spaced, can be in memory at one time.

Running the Program

When the program is loaded and RUN is entered, a title and prompt, COMMAND?, are displayed. The program will accept 17 single-letter legal commands. These are:

A ADD	Add text at the end of the text file.
B BLANK	Eliminate blank lines and renumber text file.
C COMPILE	Move words between lines in a specified block of lines to adjust lines to the correct length.
D DELETE	Delete a specified block of lines.
E EDIT	Edit a specified line.
F FORMAT	Change the format for text display or printing.
H HELP	List the 17 valid commands.
I INSERT	Insert new lines into the middle of the text.
J JUSTIFY	Right-justify the text file.
K KILL	Eliminate the current text file, and start over.
L LOAD	Load a text file from cassette tape.
M MOVE	Move specified lines to a new location in text.
P PRINT	Print the text file on the printer.
R REPLACE	Replace a specified line with a new line.
S SAVE	Record the text file on cassette tape.
V VIDEO	Display the text file on the video display unit.
X EXIT	Exit from the program

Now, let's examine these commands in detail.

ADD. The final lines of the current text (if any) are displayed, and a flashing block cursor shows where text will be added. This command automatically turns on the line number display option. Text can now be typed continuously without having to ENTER each line. When the text file is filled, a FILE FULL message is shown.

Subcommands in the ADD mode are:

← Moves the cursor left one position and erases the last character.
Shift-← Erases the entire current line.
→ Moves the cursor five positions to the right, adding five spaces.
Space bar moves the cursor one position to the right.
ENTER Ends the line, and goes to the next line before the automatic end-of-line action.
Shift- → Ends the line and moves the line text to the extreme right of the line.
↓ Ends the line and leaves an end-of-page marker, the down arrow, at the end of the line.
CLEAR Ends the line and leaves a do-not-justify marker, left arrow, at the end of the line.
Shift- ↓ Ends the line, centers the line text and leaves a do-not-justify marker at the end of the line.
Shift-@ twice Ends ADDing text and returns to command mode.
The first Shift-@ stops the program, and the second returns. Occasionally, only one Shift-@ is needed.

BLANK. All blank lines are eliminated from the text, and the line numbers are closed up. Note that blank lines are empty and this command does not effect lines that contain only spaces. During operation, this command displays DELETING BLANK LINES; when done, it automatically displays the new text file using the VIDEO command.

COMPILE. After editing or reformatting the text, some lines may be too long or too short to properly fit into the specified line length. This command shifts words between the lines of a specified block to get the best possible fit.

COMPILE works in two stages: It first checks for lines that are too long and pushes any extra words onto the following line. It then checks if any line can accept words from the following line and pulls any possible words back onto the preceding line. COMPILE should be used on only one paragraph at a time, as it left-justifies all lines except the first within

	Arrays
A$(120)	Lines of text
X$(3)	Buffer text lines for LOAD and SAVE
S(20)	Position to enter space for JUSTIFY
T(20)	Number of spaces to enter for JUSTIFY
	Variables
A$	Input entry character
B$	Blank to end of line (ASCII 30)
C$	Cursor (graphics character 143)
F$	Format string for line number ("### ")
H$	Heading for each page
L$	Left part of text string
N$	Line numbers? ("Y" or "N")
PN$	Page numbers? ("Y" or "N")
P1$	Print page no. for page no. 1? ("Y"/"N")
Q$	Number "n" string for EDIT subcommands
R$	Right part of text string
S$	Space character (" ")
X$	Temporary text manipulation string
X$(0)	Temporary text line string for EDIT
A	ASCII value of text character
B	Beginning display line
C	Cursor position
D	Cursor displacement for wrap-around lines, Temporary cursor position
F	First line for COMPILE, DELETE, or MOVE, Found character flag for EDIT search (0,1)
FL	First line of text for VIDEO or PRINT
FP	First page number to be listed
I	Multi-use Integer counter
IT	Insert text flag for INSERT (0 or 1)
J	Multi-use integer counter
K	Multi-use integer counter, L + 1
L	Line number of text
LA	Last line number of text
LL	Line length in characters
LM	Left margin to start PRINTing
M	Multi-use integer counter
N	Number "n" for EDIT subcommands, Number of spaces to skip for JUSTIFY, First new line number for MOVE
NL	Number of lines of text
P	Position in text line, PRINT line number
PL	Page length in number of lines
Q	Temporary position in text line
R	Return or REPLACE flag (0 or 1)
S	Spacing between text lines
U	ASCII value of space (32)
X	Length of A$(L) string, page number
Y	Length of A$(L + 1) string, Move cursor flag for EDIT (− 1 or + 1)
Z	Last line for COMPILE, DELETE, or MOVE

Table 1. Arrays and Variables

its range. It can also bury any end-of-page or do-not-justify markers that are not on the last line within its range. This should be avoided, as the markers are then ineffective. Extra spaces between moved words are eliminated, except at the end of sentences.

Following a period, question mark, exclamation point, or colon, three spaces are inserted. Trailing spaces are deleted, leaving only one space in a previously all-space line (to keep the BLANK command from eliminating the line).

Note that COMPILE can push extra words forward several lines, but can pull words back only one line.

During operation, this command displays COMPILING. If, after using COMPILE, the last line of the specified block is still too long, a LINE n HAS x CHARACTERS message is displayed. To correct this, INSERT an empty line and COMPILE just those two lines.

After a satisfactory COMPILE, the text file will be automatically displayed by the VIDEO command.

DELETE. This command eliminates a specified block of lines. If only one line is to be eliminated, enter that line number as both the first and last line number to DELETE. The text is displayed when done.

EDIT. The entire line—255 characters—is visible when in EDIT, including the character under the cursor. If a non-existent or empty line is specified, an ENTRY ERROR message is given. This command automatically turns on the line number display option.

Subcommands in the EDIT mode are:

n ← Moves the cursor to the left n positions. The default value of n is always one.
n → or n (space) Moves the cursor to the right n position without adding spaces.
A Again. Cancels previous editing changes and reenters EDIT mode. (List makes editing changes permanent.)
n C Change next n characters to next n entered characters. Cursor returns to start of changed block when done, as a signal that you have finished.
n D Delete the next n characters, and close up the line.
H Hack the rest of the line, and enter the INSERT mode.
I Insert characters into the line and move following characters to the right. While in the INSERT mode, you can move the cursor left or right without changes to the text by using ← or →.
L List the line and return cursor to the beginning. Also makes past editing changes immune to the Again subcommand.
n S c Search for the nth occurrance of character c. Keeps upper and lower case separate, so can be used to find out if a letter on the video display is correct case.
X Go to the end of the line and enter the INSERT mode.
Shift-↑ Exits the H, I, or X modes and returns to EDIT.
Shift-→ Moves current text to extreme right of the line.
Shift-↓ Centers current text, and adds a do-not-justify marker to the end of the line.
ENTER or Shift-@ Exits from the EDIT command.

After exiting from EDIT, if the line is too long, the LINE n HAS x CHARACTERS message is given. If the line is not too long, the text is displayed by the VIDEO command.

FORMAT. This command resets the ten text formatting parameters from their default values, either for the video display or for printing of the text.

1. Line length: Default value is 60 characters to fit on one video display line.
2. Line spaces: Default value is 0. Enter the number of blank lines that you want to appear between lines of text.
3. Line numbers: Default value is Y. To delete line numbers, enter N.
4. First line: Default value is 0 to show all lines of the text. To start the display or printing at a later line number, enter the desired beginning line number.
5. Left margin: Default value is 10 to print the default 60-character line centered on an 80-character printer. This setting affects only the printer.
6. Page length: Default value is 15. This is the number of lines to be printed on each page, so you might set it to 50 for printing unspaced text, or to 25 for spaced lines.
7. Page numbers: Default value is N, no page number. To print page numbers, enter "Y". Note: You must print page numbers if you want to print a page heading (see item 10 below).
8. First page: Default value is one to start page numbering with page number one. May be reset as required.
9. Page 1 No.: Default value is Y to print page numbers for all pages. To print page numbers for all pages *except* page No. one, enter "N".
10. Heading: Default value is one space. If a page heading is to be printed at the top of each page, enter it. For this entry only, Shift- letter for lowercase and letter for uppercase. If the heading is to have leading spaces (for centering) or punctuation, enclose it in quotes(""). The heading is printed or displayed only if the page numbering is on (items 7 and 9 above).

After going through these ten FORMAT parameters, you are returned to the command mode.

INSERT. This command inserts a line (or lines) of text into the middle of the current text. The following lines of text are moved down and renumbered. Specifying a non-existent line will give an ENTRY ERROR. To insert empty lines (for COMPILE or MOVE), just press ENTER. If the text file is filled, a "FILE FULL" message is given.

JUSTIFY. This command right-justifies all of the text, i.e., it makes the right ends of the lines even. The only exceptions are lines with a do-not-justify or an end-of-page marker at the end, or a line that has no spaces between words.

Extra spaces are inserted between words, starting randomly, but evenly distributed. Spaces may be inserted between adjacent words, or every other word, depending upon whether there are an even or odd number of words in the line. Trailing spaces are eliminated, leaving only one space for an all-space line. Leading spaces are not affected, so that indentation may be maintained.

It is suggested that JUSTIFY be used only after the text is in its final form, as the extra

Continue to next page

Program Listing 1

```
10 CLS:PRINTTAB(20)"BASIC WORD PROCESSOR"
20 ' (C) BY D.D.HINRICHS  1979
30 DEFINTA-Z:CLEAR7400:NL=120:DIMA$(NL),X$(3),S(20),T(20)
40 B$=CHR$(30):C$=CHR$(143):F$="### ":N$="Y":PN$="N":P1$="Y"
50 S$=" ":H$=S$:LA=-1:P=1:FP=1:PL=15:LL=60:LM=10:U=32:OUT254,1
60 L=LA:IT=0:R=0:A$="":PRINT:INPUT"COMMAND";A$:IFA$=""GOTO80
70 A=ASC(A$)-64:IFA>0ONAGOTO90,480,510,760,790,1220,80,1320,1350,
     1390,1510,1520,1580,80,80,1640,80,1750,1770,80,80,1830,80,19
     70
80 PRINT"** ENTRY ERROR **":GOTO60
90 CLS:D=0:N$="Y":IFLA<0THENL=0:GOTO130 ' ADD
100 IFNL=LA+1THEN210ELSEIFL>FL+12THENB=L-12ELSEB=FL
110 FORI=BTOL:X=LEN(A$(I)):D=D+INT((X+4)/64-.01)
120 GOSUB1910:NEXTI:L=L+1
130 C=(L-FL+D)*64:IFC>896PRINT:PRINT:C=896
140 PRINT@C,USINGF$;L;:PRINTA$(L);:P=LEN(A$(L))+1:C=C+P+3:K=L+1
150 PRINT@C,C$;:A$=INKEY$:PRINT@C,S$;:IFA$=""GOTO150
160 GOSUB290:ONA-7GOTO360,410,310
170 IFA=13THENA$=S$:GOTO210ELSEIFA=24THEN380ELSEIFA=31GOTO460
180 IFA=25THEN430ELSEIFA=26THEN330
190 IFA=96IFLA<LTHENLA=L:GOTO60ELSE60
200 PRINT@C,A$;:A$(L)=A$(L)+A$:IFP<=LLTHENP=P+1:C=C+1:GOTO150
210 IFRGOTO60ELSEIFNL<=KPRINT:PRINT"FILE FULL":LA=NL-1:GOTO60
220 IFLEN(A$(K))THENL=K:GOSUB1360
230 IFK>LATHENLA=K
240 IFA$=S$GOTO280
250 FORM=LL+1TO2STEP-1:A$=MID$(A$(L),M,1):IFA$<>S$NEXTM:GOTO280
260 A$(K)=RIGHT$(A$(L),LL-M+1):A$(L)=LEFT$(A$(L),M-1)
270 PRINT@C-LL+M-1,B$;:L=K:GOTO130
280 A$(L)=LEFT$(A$(L),LL):L=K:GOTO130
290 A=ASC(A$):IFA>64ANDA<91THENA=A+UELSEIFA>96ANDA<123THENA=A-U
300 A$=CHR$(A):RETURN
310 IFP>LLGOTO210 ' D
320 PRINT@C,CHR$(92);:A$(L)=A$(L)+CHR$(17):A$=S$:GOTO210
330 IFP>LLGOTO210 ' S-D
340 C=(L-FL+D)*64+4:IFC>900THENC=900
350 GOSUB1180:P=1:A$=S$:GOTO210
360 IFP=1GOTO150 ' L
370 C=C-1:PRINT@C,B$;:P=P-1:A$(L)=LEFT$(A$(L),P-1):GOTO150
380 IFP=1GOTO150 ' S-L
390 A$(L)="":P=1:C=(L-FL+D)*64+4:IFC>900THENC=900
400 PRINT@C,B$;:GOTO150
410 IFP>LL-6GOTO150 ' R
```

Continued on the next page

```
420 A$(L)=A$(L)+STRING$(5,S$):C=C+5:P=P+5:GOTO150
430 IFP>LLGOTO210 ' S-R
440 C=(L-FL+D)*64+4:IFC>900THENC=900
450 GOSUB1200:P=1:A$=S$:GOTO210
460 IFP>LLGOTO210 ' CL
470 PRINT@C,CHR$(93);:A$(L)=A$(L)+CHR$(20):A$=S$:GOTO210
480 CLS:PRINT"DELETING BLANK LINES":FORJ=LATO0STEP-1 ' BLANK
490 IFA$(J)=""FORI=JTOLA:A$(I)=A$(I+1):NEXTI:A$(LA)="":LA=LA-1
500 NEXTJ:IFRTHENRETURNELSE1830
510 INPUT"FIRST LINE TO COMPILE";F:IFF<0THENF=0 ' COMPILE
520 INPUT"LAST LINE TO COMPILE";Z:IFZ>LATHENZ=LA
530 IFF>=ZTHEN80ELSECLS:PRINT"COMPILING":FORL=FTOZ-1:K=L+1
540 X=LEN(A$(L)):X$="":IFX<2THEN620ELSEIFX<=LLGOTO600
550 FORI=XTO1STEP-1:A$=MID$(A$(L),I,1)
560 IFA$<>S$THENX$=A$+X$:NEXTI:GOTO600ELSEIFX$=""NEXTI
570 A=ASC(RIGHT$(X$,1)):IFA=33ORA=46ORA=58ORA=63THENX$=X$+"  "
580 A$(L)=LEFT$(A$(L),I-1):IFLEN(A$(K))=0THENA$(K)=X$:GOTO540
590 A$(K)=X$+S$+A$(K):GOTO540
600 X=LEN(A$(L)):IFX<2THEN620ELSEFORI=XTO2STEP-1
610 IFRIGHT$(A$(L),1)=S$THENA$(L)=LEFT$(A$(L),I-1):NEXTI
620 NEXTL:FORL=FTOZ-1:K=L+1
630 X=LEN(A$(L)):Y=LEN(A$(K)):X$="":IFX=0ORY=0GOTO750
640 A=ASC(RIGHT$(A$(L),1))
650 IFA=33ORA=46ORA=58ORA=63THENA$(L)=A$(L)+"  ":X=X+2
660 FORI=1TOY:A$=MID$(A$(K),I,1)
670 IFA$<>S$THENX$=X$+A$:NEXTIELSEIFX$=""NEXTI
680 IFLL-X<IGOTO710
690 Y=Y-I:IFY<0THENY=0
700 A$(L)=A$(L)+S$+X$:A$(K)=RIGHT$(A$(K),Y):GOTO630
710 X=LEN(A$(L)):IFX<2THEN730ELSEFORI=XTO2STEP-1
720 IFRIGHT$(A$(L),1)=S$THENA$(L)=LEFT$(A$(L),I-1):NEXTI
730 IFY<2THEN750ELSEFORI=YTO2STEP-1
740 IFLEFT$(A$(K),1)=S$THENA$(K)=RIGHT$(A$(K),I-1):NEXTI
750 NEXTL:X=LEN(A$(Z)):GOTO900
760 INPUT"FIRST LINE TO DELETE";F:IFF<0THENF=0 ' DELETE
770 INPUT"LAST LINE TO DELETE";Z:IFZ>LATHENZ=LA
780 IFF>ZTHEN80ELSEFORI=FTOZ:A$(I)="":NEXTI:GOTO1830
790 INPUT"EDIT LINE";L:IFL<0ORL>LAORA$(L)=""GOTO80 ' EDIT
800 C=4:P=1:X$(0)=A$(L):N$="Y"
810 CLS:I=L:GOSUB1910:N=1:Q$=""
820 GOSUB910:IFA>47ANDA<58THENQ$=Q$+A$:N=VAL(Q$):GOTO820
830 M=0:IFA=8THENY=-1:GOSUB940ELSEIFA=9ORA=UTHENY=1:GOSUB940
840 IFA=97THENA$(L)=X$(0):GOTO800 ' A
850 IFLEN(A$(L))>=LLGOTO870
860 IFA=25GOSUB1200ELSEIFA=26GOSUB1180
870 IFA>98ONA-98GOSUB960,1000,1960,1960,1960,1020,1030
880 IFA=115GOSUB1120ELSEIFA=12pGOSUB1170ELSEIFA=108GOTO800
890 IFM=1THENN=1:Q$="":GOTO820ELSEIFRPRINT@320,;ELSE810
900 IFLL<XPRINT"LINE";L;"HAS";X;"CHARACTERS":GOTO60ELSE1830
910 X$=MID$(A$(L),P,1)
920 PRINT@C,C$;:A$=INKEY$:PRINT@C,X$;:IFA$=""GOTO920
930 GOSUB290:X=LEN(A$(L)):IFA=13ORA=96THENR=1:RETURNELSERETURN
940 M=1:FORI=1TON:P=P+Y:IFP>XTHENP=X:RETURN
950 IFP<1THENP=1:RETURNELSEC=C+Y:NEXTI:RETURN
960 Q=P:D=C:FORI=1TON:GOSUB910:IFRORA=27THENP=Q:C=D:RETURN ' C
970 PRINT@C,A$;:GOSUB1100:P=P+1:GOSUB1110:A$(L)=L$+A$+R$
980 A=U:C=C+1:IFP<=XNEXTI
990 P=Q:C=D:RETURN
1000 IFP+N-1>XTHENN=X-P+1 ' D
1010 GOSUB1100:Q=P:P=P+N:GOSUB1110:A$(L)=L$+R$:P=Q:RETURN
1020 GOSUB1100:A$(L)=L$+S$:PRINT@C,B$ ' H
1030 GOSUB910:IFRORA=27RETURN ' I
1040 IFA=10THENA$(L)=A$(L)+CHR$(17):R=1:RETURN
1050 IFA=31THENA$(L)=A$(L)+CHR$(20):R=1:RETURN
1060 PRINT@C,A$;:IFA=8THENY=-1:GOSUB940:GOTO1030
1070 IFA=9THENY=1:GOSUB940:GOTO1030ELSEIFP>XTHENX=P
1080 GOSUB1100:GOSUB1110:A$(L)=L$+A$+R$:PRINT@C,B$;A$+R$
1090 C=C+1:P=P+1:GOTO1030
1100 L$="":IFP<2RETURNELSEL$=LEFT$(A$(L),P-1):RETURN
1110 R$="":IFP>XRETURNELSER$=RIGHT$(A$(L),X-P+1):RETURN
1120 GOSUB910:Q=P:D=C ' S
1130 FORI=1TON:F=0:FORJ=Q+1TOX:D=D+1
1140 IFMID$(A$(L),J,1)=A$THENF=1:Q=J:J=X
1150 NEXTJ:NEXTI:IFFTHENP=Q:C=D
1160 A=U:RETURN
1170 A$(L)=A$(L)+S$:P=X+1:C=P+3:GOTO1030 ' X
1180 A$(L)=STRING$((LL-LEN(A$(L)))/2,32)+A$(L)+CHR$(20) ' S-D
1190 PRINT@C,B$;A$(L);CHR$(93);:RETURN
1200 A$(L)=STRING$(LL-LEN(A$(L)),32)+A$(L) ' S-R
1210 PRINT@C,B$;A$(L);:RETURN
1220 CLS:PRINT"LINE LENGTH =";LL,:INPUT"NEW =";LL ' FORMAT
```

Continued on page 54

spaces may be incorrectly left in by COMPILE (after EDITing). During its operation, this command displays JUSTIFYING; when done, the text is displayed by VIDEO.

KILL. This command eliminates all text from the text file, leaving the program ready to accept new text. The command asks twice if you really want to KILL the text to avoid accidental loss of a text file.

LOAD. This command loads a previously SAVEd text file from cassette. It can then be treated just like a keyed-in text file. After you enter this command, the program pauses to let you get the cassette in position and set the recorder to play. Since it may take several minutes to load the text file, the program displays LOADING to reassure you that the program has not hung up.

MOVE. This command transfers a specified block of lines either forward or backward in the text file. Lines MOVEd *to* must be empty (blank); lines MOVEd *from* are left empty. Insert may be used to place empty lines where needed.

If a non-empty TO line is found, the transfer of lines stops, and a "LINE n NOT EMPTY" message is displayed. However, no text is lost. After the MOVE is complete, the text is displayed by VIDEO.

PRINT. This command prints the text file. Remember to reset FORMAT before trying to PRINT your text. If the printer is not ready to go, PRINTER NOT READY. ABORT (Y/N) is displayed, and you have a chance to get the printer ready or to go back to the command mode.

The BLANK command is automatically executed before PRINTing, as a blank line would cause a FUNCTION CALL error. During its operation, first DELETING BLANK LINES, the PRINTING is displayed. If the printer encounters an end-of-page character (ASCII 17) that page will be terminated early, the normal between-pages spacing inserted and the next full page started. Note: It is assumed that roll paper is being used; no form feeds are used.

Different printers operate somewhat differently, so you may have to change some things in lines 1680-1740. Specifically, my printer, a COMPRINT 912, interprets the ASCII control character 30 in line 1680 as continuous print (no pagination). The corresponding control character 28 in line 1740 means paginate (insert seven blank lines).

Some printers may require line feed characters (usually ASCII 10) after each printed line.

Some printers will not accept the LPRINT at the end of line 1690, and require LPRINT-CHR$(138) (or LPRINT" ") instead. If the 138 pseudo-control character in line 1710 does not work for you, use:

```
1710 M = M + 1:IFSFORK = 1TOS:LPRINT" ":NEXTK
```

Continued on page 54

A little revision should allow any printer to be used.

REPLACE. This command allows any specified line to be replaced with a newly entered line. REPLACE operates just like ADD, except that it applies to only one line. After a FILE FULL message, you cannot REPLACE a line.

SAVE. This command records the text file on cassette. The program pauses to let you get the cassette into position. It displays SAVING during operation. You can record four lines of text per data block to save time and tape. If the line length is greater than 61 characters, only three lines of text can be recorded per block. If it is greater than 82 characters, only two lines of text can be recorded per block. If it is greater than 123 characters, only one line of text can be recorded per block.

To make the changes (if necessary), change the four at the end of line 1780 of the program, the three in line 1790, and reduce the number of X$()'s in line 1810 to the number of lines per block being recorded.

To LOAD the text recorded with the revised SAVE routine, corresponding changes must be made to lines 1540 and 1550 of the program. If text lines are more than 60 characters long, NL should be redefined in program line 30 to approximately 7200/(characters per line). The exact value should be a number divisible by lines per block. NL = 120, for example 120 is evenly divisible by four.

VIDEO. This command displays the text file on the video display. If the line length is greater than 60 characters (64 if Line Nos. was set N in FORMAT), the lines will "wrap around" to the next display line.

After each displayed page of text, the program halts showing PRESS ENTER?. To display the next page just press ENTER; to return to the command mode (to EDIT a line, etc.) key in any letter, then press ENTER.

VIDEO may show one of three non-text markers at the end of a line: A left arrow for do-not-justify, a down arrow for end-of-page or an underline after any trailing space. These markers allow you to keep better track of the text.

EXIT. This command allows for a graceful end to the program. More importantly, it CLEARs the string space to its normal value, so that the next program you run does not crash. It is easy to forget to CLEAR 50. It also returns the TRS-80 to its normal speed (OUT254,0).

This command asks again if you really want to EXIT from the program to avoid accidental loss of the text file.

HELP. This command displays all 17 legal commands and their one-word definitions, to refresh your memory. It also tells how to return to the command mode, for those commands that do not automatically return.

```
1230 PRINT"LINE SPACES =";S,:INPUT"NEW =";S
1240 PRINT"LINE NOS. =  '";N$;"'",:INPUT"NEW (Y/N)";N$
1250 PRINT"FIRST LINE = ";FL,:INPUT"NEW =";FL
1260 PRINT"LEFT MARGIN =";LM,:INPUT"NEW =";LM
1270 PRINT"PAGE LENGTH =";PL,:INPUT"NEW =";PL
1280 PRINT"PAGE NOS. =  '";PN$;"'",:INPUT"NEW (Y/N)";PN$
1290 PRINT"FIRST PAGE = ";FP,:INPUT"NEW =";FP
1300 PRINT"PAGE 1 NO. = '";P1$;"'",:INPUT"NEW (Y/N)";P1$
1310 PRINT"HEADING = '";H$;"'   ",:INPUT"NEW =";H$:GOTO60
1320 CLS:PRINT"LEGAL COMMANDS ARE:" ' HELP
1330 PRINT"A  ADD","B  BLANK","C  COMPILE","D  DELETE","E  EDIT",
     "F  FORMAT","H  HELP","I  INSERT","J  JUSTIFY","K  KILL","L
      LOAD","M  MOVE","P  PRINT","R  REPLACE","S  SAVE","V  VIDEO
     ","X  EXIT"
1340 PRINT"KEY 'SHIFT-@' TWICE TO RETURN FROM A,E,I,R TO COMMAND
     MODE":GOTO60
1350 INPUT"INSERT AT LINE";L:IFL<0ORL>LAGOTO80 ' INSERT
1360 IFNL=LA+1PRINT"FILE FULL":GOTO60ELSEIFRGOTO60
1370 FORI=LATOLSTEP-1:A$(I+1)=A$(I):NEXTI
1380 A$(L)="":LA=LA+1:L=L-1:IFITRETURNELSEIT=1:GOTO90
1390 CLS:PRINT"JUSTIFYING":FORL=0TOLA:X=LEN(A$(L)) 'JUSTIFY
1400 IFX<2GOTO1500ELSEFORI=XTO2STEP-1:A=ASC(RIGHT$(A$(L),1))
1410 IFA=UTHENA$(L)=LEFT$(A$(L),I-1):X=X-1:NEXTI
1420 IFX>=LLORA=17ORA=20THEN1500ELSEJ=0:K=1:FORI=1TOX
1430 IFMID$(A$(L),I,1)<>S$THENK=0ELSEIFK=0THENK=1:S(J)=I:J=J+1
1440 NEXTI:IFJ=0GOTO1500
1450 K=RND(J)-1:IFINT(J/2)=J/2ORJ=1THENN=1ELSEN=2
1460 FORI=1TOLL-X:T(K)=T(K)+1:K=K+N:IFK>J-1THENK=K-J
1470 NEXTI:FORI=J-1TO0STEP-1:A$=STRING$(T(I),S$):T(I)=0
1480 A$(L)=LEFT$(A$(L),S(I))+A$+RIGHT$(A$(L),LEN(A$(L))-S(I))
1490 NEXTI
1500 NEXTL:GOTO1830
1510 CLS:INPUT"REALLY KILL (Y/N)";A$:IFA$="Y"THENRUNELSE60'KILL
1520 GOSUB1820:CLS:PRINT"LOADING" ' LOAD
1530 INPUT#-1,LA,LL,S,N$,FL,LM,PL,PN$,FP,P1$,H$
1540 FORI=0TOLASTEP4:INPUT#-1,X$(0),X$(1),X$(2),X$(3)
1550 FORJ=0TO3:L=I+J:X=LEN(X$(J)):A$(L)="":IFX<1GOTO1570
1560 FORK=1TOX:A$(L)=A$(L)+CHR$(ASC(MID$(X$(J),K,1))-128):NEXTK
1570 NEXTJ:NEXTI:GOTO60
1580 INPUT"FIRST LINE TO MOVE";F:IFF<0THENF=0 ' MOVE
1590 INPUT"LAST LINE TO MOVE";Z:IFZ>LATHENZ=LA
1600 IFF>ZTHEN80ELSEINPUT"FIRST NEW LINE";N:FORI=FTOZ
1610 IFLEN(A$(N))PRINT"LINE";N;"NOT EMPTY":GOTO60
1620 A$(N)=A$(I):A$(I)="":N=N+1:IFN>LATHENLA=N
1630 NEXTI:GOTO1830
1640 IFPEEK(14312)<128THENX=FP:M=FL:GOTO1670 ' PRINT
1650 INPUT"PRINTER NOT READY.   ABORT (Y/N)";A$
1660 IFA$="Y"THEN60ELSEINPUT"PRESS ENTER";A$:GOTO1640
1670 R=1:GOSUB480:CLS:PRINT"PRINTING"
1680 LPRINTCHR$(30):IFPN$<>"Y"OR(P1$="N"ANDX=1)GOTO1700
1690 LPRINTTAB(LM)H$;TAB(LL+LM-7)"Page";USING"###";X:LPRINT
1700 FORP=MTOM+PL-1:IFP>LAGOTO1740
1710 M=M+1:IFSLPRINTSTRING$(S,138)
1720 LPRINTTAB(LM);:IFN$="Y"LPRINTUSINGF$;P;
1730 LPRINTA$(P):IFASC(RIGHT$(A$(P),1))<>17NEXTP
1740 LPRINTCHR$(28):IFP>LATHEN60ELSEX=X+1:GOTO1680
1750 INPUT"REPLACE LINE";L:IFL<0ORL>LAGOTO80 ' REPLACE
1760 R=1:A$(L)="":L=L-1:GOTO90
1770 GOSUB1820:CLS:PRINT"SAVING" ' SAVE
1780 PRINT#-1,LA,LL,S,N$,FL,LM,PL,PN$,FP,P1$,H$:FORL=0TOLASTEP4
1790 FORJ=0TO3:I=L+J:X=LEN(A$(I)):X$(J)="":IFX<1GOTO1810
1800 FORK=1TOX:X$(J)=X$(J)+CHR$(ASC(MID$(A$(I),K,1))+128):NEXTK
1810 NEXTJ:PRINT#-1,X$(0),X$(1),X$(2),X$(3):NEXTL:GOTO60
1820 INPUT"READY CASSETTE, THEN PRESS ENTER";A$:RETURN
1830 CLS:X=FP-1:FORM=FLTOLASTEPPL:X=X+1 ' VIDEO
1840 IFP1$="N"ANDX=1GOTO1860
1850 IFPN$="Y"PRINTH$;TAB(LL-7)"Page";USING"###";X:PRINT
1860 FORI=MTOM+PL-1:IFI>LAGOTO1890
1870 IFSPRINTSTRING$(S-1,10)
1880 GOSUB1910
1890 NEXTI:A$="":IFI<=LAINPUT"PRESS ENTER";A$:IFA$<>""M=LA
1900 NEXTM:L=LA:GOTO60
1910 Y=LEN(A$(I)):IFYTHENA=ASC(RIGHT$(A$(I),1))ELSEA=0
1920 IFN$="Y"PRINTUSINGF$;I;
1930 PRINTA$(I);:IFA=17PRINTCHR$(92);
1940 IFA=20PRINTCHR$(93);ELSEIFA=UPRINTCHR$(95);
1950 IFN$<>"Y"ORY<>60PRINT
1960 RETURN
1970 CLS:INPUT"REALLY EXIT (Y/N)";A$:IFA$<>"Y"GOTO60 ' EXIT
1980 CLS:CLEAR50:OUT254,0:END
```

Potential Program Problems

Speed is the most noticeable problem, but this is inherent in BASIC strings. The lack of speed shows up as brief pauses in program operation. As the text file becomes full, the pauses become more frequent and longer. Each time BASIC manipulates a string, it must assign a new location for it in string space. This quickly fills up string space, so a garbage collection routine is used to delete all of the old, no-longer-needed versions.

This periodic garbage collection causes the pauses. Some other BASICs avoid this string-handling problem by requiring that the length of all strings be specified in advance, thus trading off flexibility and memory space for speed.

One partial solution to the speed problem is to install the Archbold TRS-80 clock control board. The OUT254,1 at the end of line 50 automatically increases speed by 50 percent if this has been done. Otherwise, it has no effect.

A program halt accompanied by a BASIC error message or from accidentally touching BREAK does not necessarily mean that your text file is lost. In most cases you can recover by typing GOTO 60, then pressing ENTER. This puts you back into the command mode.

A trailing space in the 60th position on a line will cause a wrap-around underline marker on the next line when displayed. Such a trailing space can be removed by using EDIT, COMPILE, or JUSTIFY.

Under some conditions, an extra blank line may be inserted in the video display when text is being keyed in. This is not an extra line in the actual text file, as can be verified by using VIDEO.

When using EDIT, if you delete all printable characters following the cursor, it stops flashing and becomes solid. This does no harm; press L to get back to normal operation.

If a text line has more than 22 words and requires over 21 spaces to justify, an error will occur in JUSTIFY. To avoid this, redimension arrays S and T in line 30, but not too much, or you will run out of memory.

Program Modifications

This program requires 7226 bytes of memory as loaded, but uses nearly all the memory of a 16K TRS-80 when RUN.

For a 32K TRS-80, change CLEAR7400 to CLEAR22000 and change NL = 120 to NL = 360 in line 30. This not only triples the size of the text file, but also increases the program's speed.

This program uses the ASCII control characters 17 and 20 as markers representing end-of-page and do-not-justify, respectively. If these characters are not NOPs (no operation) on your printer, change 17 in program lines 320, 1040, 1420, 1730, and 1930 and/or change 20 in program lines 470, 1050, 1180, 1420, and 1940 to values that do not affect your printer.

You may want to add extra features to this program. Be warned: There are less than 100 bytes of memory left unused. For everything added, something will have to be deleted. I have already minimized REMarks, used multiple statement lines, reused variables, and left out all spaces to save memory, as well as to increase speed.

To help with possible modification, I have included Table 1 Arrays and Variables. A brief REMark shows the entry point for each command. Even more brief REMarks show the entry points for subcommands in ADD and EDIT. ■

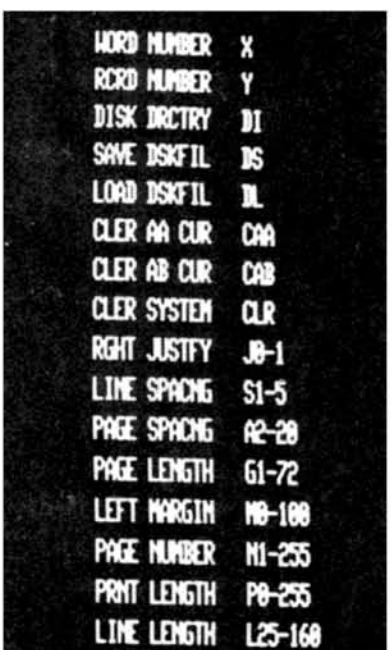
The de facto standard is challenged by Radio Shack.

Pencil VS. Scripsit

by Jim Perry
***80* Staff**

The Electric Pencil from Michael Shrayer Software and Radio Shack's Scripsit are both sophisticated word processors for the TRS-80. Scripsit is the "official" offering from Tandy Corporation, whereas Pencil has been the de facto standard for more than a year. Each program, written in assembly language, can be used with either cassette or disk systems.

Word processing is one of the microcomputer's most useful applications. Using a word processor documents can be edited, reformatted or printed out with a minimum of effort. Instead of the usual "Dear Sir or Madam," form letters can be customized with the name of the intended reader. By storing a selection of paragraphs, standard responses to a number of inquiries can be produced in a minimum of time.



Electric Pencil menu.

Its list of applications is endless. Once you have the opportunity to use such a system, you'll wonder how you ever managed without one!

Upper and Lowercase

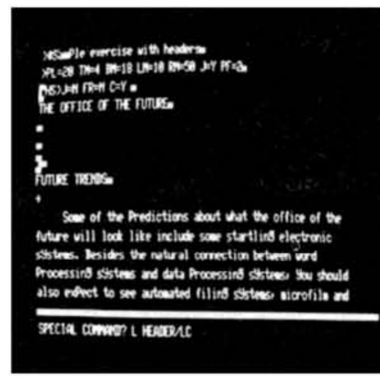
Any word processor without the ability to produce both upper and lowercase letters has limited usefulness. Because of penny-pinching at Radio Shack, the TRS-80 was produced without the five-dollar memory chip needed to display lowercase letters. Since that time, various companies have made available reasonably priced (less than $30) hardware conversion kits for the computer, but all involve opening up your keyboard and voiding Radio Shack's warranty. The most popular conversion is designed to interface with the Electric Pencil and includes an extra control key for special commands.

When Radio Shack announced Scripsit, they also released the official lowercase modification—at a cost of $99. Either Pencil or Scripsit can be used as uppercase systems without any modification to your computer.

Radio Shack's conversion has no extra key. The @ key performs as the control key (to produce @, the shift 0 is used). Luckily, for the thousands of owners with non-Radio Shack modifications, Scripsit will work with the normal Electric Pencil lowercase modification. However, anyone who has Radio Shack install a lowercase kit is stuck with Scripsit. The Electric Pencil will not produce lowercase with this official modification.

Scripsit costs $69 for the cassette version and $99 on disk. The Electric Pencil retails for $100 for the cassette only version. A cassette for disk based systems costs $150.

This review is based on the disk versions of both programs. In both cases all the cassette commands are still available.



Typical Scripsit display.

If you start off with the cassette version of either program (because you don't have disks), you can copy all your files onto disk at a later date. As supplied, neither will operate with NEWDOS, but patches are available from Apparat that allow both to work with this DOS.

Manuals

The Electric Pencil comes with a 32-page operator's manual, written as a guide rather than a textbook, and the technique works well. Even though there are more than 40 commands available, the average user can develop a good understanding of the system in an afternoon of reading and experimenting.

The Scripsit disk is supplied with sample text files, and two program versions—one for uppercase only and the other for systems with a lowercase modification.

The documentation is much more impressive than that supplied with Electric Pencil, complete with ring binder and a three-cassette audio instruction course. However, the taped instruction course needs about six hours to listen to, and the manual is written as an accessory to the audio. As a result of this programmed learning

Scripsit is cheaper and more flexible, so how can the Pencil compete?

approach, the manual is not really usable on its own. The quick reference sheet included in the package is the nearest you get to a simple user guide.

The Electric Pencil has a menu of print parameters, such as line length and page length, and you set them all before printing. The parameters are not stored with the text file, and this means that you have to remember what they were.

Scripsit differs fundamentally from the Electric Pencil in this respect. Scripsit uses format lines that are saved as a part of the text. Every time a file is used, it automatically sets its own printing parameters. This difference in operation gives Scripsit far more flexibility than the Electric Pencil. Also, the number of commands available in Scripsit is greater than the 40 or so available to the Pencil user, but this also makes it much more difficult to understand and use.

Surprisingly, three of the most useful features of Electric Pencil are not available to the Scripsit user! The suffix /PCL is automatically added to all files created by Pencil. This makes it easy to spot them on your disk directory. Pencil only recognizes files with the /PCL suffix. Scripsit does not add a suffix to files, leaving it to the user to supply them.

The command DI(:1,2 or 3) in Pencil gives a directory of all files with the /PCL suffix on the drive specified. (The default is drive 0.) In Scripsit you have to exit the system and use the DIR command when in DOS, then return to Scripsit, quite a messy procedure; and, in fact, impossible, if you make the mistake of turning Scripsit into an AUTO command on the disk, as I did!

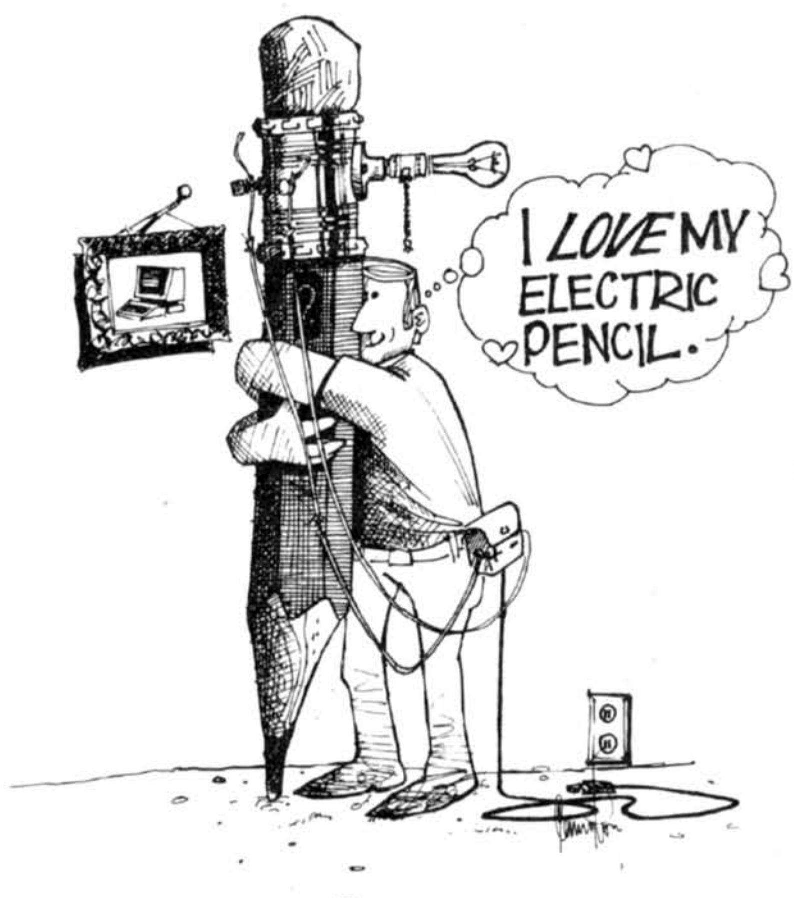
The third feature missing from Scripsit is the ability to kill files, again you are expected to return to DOS etc.

Conclusion

Apart from the three omissions described above Scripsit has a much larger set of formatting commands than Pencil—tabs, centering, flush right, alternate page headings and footnotes to name just a few. By careful use of these commands your printouts can be made to do amazing things, but the format instructions can drive you crazy. Maybe the six hour course wasn't such a loopy idea after all!

Scripsit is cheaper and more flexible, so how can the Pencil compete? Simplicity is the answer, I tend to create files in Pencil and print them out with Scripsit—the *80* office system has the unofficial modification, so both programs can be used.

For real fancy printing Scripsit is a must, but for ease of operation, Pencil can't be beat. Maybe somebody will come up with a Pencil compatible with the Radio Shack modification, or, perhaps, produce software that is as nice to use as Pencil—but as advanced as Scripsit. Until that time, Scripsit is the new champion of word processing. ■



Psst, want some extra commands—try Level III.

Level III

Ronald H. Bobo
3246 Gravois
St Louis MO 63118

Level III was written by Bill Gates, president of Microsoft, which produced Level II BASIC for Radio Shack, so there is no question of compatibility. Furnished on cassette, Level III is attractively packaged and is accompanied by an excellent instruction manual as well as a convenient fold-up card containing all statements, commands and functions of both Levels II and III. Level III statements are printed in red on the card.

As stated in the instruction manual, Level III is not just another BASIC, but an actual enhancement to the already versatile Level II BASIC. Requirements for using Level III are a Radio Shack TRS-80 computer with Level II BASIC and a minimum of 16K random access memory. According to the manual, Level III occupies about 5.25K of existing RAM. This still leaves 10K for programming space in a minimum system.

Although Level III will not work with Radio Shack's disk BASIC, a disk file version is included on the tape for those who might wish to save it on disk. Presumably, Level III can be used instead of Disk BASIC with a disk system.

I can...testify that in copying a couple of lengthy game programs using just three commands ...saved considerable typing.

Level III Features

As soon as Level III is loaded, you should notice one big difference. No more keyboard bounce! More reliable tape loading is also assured. Volume setting on the tape recorder is not nearly so critical as with Level II. Two new commands, SAVE and LOAD replace CSAVE and CLOAD. Of course, a program saved with the new command will still load under the CLOAD command when using Level II, but volume setting will again be critical. Although all recorders are not the same, most programs in Level II load at a volume setting between 4 and 5 on my machine. Curiously, I must crank it up to about 8 before the Level III cassette will load.

For those who miss the one-letter abbreviations of Level I BASIC, behold! Type LSET LIST, hit ENTER, and 26 one-letter abbreviations will appear on the CRT. These are activated by pressing the shift key and the desired letter key simultaneously. Some even have ENTER included, i.e., press shift and R and RUN is printed out and automatically entered; you don't have to press the ENTER key.

If you have data that is to be entered in a program frequently and it is not included in the list of abbreviations, despair not. All LSET entries are user-changeable. For instance, in writing a graphics program, you may want to use CHR$ often. Type LSET L = "CHR$(", enter, and now, when you type a shifted L, CHR$ will be printed, followed by a left parenthesis. Typing LSET L = "LIST" will return the shifted L to its original definition

What about using a printer with lowercase letters? Nothing simpler. Typing LSET RESET will turn off all shift-key entries. Type LSET SET to turn them back on again.

Shift-key entries need not be commands. If you're running a program that requires certain phrases to be input repeatedly, these may be set up as shift-key entries. The string expression in an LSET may be any string up to 15 characters.

I can personally testify that in copying a couple of lengthy game programs from a book, using just three of the LSET commands (T for THEN, F for GO TO and G for GOSUB) saved considerable typing.

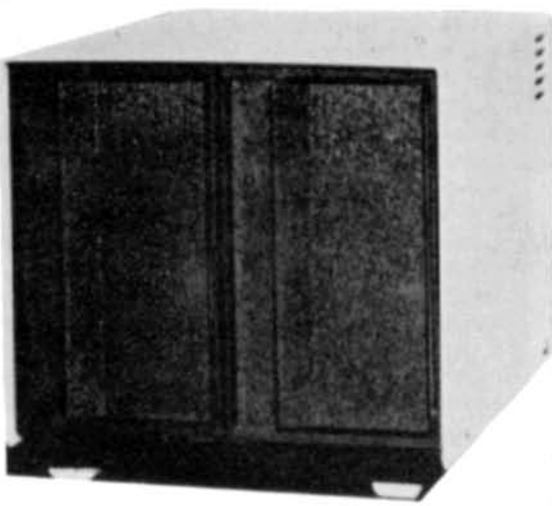
Another special feature of Level III is Line Renumbering. This is a flexible command, and all of its possibilities will not be dealt with here. Suffice it to say that typing NAME will renumber all program lines, starting with 10 as the first line and incrementing by 10. Other possibilities allow you to specify that renumbering begin at a specific line, specify new beginning line number and specify the increment, as well as various combinations of these.

Now we can make all the inevitable changes and insertions in our programs and still end up with an attractive looking format. All GOTOs, GOSUBs and

NEECO (THE PET PEOPLE) ANNOUNCES

400K DISK STORAGE (Before DOS)

FOR YOUR TRS-80*
(in a single housing)



$1295.00

- ★ **400K Raw Storage**
- ★ **Plug Compatible to TRS-80***
- ★ **Uses TRS-80* DOS**
- ★ **Available from Stock**

• Painted TRS-80* grey • Shipping Wt. 21 lbs.

DEALER and OEM INQUIRIES INVITED

NEECO
143

(617) 449-1760

679 Highland Ave.
Needham, Massachusetts 02194
TELEX 951021

Mon-Fri 9:30-5:30
MasterCharge &
Visa Accepted

***TRS-80 is a trademark of Tandy Corp.**

other references to line numbers within program statements are, of course, changed to agree with the new numbering.

When you get an error message, you'll no longer have to grab for the Level II manual to interpret that pesky two-letter code. Level III spells out all error messages. For those who own the Radio Shack expansion interface but no disk drive, Level III allows access to the integral clock-calendar.

Under the Radio Shack Disk Operating System, DATE sets the date and TIME sets the time of day, which may then be accessed by the CLOCK command under DOS or TIME$ under disk BASIC.

Level III simplifies this procedure considerably. Type CMD "R" and you can then set date and time all in one operation. PRINT TIME$ will then display date and time in a 24-hour format. TIME$ may be used in conjunction with RIGHT$,LEFT$ or MID$ to access one or any combination of date, hours, minutes and seconds.

Although BASIC programs require decimal numbers, Level III will do the converting for you. Preface a hexadecimal number with &H or an octal number with & and conversion will be automatic. Caution: This will not work in DATA statements or in response to INPUT statements.

Probably all of us have experienced a system lockout at one time or another. Control of the TRS-80 is lost and, no matter which key you hit, nothing happens. You can recover from this state by pressing the reset button, but if you're using the expansion interface, hitting reset results in a complete loss of memory. Level III allows you to use the BREAK key instead of reset. Using BREAK will recover control and memory will not be lost, even with the expansion interface attached.

Graphics

Level III graphic capabilities are considerably advanced over those of Level II. Level II requires that, when drawing a line or rectangle, you define all the points yourself. In Level III you need only define two points on the screen and a line or rectangle will be drawn between them. Level III also allows retrieval of a graphics array for storage and later use in the same or a different screen area. This greatly simplifies the writing of programs using graphs, pictures or animation.

Two graphics modes, designated Character Mode and Graphics Mode, are used in Level III. In Character Mode, the screen is divided into a grid composed of 64 character positions across by 16 down. Each position is further divided into a two-by-three graphics block. In Character Mode, any alphanumeric character or any of the 63 special graphics characters may be placed at any specified character position using X-coordinates (0-63) and Y-coordinates (0-15). Graphics characters are printed by specifying the appropriate CHR$ (129 to 191).

In Graphics Mode a finer grid, measuring 128 across by 48 down, is used. The Level II functions SET, RESET and POINT are used here.

The new Level III graphic statements LINE, GET@ and PUT@ may be used in either Character or Graphics mode.

LINE is used to draw a line between any two points on the screen. It will also draw a rectangle, providing the two specified points are diagonally opposite corners. In the Character Mode, the line or rectangle is composed of the first character of a string expression that follows the specified coordinates. LINE, followed by two sets of coordinates and a string, results in a straight line of Xs, or whatever the first character may be. Adding a "comma B" after the string will cause a rectangle of characters to be printed. For example, 10 LINE (32, 14) (45,10), "X", B will print a rectangle composed of Xs near the bottom of the screen and slightly to the right of center. Using BF instead of B alone will fill in the rectangle with Xs.

In the Graphics Mode, SET is used instead of the string to turn on graphics blocks as in Level I or II. Again B or BF may be used to outline a rectangle or print a solid white block, respectively. With proper use of RESET, black graphics may be printed on a white background.

Let me add that these graphics are fast! Typing 10 LINE (0,0)-(127,47), SET, B and RUNing will result in a white border around your entire screen instantly! No more waiting for a slow line to be drawn.

GET@ and PUT@ are versatile graphics statements. GET@ allows you to save in an array all graphics that are within the defined area of the screen. Once saved, the graphics may be replaced at any area of the screen using PUT@ followed by the desired coordinates.

There are five options when PUT@ is used. Using SET in the PUT@ statement returns the array to the CRT exactly as it was saved. RESET reverses everything. All "on" blocks are turned off and all "off" blocks are turned on. Using AND will result in a block being turned on only if it is "on" both on the screen and in the array. OR causes a position to be turned "on" if it is "on" the screen or in the array or both. Using XOR, a position will be turned "on" only if its status in the array is opposite that on the screen.

Graphics arrays must be properly dimensioned in order to avoid an error message. A one-line array requires 64 bytes, while the entire screen will be 1024. A formula is given in the manual for determining what the needed array space should be.

Other Features

The following four features, which appear to be identical to those in Radio Shack disk BASIC, will be discussed briefly.

MID$ in Level III may be used on the left as well as the right side of an assignment statement. This will allow you to replace part of one string with another.

INSTR searches one string for another string. A subroutine for this was given in the Level II manual. This need no longer be used.

DEF FN, a sad ommission from Level II, allows user-defined functions. String as well as numeric functions may be defined.

Up to ten different machine-language routines may now be stored simultaneously in memory and accessed with the USR function. DEFUSR informs BASIC of the starting address of a machine-language routine. It is no longer necessary to POKE the starting address, although still possible.

The LINE INPUT statement appears to differ from that in disk BASIC. According to a listing of disk BASIC, version 2.1 in one of the Radio Shack brochures, LINE INPUT is used to input a line of data from disk into a string variable.

In level III, LINE INPUT is similar to INPUT with added features. LINE INPUT assigns a string variable name to an entire line of input, including commas or other punctuation. Everything typed before ENTER will be part of the string.

Adding #LEN to either the INPUT or LINE INPUT statements will set a limit on the amount of time allowed for response to the INPUT. 10 INPUT #LEN 4,100;Q will result in the program branching to line 100 if Q is not entered within four seconds. All timing is done in software, so the expansion interface is not necessary.

Speaking of the expansion interface, the system clock is prone to bollixing up tape operations. The clock should always be turned off before a SYSTEM, PRINT# – 1 or INPUT# – 1 operation. Entering CMD "T" will turn the clock off, and CMD "R" turns it back on. LOAD and SAVE do their own turning on and off automatically.

PRINT # – 3 allows output to a printer or other peripheral that may be attached to your RS-232 port. Input from the port will still require a machine-language routine.

Level III BASIC appears to be a worthwhile acquisition for any serious TRS-80 user, especially one who would like the power of disk BASIC without the added expense of a disk drive. Add in all the extras not contained in disk BASIC, and it becomes desirable even for a disk owner. ■

With the price of gas it makes sense to share a ride to and from work.

Carpool

Walter K. McCahan
PO Box 3314
Shiremanstown PA 17011

While driving to work and contemplating nontrivial uses for my TRS-80 microcomputer, I noticed that almost every car on the road was occupied by only one person. Keeping in the mind the high cost of gasoline and other automobile operating expenses, I felt that a bit of research was in order.

After a hasty and short survey of a local office building, I found that the overwhelming reason that car pools had not been formed was the employees' lack of knowledge as to which other employees lived close to them and were available to form a pool. I also found that the employer was willing to pay a healthy sum to provide their employees with the information with which to form car pools. To attack and solve this problem I devised the following plan of action:

1. Find the home address of each employee.
2. Reduce these addresses into zones.
3. Write a program to match employees that live close to one another.
4. Give each employee a list of these other nearby employees and instruct them to form car pools.

Step 1.

Gathering personal information from the employees in this building was not a difficult task and was accomplished by the letter shown in Fig. 1. The letter was copied on the company's copy machine and was distributed with each employee's payroll check.

I found that since there was a direct savings to them, the employees were quick to complete the information and return the questionnaire.

Step 2.

After looking into several methods of relating addresses to locations, including the name

Fellow Employees,

In these times it is no longer economical for each employee to drive his or her car to work alone.

In an effort to conserve gasoline (and therefore money) your company is providing a computer service that will match your name with other employees that work with you in order that you can form a carpool.

Attached you will find a map of our local area which has been divided into 16 zones. You will also find below a form that is to be clipped and returned to the personnel department after it is completed.

All items on the form are self-explanatory except for the one labeled "Zone." In this blank you should enter the number of the zone in which you live according to the numbers on the map. If you do not live within one of the zones, enter the zone nearest to your home.

After all of the forms have been returned and entered into the computer you will be furnished with a list of names and instructions on how to set-up your carpool.

Sincerely,

The Management

CARPOOL QUESTIONNAIRE

NAME ________________
STREET ADDRESS ________________
CITY & ZIP ________________
TELEPHONE NO. ________________
ZONE ________________
SHIFT ________________
BUILDING ________________

Fig. 1. Gathering employee information.

of the town and zip code, it was decided that a completely independent method of zoning must be developed.

Using a map of the local area (Harrisburg PA), I produced a grid of 16 sections (see Fig. 2). Each zone on this grid represents 225 square miles, so that the entire grid represents the 3600 square miles centered on Harrisburg, an area of about 1.5 million people (unless we have another problem at Three Mile Island).

Each employee was provided with a copy of this map and was instructed to enter the proper zone on the questionnaire.

Step 3.

To accomplish matching employees the following Level II, TRS-80 program was written. Before getting into a general discussion of the program, it would be well mentioning a somewhat unorthodox method of saving and loading data to and from tape. When first looking at the program, it is surprising that only one array is dimensioned – line 50. This is all that is necessary since all information for each record is concatenated into one string called N$. This is done at lines 390 or 410. By using this method, record and retrieve times to and from cassette tape are greatly reduced.

The Program

Lines 10 through 220 contain the program header, initialize the program, print instructions and print the menu of functions available. The multiple branch statement in line 230 sends the pointer to the proper section of the program as selected from the menu.

Car Pooling program

```
10 '            ##       CAR POOLING       ##
20 '            ##    WALTER K. MCCAHAN    ##
30 '            ##                         ##
40 '            ##     TRS-80  LEVEL II    ##
45 CLEAR 8000
50 DIM N$(81)
60 CLS:PRINT:PRINT:PRINT"THIS PROGRAM MATCHES RIDES THAT ARE AVAILABLE FOR CAR PO
OLING":PRINT
70 PRINT"THE OUTPUT OF THE PROGRAM WILL BE THE NAME AND ADDRESS
80 PRINT"OF THE PERSON FOR WHOM A MATCH IS SOUGHT FOLLOWED BY A"
90 PRINT"LIST OF  FIRST CHOICES THEN  A LIST OF  SECOND CHOICES,"
100 PRINT"AND  A LIST OF  THIRD CHOICES.   THE PERSON  SHOULD BE"
110 PRINT"GIVEN THE LIST ALONG WITH  INSTRUCTIONS TO CONTACT THE"
120 PRINT"THE LISTED PEOPLE.  IN ORDER,  AND ARRANGE  A CAR POOL"
140 PRINT:PRINT:PRINT:PRINT:INPUT"    TO PROCEED PRESS ENTER";X
150 CLS:PRINT:PRINT:PRINT"THE FOLLOWING FUNCTIONS ARE AVAILABLE:"
160 PRINT"       1. ENTER OR ADD TO A LIST OF NAMES
170 PRINT"       2. MAKE CORRECTIONS TO AN EXISTING NAME
180 PRINT"       3. FIND A MATCH"
200 PRINT"       4. SAVE A FILE ONTO CASSETTE
210 PRINT"       5. LOAD A FILE FROM CASSETTE
220 PRINT:PRINT:INPUT"ENTER THE NUMBER OF YOUR CHOICE";X
230 ON X GOTO 300,500,800,1200,1400
300 CLS:PRINT:PRINT:PRINT:INPUT"ENTER THE NUMBER OF NAMES ALREADY ON THE FILE";I
310 CLS:PRINT"ENTER THE PROPER INFORMATION IN RESPONSE TO EACH PROMPT":PRINT
315 FOR I=I+1 TO 80
320 INPUT"NAME";N1$:PRINT
325 M1=30-LEN(N1$):M$=STRING$(M1," "):N1$=N1$+M$
330 INPUT"STREET ADDRESS";N2$:PRINT
335 M2=30-LEN(N2$):N2$=STRING$(M2," "):N2$=N2$+N2$
340 INPUT"CITY & ZIP";N3$:PRINT
345 M3=30-LEN(N3$):N3$=STRING$(M3," "):N3$=N3$+N3$
350 INPUT"TELEPHONE NUMBER";N4$:PRINT
355 M4=15-LEN(N4$):N4$=STRING$(M4," "):N4$=N4$+N4$
360 INPUT"ZONE";N5$
370 PRINT:INPUT"DO YOU HAVE ANOTHER NAME TO ENTER";Q$
380 IF Q$="NO" GOTO 410
390 N$(I)=N1$+N2$+N3$+N4$+N5$
400 CLS:NEXT I
410 N$(I)=N1$+N2$+N3$+N4$+N5$
420 P1=I
430 GOTO 150
500 '           ##    CORRECTION SUBROUTINE    ##
510 CLS:PRINT:PRINT
520 INPUT" ENTER NAME FOR WHICH CORRECTION IS TO BE MADE";S$
522 POKE 16422,88:POKE 16423,4
525 M1=30-LEN(S$):M$=STRING$(M1," "):S$=S$+M$
530 FORI=1TOP1:IF LEFT$(N$(I),30)=S$ GOTO 570
535 NEXT I
540 CLS:PRINT S$;" NOT FOUND ON FILE"
550 PRINT:INPUT" WANT TO TRY AGAIN";Q$
555 IF Q$="NO" GOTO 150
560 GOTO 500
570 GOSUB 3000
580 CLS:PRINT"EACH LINE OF INFOMATION WILL BE DISPLAYED AS"
590 PRINT"IT APPEARS ON THE FILE.  IF IT IS TO BE CORRECTED"
600 PRINT"THE CORRECT INFORMATION SHOULD BE TYPED IN.  IF"
610 PRINT"NO CORRECTION IS TO BE MADE TO THAT LINE PRESS ENTER"
620 PRINT:PRINT N1$
630 INPUT N1$
635 M1=30-LEN(N1$):M$=STRING$(M1," "):N1$=N1$+M$
640 PRINT N2$
650 INPUT N2$
655 M2=30-LEN(N2$):N2$=STRING$(M2," "):N2$=N2$+N2$
660 PRINT N3$
670 INPUT N3$
675 M3=30-LEN(N3$):N3$=STRING$(M3," "):N3$=N3$+N3$
680 PRINT N4$
690 INPUT N4$
695 M4=15-LEN(N4$):N4$=STRING$(M4," "):N4$=N4$+N4$
700 PRINT N5$
710 INPUT N5$
715 N$(I)=N1$+N2$+N3$+N4$+N5$
720 CLS:PRINT " FOLLOWING IS THE CORRECTED FILE"
730 GOSUB 4000
735 POKE 16422,141:POKE 16423,5
740 PRINT:INPUT"DO YOU HAVE ANOTHER CORRECTION TO MAKE";Q$
750 IF Q$="YES" GOTO 500
760 GOTO 150
800 '           ##    MATCH A SPECIFIC PERSON    ##
810 CLS:PRINT:PRINT:INPUT"ENTER NAME FOR WHICH A MATCH IS SOUGHT";S$
815 M1=30-LEN(S$):M$=STRING$(M1," "):S$=S$+M$
820 FOR I=1TO P1:IF S$=LEFT$(N$(I),30)GOTO 840
825 NEXT I
830 PRINT S$;" NOT FOUND ON FILE": GOTO 920
840 GOSUB 3000
845 N9$=N5$:N5=VAL(N5$)
850 GOSUB 4000
860 LPRINT:LPRINT"          FOLLOWING IS A LIST OF POSSIBLE RIDES FOR THE"
870 LPRINT"         PERSON NAMED ABOVE.  THE RIDES LISTED UNDER FIRST"
880 LPRINT"         CHOICE ARE MOST DESIRABLE, BUT THE SECOND CHOICES"
890 LPRINT"         ARE ALSO VERY BENEFICIAL"
900 GOSUB 5000
910 GOSUB 6000
920 PRINT:PRINT:INPUT"DO YOU HAVE ANOTHER NAME TO MATCH";Q$
930 IF Q$="YES" GOTO 800
940 GOTO 150
1200 '          ##      SAVE ONTO CASSETTE      ##
1210 CLS:PRINT:PRINT:PRINT
1220 INPUT"WHEN CASSETTE IS READY PRESS ENTER";X
1230 CLS:PRINT:PRINT:PRINT
1240 PRINT"NOW RECORDING ONTO CASSETTE"
1250 PRINT#-1,P1
1260 FOR  I=1 TO P1
1270 PRINT#-1, N$(I)
1280 NEXT I
1290 CLS:PRINT:PRINT:PRINT
1300 INPUT" RECORDING NOW COMPLETE - PRESS ENTER  TO PROCEED";X
1310 GOTO 150
1400 '          ##      LOAD FROM CASSETTE      ##
1410 CLS:PRINT:PRINT:PRINT
1420 INPUT"WHEN CASSETTE IS READY PRESS ENTER";X
1430 CLS
1440 PRINT:PRINT:PRINT:PRINT"NOW LOADING FROM CASSETTE"
1445 INPUT#-1,P1
1450 FOR I=1 TO P1:INPUT#-1,N$(I):NEXT I
1460 CLS:PRINT:PRINT:PRINT" LOADING NOW COMPLETE";P1;" ITEMS ON THE FILE"
1470 PRINT:PRINT:INPUT "PRESS ENTER TO PROCEED";X
1480 GOTO 150
3000 '          ##      TO BREAK DOWN STRINGS      ##
3010 N1$=LEFT$(N$(I),30)
3020 N2$=MID$(N$(I),31,30)
3030 N3$=MID$(N$(I),61,30)
3040 N4$=MID$(N$(I),91,8)
3050 N5$=RIGHT$(N$(I),2)
3055 SW=1
3060 RETURN
4000 '        ##  TO PRINT A RECORD      ##
4010 LPRINT:LPRINT TAB(10)N1$:LPRINTTAB(10)N2$:LPRINTTAB(10)N3$:LPRINTTAB(10)N4$:
LPRINTTAB(10)N5$
4020 RETURN
5000 '          ##      TO SELECT FIRST CHOICES      ##
5005 LPRINT:LPRINT:LPRINT"                        FIRST CHOICES":LPRINT
5010 FOR I=1 TO P1+1
5015 GOSUB 3000
5020 IF N9$=N5$ AND S$<>N1$ GOTO 5030 ELSE GOTO 5040
5030 GOSUB 4000
5035 SW=0
5040 NEXT I
5050 RETURN
6000 '          ##      TO SELECT SECOND CHOICES      ##
6005 LPRINT:LPRINT:LPRINT"                       SECOND CHOICES":LPRINT
6010 FOR I=1 TO P1
6020 R=VAL(RIGHT$(N$(I),2))
6030 IFR=2AND N5=1GOSUB3000
6040 IFR=5 AND N5=1GOSUB3000
6050 IFR=6ANDN5=1GOSUB3000
6060 IFR=1AND N5=2GOSUB3000
6070 IFR=3AND N5=2GOSUB3000
6080 IFR=5ANDN5=2GOSUB3000
6090 IFR=6ANDN5=2GOSUB3000
6095 IF R=7ANDN5=2GOSUB3000
6100 IFR=2ANDN5=3GOSUB3000
6110 IFR=4ANDN5=3GOSUB3000
6120 IFR=6ANDN5=3GOSUB3000
6130 IFR=7ANDN5=3GOSUB3000
6140 IFR=8ANDN5=3GOSUB3000
6150 IFR=3ANDN5=4GOSUB3000
6160 IFR=7ANDN5=4GOSUB3000
6170 IFR=8ANDN5=4GOSUB3000
6180 IFR=1ANDN5=5GOSUB3000
6190 IFR=2ANDN5=5GOSUB3000
6200 IFR=6ANDN5=5GOSUB3000
6210 IFR=9ANDN5=5GOSUB3000
6220 IFR=10ANDN5=5GOSUB3000
6230 IFR=1ANDN5=6GOSUB3000
6240 IFR=2ANDN5=6GOSUB3000
6250 IFR=3ANDN5=6GOSUB3000
6260 IFR=5ANDN5=6GOSUB3000
6270 IFR=7ANDN5=6GOSUB3000
6280 IFR=9ANDN5=6GOSUB3000
6290 IFR=10ANDN5=6GOSUB3000
6300 IFR=11ANDN5=6GOSUB3000
6310 IFR=2ANDN5=7GOSUB3000
6320 IFR=3ANDN5=7GOSUB3000
6330 IFR=4ANDN5=7GOSUB3000
6340 IFR=6ANDN5=7GOSUB3000
6350 IFR=8ANDN5=7GOSUB3000
6360 IFR=10ANDN5=7GOSUB3000
6370 IFR=11ANDN5=7GOSUB3000
6380 IFR=12ANDN5=7GOSUB3000
6390 IFR=3ANDN5=8GOSUB3000
6400 IFR=4ANDN5=8GOSUB3000
6410 IFR=7ANDN5=8GOSUB3000
6420 IFR=11ANDN5=8GOSUB3000
6430 IFR=12ANDN5=8GOSUB3000
6440 IFR=5ANDN5=9GOSUB3000
6450 IFR=6ANDN5=9GOSUB3000
6460 IFR=10ANDN5=9GOSUB3000
6470 IFR=13ANDN5=9GOSUB3000
6480 IFR=14ANDN5=9GOSUB3000
6490 IFR=5ANDN5=10GOSUB3000
6500 IFR=6ANDN5=10GOSUB3000
6510 IFR=7ANDN5=10GOSUB3000
6520 IFR=9ANDN5=10GOSUB3000
6530 IFR=11ANDN5=10GOSUB3000
6540 IFR=13ANDN5=10GOSUB3000
6550 IFR=14ANDN5=10GOSUB3000
6560 IFR=15ANDN5=10GOSUB3000
6570 IFR=6ANDN5=11GOSUB3000
6580 IFR=7ANDN5=11GOSUB3000
6590 IFR=8ANDN5=11GOSUB3000
6600 IFR=10ANDN5=11GOSUB3000
6610 IFR=12ANDN5=11GOSUB3000
6620 IFR=14ANDN5=11GOSUB3000
6630 IFR=15ANDN5=11GOSUB3000
6640 IFR=16ANDN5=11GOSUB3000
6650 IFR=7ANDN5=12GOSUB3000
6660 IFR=8ANDN5=12GOSUB3000
6670 IFR=11ANDN5=12GOSUB3000
6680 IFR=15ANDN5=12GOSUB3000
6690 IFR=16ANDN5=12GOSUB3000
6700 IFR=9ANDN5=13GOSUB3000
6710 IFR=10ANDN5=13GOSUB3000
6720 IFR=14ANDN5=13GOSUB3000
6730 IFR=9ANDN5=14GOSUB3000
6740 IFR=10ANDN5=14GOSUB3000
6750 IFR=11ANDN5=14GOSUB3000
6760 IFR=13ANDN5=14GOSUB3000
6770 IFR=15ANDN5=14GOSUB3000
6780 IFR=10ANDN5=15GOSUB3000
6790 IFR=11ANDN5=15GOSUB3000
6800 IFR=12ANDN5=15GOSUB3000
6810 IFR=14ANDN5=15GOSUB3000
6820 IFR=16ANDN5=15GOSUB3000
6830 IFR=11ANDN5=16GOSUB3000
6840 IFR=12ANDN5=16GOSUB3000
6850 IFR=15ANDN5=16GOSUB3000
6860 IF SW=1ANDS$<>N1$GOSUB4000
6865 SW=0
6870 NEXT I
6880 RETURN
```

Lines 300 through 430 accomplish the function of building or adding to a file of data. Each line of data is prompted by the program, and the operator need only answer each prompt to build a file. Lines 325, 335, 345 and 355 set the size of each line. The string sizes of name, street, city and zip are set at 30 characters each, while the telephone number has a field of 15 characters and zone is left at either one or two characters, depending on the zone entered. This field sizing is done by adding "null characters" up to the field size.

The reason behind all this string manipulation is to ensure that each data element such as name, address, city, etc., starts at the same position in the array containing the N$ variables. This must be done so that the portion of the program that breaks N$ back into the various fields knows where to start looking for each data element.

Lines 500 through 760 allow changes to be made to data already in the file. Here again each string is set to size and concatenated after the correction is made. Lines 800 through 940 are used to establish and print out the desired relationship between the zones of the employees in the file.

```
JAMES P SMITH
816 B STREEN
MILLERSTOWN PA 17777
766-2311
 1

WILLIAM MELLAN
12 HERO CIRCLE
NEWPORT PA 17689
766-1818
 1

HOWARD CARR
17 SOUTH RD
LIVERPOOL PA 17532
737-2218
 2

HAROLD BRISKIN
18 ROBERTS RD
MILLERSBURG PA 17505
761-6160
 2

WAYNE L HELD
4717 WILLOW RD
HAILFAX PA 17535
761-9494
 2

EUGENE BLACK
22 CENTER HILL DR
ELIZABETHVILLE PA 17199
761-7714
 3

ALBERTA M CHASE
44 MEADOW LANE
ELIZABETHVILLE PA 17199
761-1141
 3

KENNETH KAIN
5703 WALTERS LANE
PINE GROVE PA 17212
737-7311
 4

PHILLIP BURNS
181 PINE ST
WILLIAMSTOWN PA 17222
737-1921
 4

SAMUEL H TOWELL
10 BRIDGEVIEW
NEW BLOOMFIELD PA PA 17515
733-3311
 5

JOHN DAY
123 CEDAR COURT
NEW BLOOMFIELD PA 17515
733-1977
 5

MARTIN MACK
75 LEE ST
ELLIOTSBRUG PA 17512
733-7119
 5

CHARLES MARCH
12 MALLARD WAY
DUNCANNON PA 17085
733-0110
 6

ROBERT LONG
828 PARK
DUNCANNON PA 17085
733-4414
 6

LEE PARKER
146 SHIREMONT DR
ENOLA PA 17099
733-7369
 6

ROD OLSON
111 CARROL ST
HARRISBURG PA 17019
777-4219
 7

ALAN ROTH
78 BROWNING RD
HARRISBURG PA 17108
777-7272
 7

BRUCE BROWN
69 KENWOOD AE
HARRISBURG PA 17108
777-9891
 7

RICHARD QUINN
1111 MARCH RD
LINGLESTOWN PA 17100
777-8282
 7

TED BARD
5 OAK DRIVE
HERSHEY PA 17119
233-2347
 8

THOMAS CORRY
2022 SOUTH ST.
PALMYRA PA 17087
234-5066
 8

MICHAEL FAME
12 MAYWOOD AVE
HERSHEY PA 17119
233-7111
 8

PAUL HERBERT
1216 STONER DR
CARLISLE PA 17212
234-1211
 9

DAVID HOOVER
3 ARBAN CIRCLE
MECHANICSBURG PA 17177
234-4427
 9

DANIEL HAWKINS
1212 SPRING ST
CARLISLE PA 17212
234-2985
 9

GERALD GREEN
8 MAYS AVE
NEW CUMBERLAND PA 17070
766-9312
10

JAMES ALEXANDER
42 GRANITE RD
CAMP HILL PA 17011
766-4102
10

HENRY HERD
72 WYNGATE RD
MIDDLETOWN PA 17043
739-0421
11

MARY WARD
12 OLD MILL
STEELTON PA 17144
739-1109
11

EDWARD KOONS
400 BAKER ST
ELIZABETHTOWN PA 17771
739-9678
12

ROGER GREENWOOD
11 PHILLIPS AVAE
LATIMORE PA 17041
737-2391
13

NANCY WELLS
665 BEDOFRD
EAST BERLIN PA 17222
739-1904
14

RITA ULMER
4 WEST AVE
CLEAR SPRINGS PA 17219
766-3982
14

MARK WILLIAMS
18 LINCOLN WAY
YORK PA 17843
7337-190
15

ROSE SIMONS
18 BLAIR AVE
YORK PA 17841
761-1002
15

RICHARD SWOPE
OLD ORCHARD RD
WRIGHTSVILLE PA 17759
737-7288
16
```

Fig. 3. Sample list of employees.

The subroutine in lines 3000 through 3060 is used to break down the concatenated string N$ back into the various elements represented by N1$, N2$, N3$, N4$ and N5$. The actual printing of the data elements contained in strings N1$ through N5$ is accomplished by the subroutine in lines 4000 through 4020.

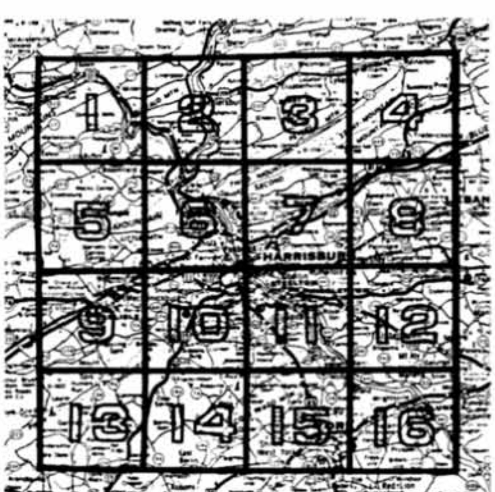


Fig. 2 Grid map of Harrisburg PA area.

First choice matching—where the zone is the same as that for the person being matched—is done by subroutine 5000 through 5050. The actual matching of zones is in line 5020. This line also prevents the printing of the person being matched as a choice.

The subroutine in lines 6000 through 6880 selects the second-choice matches. A name is selected as a second choice if his zone is adjacent to the zone of the person being matched.

Using the Program

In order to ensure the least complicated operation, the program is written in the "tutorial mode," that is, the operator need only answer the questions displayed by the computer in order to run all of the functions.

Function 1—This function is used for initial entry of data into the file and is a simple matter of entering the required information in response to the questions, "NAME?", "STREET ADDRESS?", "CITY & ZIP?", "TELEPHONE NO.?" and "ZONE?". The information is entered directly into the computer from the questionnaires returned by the employees and need not be in any particular order. The file is closed upon answering "NO" to the question "DO YOU HAVE ANOTHER NAME TO ENTER?" or upon reaching 80 names on the file.

Function 2—Since most op-

✓Reader Service—see page 162

erators are not trained typists, I felt it was necessary to allow a simple means of correcting inaccurate data. Entering the name of the person for whom the data is to be corrected will result in each line of data being displayed—one at a time—exactly as it appears on the file. If that line is to be changed, the correct information is typed in. If no correction is to be made to that line, the operator may move on to the next line simply by pressing enter. After all corrections have been completed, the entire corrected file is displayed.

Function 3—This function prints out the name, address and telephone number of the person for whom a match is sought, followed by short instructions, then a list of first choices and then a list of second choices.

To illustrate this function the names appearing in Fig. 3 were entered using function 1. The results of the matching, using function 3, are shown in Fig. 4. As can be seen by this example, first choices are those employees that have the same as the person for whom a match is being sought, while second choices are those who live in a zone which is adjacent. By studying the zoned map and comparing it to the chart in Fig. 5 you will see this relationship.

Functions 4 and 5—These functions are straight-ahead applications for saving and recalling data to and from a cassette file. As explained above, this process is shorter than usual due to the fact that

Fig. 4. Sample run.

```
 HOWARD CARR
 17 SOUTH RD
 LIVERPOOL PA 17532
 737-2218
  2

FOLLOWING IS A LIST OF POSSIBLE RIDES FOR THE
PERSON NAMED ABOVE.  THE RIDES LISTED UNDER FIRST
CHOICE ARE MOST DESIRABLE, BUT THE SECOND CHOICES
ARE ALSO VERY BENEFICIAL


             FIRST CHOICES


 HAROLD BRISKIN
 18 ROBERTS RD
 MILLERSBURG PA 17505
 761-6160
  2

 WAYNE L HELD
 4717 WILLOW RD
 HAILFAX PA 17535
 761-9494
  2


           SECOND CHOICES


 JAMES P SMITH
 816 B STREEN
 MILLERSTOWN PA 17777
 766-2311
  1

 WILLIAM MELLAN
 12 HERO CIRCLE
 NEWPORT PA 17689
 766-1818
  1

 EUGENE BLACK
 22 CENTER HILL DR
 ELIZABETHVILLE PA 17199
 761-7714
  3

 ALBERTA M CHASE
 44 MEADOW LANE
 ELIZABETHVILLE PA 17199
 761-1141
  3
```

```
SAMUEL H TONNELL
10 BRIDGEVIEW
NEW BLOOMIFIELD PA 17515
733-3311
 5

JOHN DAY
123 CEDAR COURT
NEW BLOOMFIELD PA 17515
733-1977
 5

MARTIN MACK
75 LEE ST
ELLIOTSBRUG PA 17512
733-7119
 5

CHARLES MARCH
12 MALLARD WAY
DUNCANNON PA 17085
733-0110
 6

ROBERT LONG
828 PARK
DUNCANNON PA 17085
733-4414
 6

LEE PARKER
146 SHIREMONT DR
ENOLA PA 17099
733-7369
 6

ROD OLSON
111 CARROL ST
HARRISBURG PA 17019
777-4219
 7

ALAN ROTH
78 BROWNING RD
HARRISBURG PA 17108
777-7272
 7

BRUCE BROWN
69 KENWOOD AE
HARRISBURG PA 17108
777-9891
 7

RICHARD QUINN
1111 MARCH RD
LINGLESTOWN PA 17100
777-8282
 7
```

Zone	Zones that are adjacent
1	2,5,6
2	1,3,5,6,7
3	2,3,6,7,8
4	3,7,8
5	1,2,6,9,10
6	1,2,3,5,7,9,10,11
7	2,3,4,6,8,10,11,12
8	3,4,7,11,12
9	5,6,10,13,14
10	5,6,7,9,11,13,14,15
11	6,7,8,10,12,14,15,16
12	7,8,11,15,16
13	9,10,14
14	9,10,11,13,15
15	10,11,12,14,16
16	11,12,15

Fig. 5. Relationship of zones used in selecting second-choice matches.

only one string is transferred to the cassette file.

Step 4.

A match was printed for each person who participated in the project and a copy (similar to that in Fig. 4) was sent to each employee along with the letter shown in Fig. 6.

Conclusion

The results of this venture were very gratifying, in that the number of cars in the company's parking lot have decreased substantially while the number of employees working in the building has increased.

The program has since been used by others, including one large company who ran the program through several times, entering only those who worked on one shift in each run. It has also been used by a company for the registrants of a seminar they were holding.

It is quite evident that car pooling will become more popular—and necessary—in the near future, so I hope that by the use of this program I have contributed my share to the energy-saving program. ■

Dear Fellow Employee:

About two weeks ago you were asked to complete and return a questionnaire which was concerned with car pooling.

Attached is a computer printout of the results of this project as they relate to you.

On the printout you will find a list of people who represent the optimum prospects for forming a car pool—these people are under the category of first choices—and all live quite close to you. The second category is those people who are second choices. These people do not live as close as those in the first category, but do live close enough to make a carpool worthwhile.

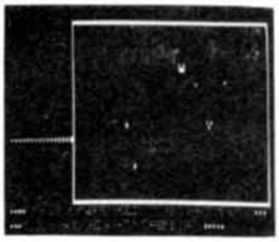
With this list we have provided you with the basic information needed to form your own car pool—the rest must be worked out by you. The easiest way to form a pool is to start calling your fellow employees starting from the top of the list (each employee's telephone number is given) and make your own meeting arrangements.

Sincerely,

The Management

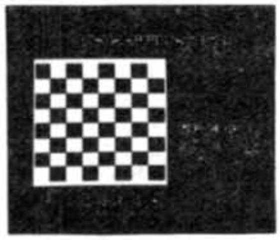
Fig. 6.

SOFTWARE → TRS-80 ← SOFTWARE



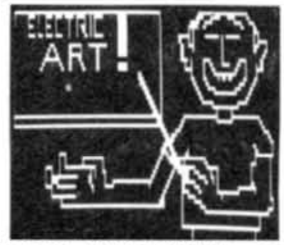
PACKAGE ONE INCLUDES: GRAPHIC-TREK "2000" — This full graphics, real time game is full of fast, exciting action! Exploding photon torpedoes and phasers fill the screen! You must actually navigate the enterprise to dock with the giant space stations as well as to avoid klingon torpedoes! Has shields, galactic memory readout, damage reports, long range sensors, etc! Has 3 levels for beginning, average, or expert players! ★ **INVASION WORG** — Time: 3099, Place: Earth's Solar System Mission: As general of Earth's forces, your job is to stop the Worg invasion and destroy their outposts on Mars, Venus, Saturn, Neptune, etc! Earth's Forces: Androids — Space Fighters — Lazer Cannon — Neutrino Blasters! Worg Forces: Robots — Saucers — Disintegrators — Proton Destroyers! Multi level game lets you advance to a more complicated game as you get better! ★ **STAR WARS** — Manuever your space fighter deep into the nucleus of the Death Star! Drop your bomb, then escape via the only exit. This graphics game is really fun! May the Force be with you! ★ **SPACE TARGET** — Shoot at enemy Ships with your missiles. If they eject in a parachute, capture them — or if you're cruel, destroy them! Full graphics, real time game! ★ **SAUCERS** — This fast action graphics game has a time limit! Can you be the commander to win the distinguished cross! Requires split second timing to win! Watch out!

ONLY $12.95



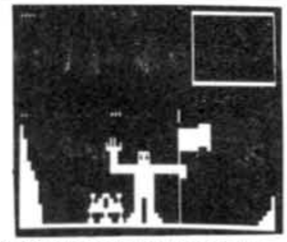
PACKAGE TWO INCLUDES: CHECKERS 2.1 — Finally! A checkers program that will challenge everyone! Expert as well as amateur! Uses 3-ply tree search to find best possible move. Picks randomly between equal moves to assure you of never having identical games. ★ **POKER FACE** — The computer uses psychology as well as logic to try and beat you at poker. Cards are displayed using TRS-80's full graphics. Computer raises, calls, and sometimes even folds! Great practice for your Saturday night poker match! (Plays 5 card draw). ★ **PSYCHIC** — Tell the computer a little about yourself and he'll predict things about you, you won't believe! A real mind bender! Great amusement for parties. ★ **TANGLE MANIA** — Try and force your opponent into an immobile position. But watch out, they're doing the same to you! This graphics game is for 2 people and has been used to end stupid arguments. (And occasionally starts them!) ★ **WORD SCRAMBLE** — This game is for two or more people. One person inputs a word to the computer while the others look away. The computer scrambles the word, then keeps track of wrong guesses.

ONLY $12.95



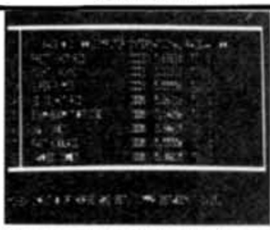
PACKAGE THREE INCLUDES: POETRY — This program lets you choose the subject as well as the mood of the poem you want. You give TRS-80 certain nouns or hames, then the mood, and it does the res'! It has a 1000-word + vocabulary of nouns, verbs, adjectives and adverbs! ★ **ELECTRIC ARTIST** — Manual: draw, erase, move as well as, Auto: draw, erase and move. Uses graphics bits not bytes. Saves drawing on tape or disk! ★ **GALACTIC BATTLE** — The Swineus enemy have long range phasers but cannot travel at warp speed! You can, but only have short range phasers! Can you blitzkrieg the enemy without getting destroyed! Full graphics — real time! ★ **WORD MANIA** — Can you guess the computer's words using your human intuitive and logical abilities? You'll need to, to beat the computer! ★ **AIR COMMAND** — Battle the Kamikaze pilots. Requires split second timing. This is a FAST action arcade game.

ONLY $12.95



PACKAGE FOUR INCLUDES: LIFE — This Z-80 machine language program uses full graphics! Over 100 generations per minute make it truly animated! You make your starting pattern, the computer does the rest! Program can be stopped and changes made! Watch it grow! ★ **SPACE LANDER** — This full graphics simulator lets you pick what planet, asteroid or moon you wish to land on! Has 3 skill levels that make it fun for everyone. ★ **GREED II** — Multi-level game is fun and challenging! Beat the computer at this dice game using your knowledge of odds and luck! Computer keeps track of his winnings and yours. Quick fast action. This game is not easy! ★ **THE PHARAOH** — Rule the ancient city of Alexandria! Buy or sell land. Keep your people from revolting! Stop the rampaging rats. Requires a true political personality to become good! ★ **ROBOT HUNTER** — A group of renegade robots have escaped and are spotted in an old ghost town on Mars! Your job as "Robot Hunter" is to destroy the pirate machines before they kill any more settlers! Exciting! Challenging! Full graphics!

ONLY $12.95



PACKAGE FIVE INCLUDES: SUPER HORSERACE — Make your bets just like at the real racetrack! 8 horses race in this spectacular graphic display! Up to 9 people can play! Uses real odds but has that element of chance you see in real life! Keeps track of everyone's winnings and losses. This is one of the few computer simulations that can actually get a room of people cheering! ★ **MAZE MOUSE** — The mouse with a mind! The computer generates random mazes of whatever size you specify, then searches for a way out! The second time, he'll always go fastest route! A true display of artificial intelligence! Full graphics, mazes & mouses! ★ **AMOEBA KILLER** — You command a one man submarine that has been shrunken to the size of bacteria in this exciting graphic adventure! Injected into the president's bloodstream, your mission is to destroy the deadly amoeba infection ravaging his body! ★ **LOGIC** — This popular game is based on Mastermind but utilizes tactics that make it more exciting and challenging — has 2 levels of play to make it fun for everyone. ★ **SUBMARINER** — Shoot torpedoes at the enemy ships to get points. Fast action graphics, arcade type game is exciting and fun for everybody!

ONLY $12.95

HARDWARE → TRS-80 ← HARDWARE

MICRO SPEED

Upgrade your "slow" TRS-80 to a **SUPER FAST MACHINE!!** (2.66 MHZ) over **50% FASTER!** Some of the features:

Auto turn-off during cassette or disk access. (This means NO lost programs EVER!) (Turns back on automatically too!) MANUAL control. (Unit may be turned on or off at any time. Yes even during program execution!) Keyboard indicator light "blinks" when micro-speed is on. Stops blinking when off! **Don't wait for SARGON II or any other program!!!** Comes with easy to follow instructions. (Some soldering required.) **OR** take to your local computer store or TV-Appliance Center for quick installation. (5-10 minutes!!) Works with any model, TRS-80.

ONLY $24.95 complete

MICRO BEEP

Simple hook up: Just plug cassette remote jack into unit.

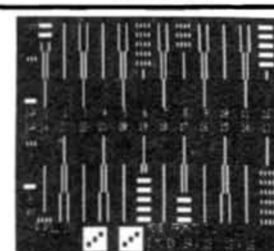
EASILY CONTROLLED FROM BASIC:

OUT 255,4 = on
OUT 255,0 = off

MICRO-BEEP make games more fun as well as provide useful sound output for professional applications!

Works with Any Model I TRS-80

ONLY $9.95 complete



PACKAGE SEVEN INCLUDES: BACKGAMMON 5.0 — 2 different skill levels make this game a challenge to average or advanced players FAST (15 second avg) Looks for best possible move to beat you! FANTASTIC GRAPHICS. Plays doubles and uses international rules ★ **SPEED READING** — Increases your reading speed. Also checks for comprehension of material. Great for teenagers and adults to improve reading skills ★ **PT 109** — Drop depth charges on moving subs. Lower depths get higher points in this fast action graphics game. ★ **YAHTZEE** — Play Yahtzee with the computer. This popular game is even more fun and challenging against a TRS-80! ★ **WALL STREET** — Can you turn your $50.000 into a million dollars? That's the object of this great game Simulates an actual stock market!

ONLY $12.95

PACKAGE SIX INCLUDES: 20 HOME FINANCIAL PROGRAMS — Figures amortization, annuities, description rates, interest tables, earned interest on savings and much, much more. These programs will get used again and again. A must for the conscientious, inflation minded person.

ONLY $12.95

SIMUTEK ✓19

Exceptional Products through Research & Imagination

Send Check, Money Order or Bank Card No. orders to:

SIMUTEK
P.O. Box 35298
Tucson, AZ 85740

Please Add 2.50 Per Order For Postage & Handling

Master Charge — **24 HOUR (7 days) HOTLINE (602) 882-3948** (C.O.D. $3 extra) — Visa

Same Day Shipment on Bank Cards, Money Orders & C.O.D.

All Tape Programs Require a Minimum of 16K Level 2
Packages Available on Diskette (32K System) $4.25 Extra

3 or More Packages Get 10% Discount

Dealer Inquiries Invited

TRS-80 IS A REGISTERED TRADEMARK OF TANDY CORP.

✓Reader Service — see page 162

Add extra memory to your 80 without the expense of an expansion interface.

Home Brew Memory

Richard Ragucci
507 Birchwood Ct.
North Brunswick, NJ 08902

After familiarizing myself with a Level I BASIC TRS-80, I decided to upgrade to Level II adding 16K of dynamic RAM.

But, upgrading any further—adding more memory and I/O interfacing—meant spending $250 for an expansion interface and that's without extra memory.

I also owned a custom-designed Z-80 processor with a 16K static RAM board and I/O board, so I decided to interface as much of my Z-80 system to the TRS-80 as possible.

This article will discuss how simple it is to add extra memory to the TRS-80 without the expense of an expansion interface.

The memory map (Fig. 1) of the TRS-80 shows the first 16K of usable memory from 4000 to 7FFF and a second block of unused memory from 8000 to FFFF. The second block is where the extra 32K of memory is decoded in order for the TRS-80 to use it. Therefore, my 16K static RAM board had to be decoded at addresses 8000 to BFFF.

HEX ADDRESS	DESCRIPTION OF USAGE	
0000 TO 0FFF	LEVEL I ROMS	LEVEL II
1000 TO 3FFF		
3800 TO 38FF	KEYBOARD	
3900 TO 3BFF	NOT USED	
3C00 TO 3FFF	VIDEO DISPLAY	
4000 TO 41FF	RAM USED BY LEVEL I BASIC	
4200 TO 4FFF	USABLE RAM STARTS HERE	4K 8K 16K
5000 TO 5FFF	RAM	
6000 TO 7FFF	RAM	
8000 TO BFFF	NORMALLY NOT USED 16K RAM (EXTRA)	32K BLOCK
C000 TO FFFF	NORMALLY NOT USED 16K RAM (EXTRA)	48K BLOCK

Fig. 1. TRS-80 Memory Map.

Hardware

Fig. 2 shows the simple scheme used in decoding the static RAM. The 74139 is a two to four line decoder; putting A14 and A15 along with RAS* into the A & B select inputs, will decode the RAM board into four possible locations.

When A14 and A15 are low, the board is at location 0000 to 3FFF, but that can't be used due to the locations of ROM and keyboard (see memory map).

With A14 high and A15 low, the decoded board address is 4000 to 7FFF, the memory location of the 16K dynamic RAM on the TRS-80's mother-board.

The third location, A15 high and A14 low, decodes at address 8000 to BFFF, where the 16K memory board should be located. For maximum system memory, another 16K board could be located at the fourth location, A14 and A15 high, C000 to FFFF.

Output of the 74LS139 is "or-gated" with RAS* and enables the bus transceiver (74LS245), which isolates the memory board from the data bus when not needed.

RD* enables direction control of the transceiver. A low on RD*, RAS* and the board select allows data to be read from memory. The WR* is connected to RAM and tells it whether to read or write.

The previous portion of the decoding can be used for any 16K random access memory board. The 2114s in my board are 1K by 4 RAM, meaning two chips are needed for a full 1K by eight words. Therefore a 16K board needs 32-2114 RAM and 16 separate chip selects—one for each 1K of memory.

The two 74LS138s (Fig. 2) are three to eight-line decoders, giving a total 16 decoded chip selects. Address lines A10, A11, A12 decode the order of selects for each chip, 0-7 and A13 is used to decode whichever one of the 74LS138s is on at any given time.

Other Types of Memory

Decoding chip selects for other types of memory would be different. Another type of mem-

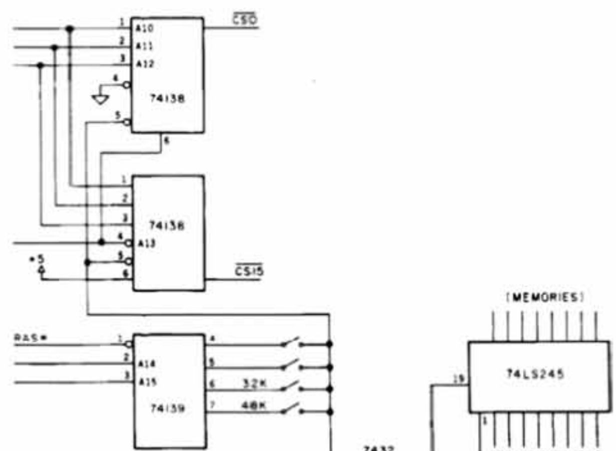


Fig. 2. Static Memory Interface.

Okay, now you've had a chance to see what I have in mind for you with *80 MICROCOMPUTING*. Oh, I admit that we're just getting started and that the magazine will be improving a lot as we go along. We have some interesting ideas in the works for you.

With the TRS-80 (or 90 . . . etc.) being the most popular microcomputer in the entire world, you are going to benefit from this in many ways. The more computers there are out there of one kind . . . the more good programs you are going to have for this system. I hope that is obvious. You may be sure that *80 MICROCOMPUTING* will be packed with the shorter programs and reviews of the larger ones. You can waste an awful lot of money on stuff that looks great in the ads, but fizzles out when you try to use it. You need our reviews.

The wealth of programs will also mean that there will be much better programs for the TRS-80 than any other system. Put yourself in the seat of a computer programmer and you'll understand this. If you are going to spend several months developing a comprehensive program, and it takes all of that to write and debug a big program, would you write it for a system which has sold one hundred units or one which has sold over 300,000 systems? The answer is obvious . . . and this is why we are already seeing programs coming out for the "80" which are far better than anything for any other system on the market. This is tough for other systems . . . the law of the computer jungle.

Between our connections with Instant Software, the largest publisher of microcomputer programs in the world, and Kilobaud MICROCOMPUTING, you know that *80 MICROCOMPUTING* is going to be your most important link with software for the TRS-80.

With Instant Software being sold and promoted in every country in the world where the TRS-80 is being sold, our input of programs is also the best in the world. We get programs submitted from everywhere . . . often from 50 to 100 a week! You'll get the cream of the crop either published or reviewed in "80".

HARDWARE TOO

The same law of the computer jungle holds for hardware. Would you, as a manufacturer, market an accessory for a system which has sold 100 units or would you go first for the one which has sold hundreds of thousands. It is, as with software, self-evident why the great bulk of the hardware accessories for computers are for the TRS-80 these days.

80 MICROCOMPUTING has the advantage of the use of the largest and most complete microcomputer lab in the world . . . the one developed for Instant Software and Kilobaud MICROCOMPUTING. This means that most new pieces of equipment are tested and in use by our staff . . . and this means that we can tell you what we think is outstanding . . . and where we find ripoffs. This lab is important to you.

SUBSCRIBE

If you are not already a subscriber to *80 MICROCOMPUTING*, please get signed up right now. The yearly rates are $15, and that is a bargain. Just one single program of use to you can be worth much more than that. One review of an accessory could save you many times that much investment. I would appreciate it if you would appoint yourself a committee of one to get more subscribers for the magazine. You will benefit even more than we do here at the magazine . . . because the more readers we have, the more ads we will be able to attract . . . and the more ads, the more pages of articles you will get every month.

The "80" market can, I think, support a couple of hundred pages of ads . . . and that would mean a magazine of nearly 500 pages a month. That should hold you. You may not have time left to use your computer.

ENCYCLOPEDIA

If you've read Kilobaud MICROCOMPUTING, you know that I try hard not to duplicate published material. My concept is that every reader should save every issue (we sell inexpensive boxes for this so they can sit on your library shelf) and treat the magazine as a continuing encyclopedia of computing. I make sure that much of the material in each issue is written in simple language so it will be understandable by even the rawest newcomer to computers. Oh, I have articles for the more advanced users too, so you'll have something to look back over later and use as your understanding of your system grows.

Try to think of *80 MICROCOMPUTING* as more of a large club newsletter than an ivory tower high-level publication. I'll leave the pomp to other publishers . . . the ones with the well-deserved inferiority complexes who cater to their inadequacies by publishing esoteric baloney. This magazine is written by the readers and edited by people whose aim is to help you enjoy your TRS-80.

SAVE

With each issue costing $2 at your computer store, that's $24 a year. For $15 a year you can subscribe . . . at least for now. As the magazine expands, please do not be surprised if the cover price increases, along with the subscription price. I started 73 Magazine for radio amateurs twenty years ago with a cover price of 37¢ (two for 73¢) and it is up to $2.50 a copy now (and it is the largest of the ham magazines).

For you bargain hunters . . . and those who find that one year goes by all too rapidly, the three year rate for "80" is $36. This, too, will be going up . . . reflecting the inflation, paper increases, postage increases, and a short vacation for me in Hong Kong next year. Someone has to pay for that.

If coupon below has been used, please fill out subscription form on the Reader Service card in the back of the magazine.

Yes! Sign me on as a subscriber to 80 Microcomputing for only $15 a year!

Card # ______________________ Exp.__________

Signature ______________________

Name ______________________

Address ______________________

City ______________ State ________ Zip ________

☐ 12 issues – $15
☐ 36 issues – $36
☐ Please bill me
☐ Payment Enclosed
☐ Master Charge
☐ VISA
☐ American Express

80 microcomputing™

Subscription begins with next published issue.
Back issues, while available, are $3 each.
Canada: $15 per year US funds.
All other foreign subscriptions $20, one year only, US currency only.

PO Box 981 • Farmingdale NY 11737

305B8

ory easily interfaced to the TRS-80 for very little money, are 16K dynamic RAMs, such as 4116s.

Board decoding would be nearly the same as previously discussed, however, there would be two 74LS245s, with the 48K line from the 74LS139 "or-gated" into the other 74LS245 to control the 48K block of memory. Refresh and multiplexing for the address bus must be included, but otherwise, the rest of the board select circuitry is the same.

The 74LS157s are quad two to one-line selectors and are used to multiplex the higher order addresses with the lower order addresses. The sequence of events occurs as follows; the MUX signal from the TRS-80 goes low, allowing A0 to A6 address lines to go to the RAM, with RAS* also low to allow the RAM to accept these address lines. When MUX goes high, A7 to A13 will be presented to the RAM and CAS* will go low to let the RAM accept the higher order addresses. The WR* signal again tells RAM whether to read or write, as was the case with the static RAM.

If you have 16K of static RAM around, interfacing is relatively simple and very cheap. If not, adding up to 32K of dynamic RAM (16K chips) can be done for less than $200 including memory with the method described above. ■

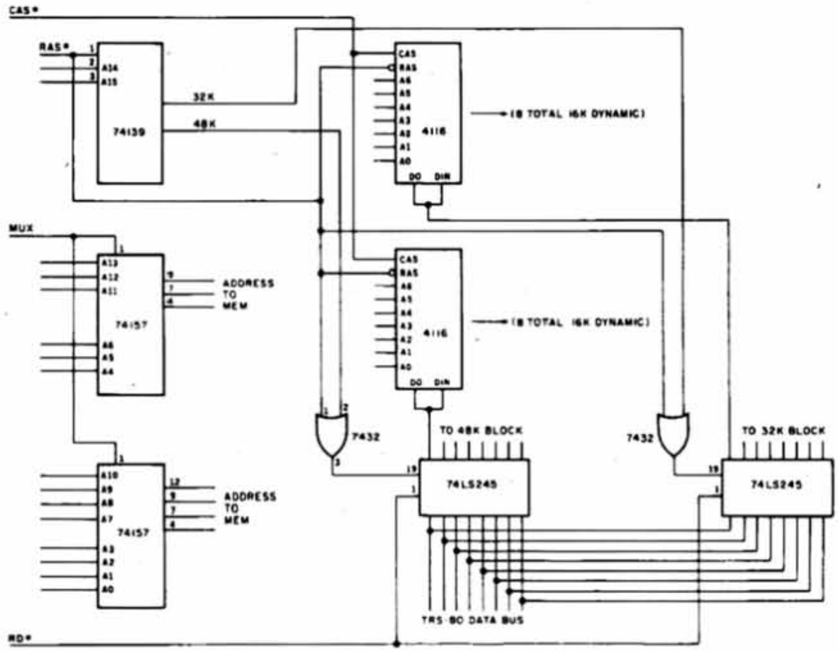
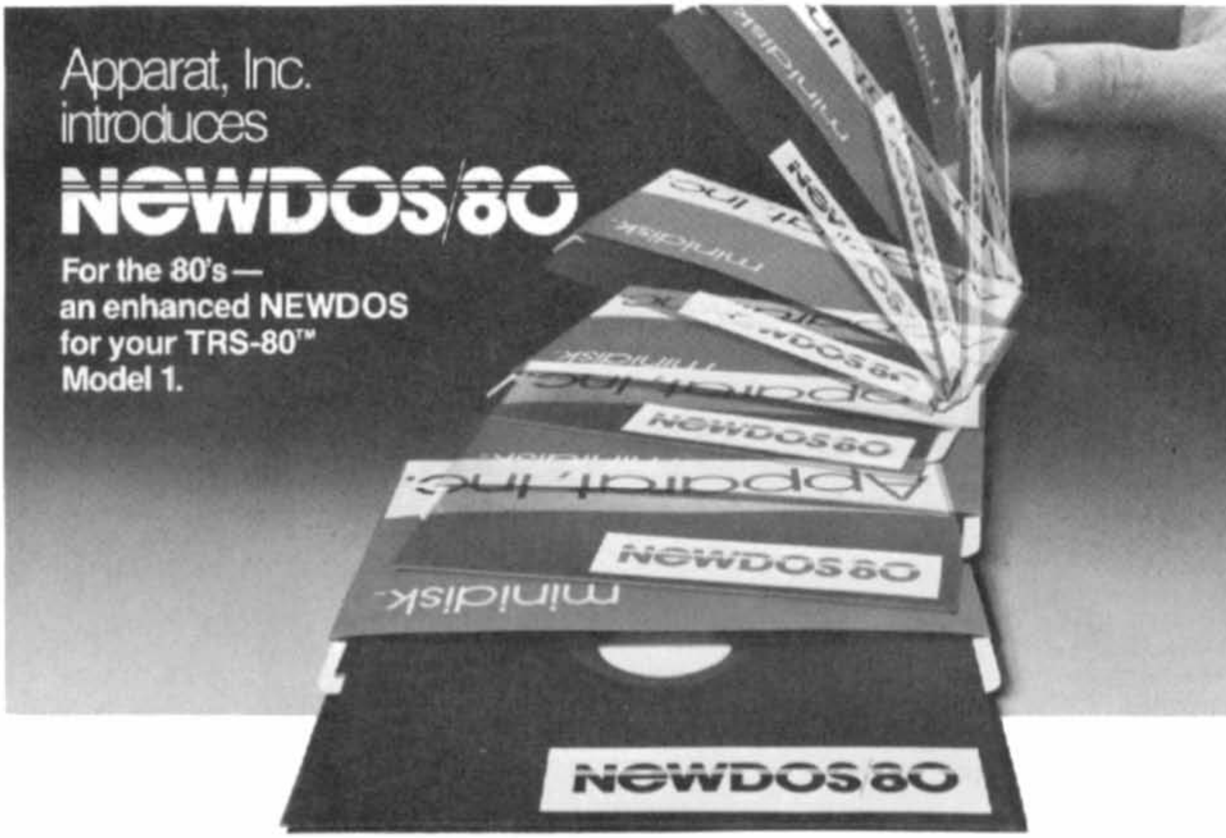


Fig. 3. Inexpensive method to add dynamic RAM.

Apparat, Inc. introduces

NEWDOS/80

For the 80's — an enhanced NEWDOS for your TRS-80™ Model 1.



Apparat, Inc., announces the most powerful Disk Operating System for the TRS-80®. It has been designed for the sophisticated user and professional programmer who demands the ultimate in disk operating systems.

NEWDOS/80 is not meant to replace the present version of NEWDOS 2.1 which satisfies most users, but is a carefully planned upward enhancement, which significantly extends NEWDOS 2.1's capabilities. This new member to the Apparat NEWDOS' family is upward compatible with present NEWDOS 2.1 and is supplied on Diskette, complete with enhanced NEWDOS + utility programs and documentation. Some of the NEWDOS/80 features are:

- New BASIC commands that supports files with variable record lengths up to 4095 Bytes long.
- Mix or match disk drives. Supports any track count from 18 to 80. Use 35, 40 or 77 track 5" mini disks drives or 8" disk drives, or any combination.
- A security boot-up for BASIC or machine code application programs. User never sees "DOS READY" or ">READY" and is unable to "BREAK", clear screen, or issue any direct BASIC statement including "LIST".
- New editing commands that allow program lines to be deleted from one location and moved to another or to allow the duplication of a program line with the deletion of the original.
- Enhanced and improved RENUMBER that allows relocation of subroutines.
- Powerful chaining commands.
- Device handling for routing to display and printer simultaneously.
- CDE function; simultaneous striking of the C, D and E keys will allow the user to enter a mini-DOS to perform some DOS commands without disturbing the resident program.
- Upward compatible with NEWDOS 2.1 and TRSDOS 2.3.
- Includes Superzap 3.0 and all Apparat 2.1 utilities.

NEWDOS/80 with all of the NEWDOS + utility programs, many of which have been enhanced, is priced at just $149.00 and is available at most TRS-80 dealers. Previous NEWDOS owners may receive full trade in allowance toward the purchase of NEWDOS/80 by including with their order the *serial number* of their NEWDOS 2.1 diskette, the *price paid* and *where purchased*. In most cases that purchase price will be subtracted from the price of NEWDOS/80. As with NEWDOS 2.1, NEWDOS/80 relies on the TRSDOS and Disk Basic Reference Manual published by Radio Shack. NEWDOS/80 documentation supports its enhancements and upgrades only.

VISA | master charge

Apparat, Inc. ✓264

MICROCOMPUTER TECHNOLOGY INCORPORATED

TO UPGRADE TO NEWDOS/80, COMPLETE AND MAIL TO:

Apparat, Inc.
7310 E. Princeton Ave.
Denver, CO 80237
303/758-7275 303/741-1778

OR

Microcomputer Technology, Inc.
3304 W. MacArthur Blvd.
Santa Ana, CA 92704
714/979-9923

PREVIOUS NEWDOS OWNERS ONLY

Serial # ____________________

Price ____________ Purchased ____________

☐ Check ☐ Money Order ☐ Master Charge ☐ Visa

Card No. ____________ Expiration Date ____________

Colo. residents add 6.5% sales tax. Cal. residents add 6% sales tax.
Add $10.00 postage and handling.

Please rush ________ NEWDOS/80 to:

Name ____________________

Address ____________________

City ____________ State ____________ Zip ________

Phone ____________

Upgrade offer expires July 31, 1980. 0/1

A way of preventing your delicate cassette relay from welding itself shut.

Destick Your Relay

Herman Lukoff
506 Dreshertown Rd.
Fort Washington PA 19034

I had heard about a relay sticking problem in the TRS-80 recorder but it really didn't worry me, until one day when my tape ran beyond the end of data and there was nothing I could do to stop it short of turning power off.

A tiny reed relay (K1) on the computer board provided the contacts for turning the motor on in the recorder. Using a milliammeter, I measured the running current at 130 milliamperes. The starting current, however, was different with peaks of ½ ampere noted. This high starting current was undoubtedly welding the relay contacts together.

Solution

There are two solutions to the problem. One is to replace the tiny reed relay with a more substantial one capable of handling the high starting current. A look at the computer board convinced me that it is impossible to put a larger relay in the same position.

You could install a larger relay elsewhere on the board and run wire leads, but this presents a problem to many people. For one, it voids the warranty and secondly, there is always the chance of damaging another component.

The second solution is the transistorized current amplifier shown in Fig. 1 which requires no modification to the TRS-80.

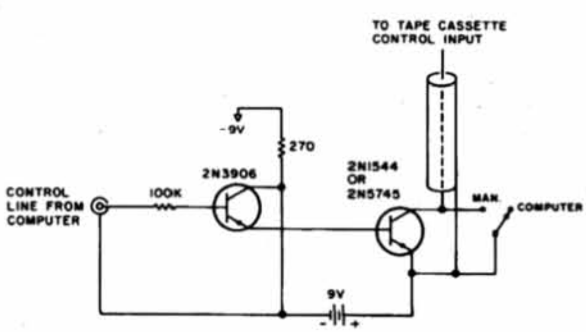


Figure 1

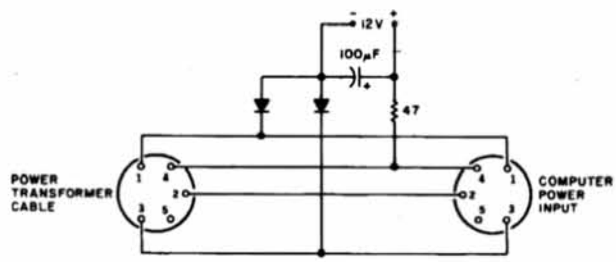


Fig. 2

The output transistor handles the high starting current of the motor while the reed relay contacts see a miniscule current of 80 microamperes: at that level the reed relay contacts should last forever.

The motor operates at 6 volts; the voltage drop across the output transistor has to be very small by comparison. But how small?

In one test in which the voltage was dropped .4 volt, there was no detectable change in tape speed. The measurement was made by reading the cassette counter after five minutes of running the tape.

The motor probably has some form of speed control to desensitize it to voltage changes. Nevertheless, keep the voltage drop to less than .2 volt.

The secret to obtaining a low voltage drop across the transistor is to select a high current, moderate/high gain transistor and operate it at a fraction of its rated current. In this mode the current gain will be very high and the transistor will operate deep into saturation, producing a low voltage drop. A PNP transistor is required in this application.

My junk box held a 2N1544 germanium power transistor which developed only a .05 volt drop. This transistor is sold in the surplus market. A silicon PNP power transistor type 2N5745 produced a respectable .1 volt drop. Other transistors with similar parameters will probably work as well.

Minimal Hardware

Even though a large transistor is used as the motor driver, the actual power dissipated in it is very low, less than 25 milliwatts, therefore, no heat sink is required. Also, notice that no protective diode is needed at the collector of the transistor. Even though the motor is an inductive load, other circuits, including a large electrolytic capacitor, limit any inductive voltage peaks.

The input 2N3906 transistor functions as an emitter-follower current amplifier and is available from many sources. A negative voltage source is required to provide base current for the two transistors.

Two options are shown. In Fig. 1, a small 9 volt battery similar to those found in transistor radios is used. The current drawn is approximately 10 milliamperes and long battery life is guaranteed, for current is required only when the tape runs.

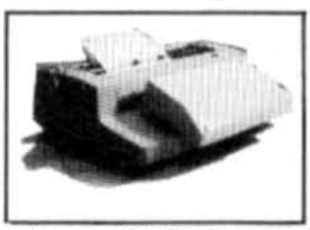
If you wish to avoid the battery, an alternate approach is shown in Fig. 2. The A.C. power cable, formerly plugged into the back of the computer, is now passed through the motor control unit by way of two DIN connectors. A few components provide full wave rectification and −12 volts output that can be substituted directly for the 9 volt battery. The drain on the TRS-80 power transformer is negligible.

A must feature that is easy to include is a small toggle switch to change from computer control to manual. In the manual position, the tape can be rewound, moved forward, etc., without having to pull the control cable out of the cassette unit.

The few components can be mounted on perf board and easily placed in a 4" × 2" × 1½" box. My box was made of aluminum and I decided the safest approach was to electrically isolate the case from any of the circuitry. Therefore, the input jack, output lead and other wiring should not touch the case.

The appropriate jack, plug and box can be obtained from Radio Shack.

Incidentally, the one time the relay contacts stuck, they remained stuck and the only way I could get the relay to work was to give it a sharp rap with my knuckle. Hopefully, this is the last I will hear of the relay sticking problem. ■



Not mathematical—this program 'roots out' your defective memory locations.

Babyroot

Dennis Bathory Kitsz
Roxbury, VT 05669

Babyroot is a program that exercises the TRS-80 RAMs. I developed this program because, during a particularly warm and humid day, my programs started to crash for no apparent reason. I scratched out an inelegant but workable first version and located a wayward address, 4A02.

The program exercises all combinations of bits in three neighboring bytes simultaneously. Locations are repeatedly written to and read from to assure that they are functioning.

The Program

Prepare Babyroot as a BASIC program or use T-BUG to enter the machine code and create a SYSTEM tape.

Load Babyroot using the SYSTEM command; enter the name RAMCHK and when the program has finished loading (about 8 seconds), enter /20480. The program, now located at 5000 hex, works over the RAM from address 6000 to 7FFF hex.

At the end of this sequence, it transfers itself to 7000 hex, exercising RAM from 4000 to 6FFF hex. Each complete cycle takes 12 1/2 minutes on an unmodified TRS-80.

At the beginning of the program, the screen clears and a star appears in the bottom right-hand corner. About five minutes later, the program transfers to high memory. The star vanishes and a graphics block appears in the next screen position. Address 3FFF is presented at the top of the screen, indicating that address 3FFF is "bad". The program reads that address as bad because the video memory bank is deliberately designed short one bit.

The video memory ends at address 3FFF and system RAM begins at 4000. This memory location is inadvertently exercised by the program when it begins working over address 4000 and its two neighbors. It serves as a check that Babyroot itself is operating correctly. (If your TRS-80 has a lowercase hardware modification that saves the extra bit, this memory location should read correctly. No bad bit will be presented.)

Errors in address 4000 to 41E5 are usually terminal. 41E6 to 42E8 might or might not cause significant grief. 42E9 and up is where the BASIC programs sit.

If the bad bit coincides with a fixed program bit, everything will look fine. If it is different from a fixed program bit, that program might never LIST or RUN correctly. If the bad bit coincides with a variable location, it will be invisible at first.

But watch those results!

Reading down from address 7FFF, you find control information, strings and the BASIC stack. Errors here can wipe out the program. So, there is a chance that in normal operation a bad bit might go unnoticed for months.

Keep this program running (it will continuously loop) for no less than 8 hours, if you suspect a bad bit. It took 9 1/2 hours before conditions were right and the balky bit turned up in my system.

The computer can be restored for normal use only by turning it off. All system RAM has been upset by Babyrooting around. If the RAM is too badly damaged, Babyroot itself may not load. Write me for a special version of the program for use in this case.

A Bit Error

When you've discovered a bad RAM, what do you do? A bit error is a sophisticated enough problem that you first might want to leave it to your friendly Radio Shack repair center. However, if you find this suggestion unappealing, you'll want to tackle it yourself.

First, you'll probably want to find the bad bit's position in the byte. Now that you know its address, Table 1 lists a short program to identify that bit.

Enter the least significant byte of the bad RAM address in-

5000	•	21	00	3C	11	01	3C	01	FF	03	36	20	ED	B0	06	08	21
5010	•	A0	3C	36	2A	23	10	FB	06	00	0B	21	##	§§	78	77	AE
5020	•	28	F7	C5	06	08	21	A0	3C	07	38	05	23	10	FA	18	05
5030	•	36	58	23	10	F3	21	A0	3C	C1	18	DE	C9	00	00	00	00

Table 1. Bit Search program.

to the position marked ##. Enter the most significant byte into the position marked §§. (You may relocate this program anywhere in memory, if necessary, to avoid balky addresses.) Now set this program in motion. A good byte will be displayed thus:

****

As bad bits are found, the stars will be replaced by letter X, so:

***X*X**

Bit 7 is left-most in this display and bit 0 is to the right. It will continuously loop, exercising this one address, just as the Babyroot program exercised the whole of the memory. After a given period of time, all the bad bits should be indicated by an X.

Next you'll need to buy replacement 4116 RAMs ($8 to $15), one for each bad bit position. Dynamic RAMs are curious devices. If you don't know how to handle these sensitive MOS devices, find out! Look in *Popular Electronics*, on data sheets, in books, ask a friend or whatever, but don't proceed until you know how to handle them! (And put your cats in the bathroom.) For greatest safety, use a static-free workbench.

Get a TRS-80 technical manual or RAM expansion instruction sheet. Open the case, find the RAM(s) corresponding to the bad bit(s). Pull the RAM(s). Replace each with a new 4116 and run Babyroot again. If the error is gone (and it should be), you're set. If not, double check your work, making sure you've replaced the correct RAM. Switch RAMs. See if the bad bit changes position. Go down the line. You may even have a bad new RAM.

If you're desperate, call me; I know the feeling of sitting awake at 3 a.m., wondering whether or not I should turn my TRS-80 into a flower pot. I'll try to help you with your diagnosis. I hope the program runs for days without turning up a bad bit. Good luck.■

```
5000  •  21 00 3C 11 01 3C 01 FF 03 36 20 ED B0 DD 21 80
5010  •  50 3A FE 3F EE 0A 32 FE 3F DD F9 06 7F 33 10 FD
5020  •  21 00 60 AF 77 4F F5 79 77 23 7C FE 80 28 30 E5
5030  •  DD E5 E1 BC 28 29 E1 F1 77 2B 2B 77 47 7E B8 28
5040  •  05 16 1B D5 18 1B 23 23 7E B8 28 05 16 10 D5 18
5050  •  10 2B 34 0C 7E 47 FE 00 20 CC 23 0E 00 18 C7 18
5060  •  64 7C E6 F0 0F 0F 0F 0F 16 22 18 22 7C E6 0F 16
5070  •  1B 18 1B 7D E6 F0 0F 0F 0F 0F 16 10 18 10 7D E6

5080  •  0F 16 09 18 09 D1 3E D3 92 DD 77 0D 18 FF D5 F5
5090  •  11 00 3C 1A FE 20 28 03 13 18 F8 7B FE 04 20 0A
50A0  •  3E 20 1B 12 1B 12 1B 12 1B 12 F1 FE 0A 30 04 C6
50B0  •  30 18 02 C6 37 12 C5 06 FF 10 FE C1 01 3E C9 92
50C0  •  DD 77 44 18 FF DD E5 DD E5 E1 D1 7A EE 20 57 D5
50D0  •  06 80 2B 1B 10 FC 01 FF 00 ED B0 DD E1 3E 20 DD
50E0  •  AE A2 DD 77 A2 3E F0 DD AE AC DD 77 AC DD E5 E1
50F0  •  2E 11 E9 44 45 4E 4E 49 53 20 4B 49 54 53 5A 00
```

```
10   FOR X = 20480 TO 20735 : READ A : POKE X,A : NEXT X
20   POKE 16526,0 : POKE 16527,80 : PRINT USR(0)
30   DATA 33,0,60,17,1,60,1,255,3,54,32,237,176,221,33,128
40   DATA 80,58,254,63,238,10,50,254,63,221,249,6,127,51,16,253
50   DATA 33,0,96,175,119,79,245,121,119,35,124,254,128,40,48,229
60   DATA 221,229,225,188,40,41,225,241,119,43,43,119,71,126,184,40
70   DATA 5,22,27,213,24,27,35,35,126,184,40,5,22,16,213,24
80   DATA 16,43,52,12,126,71,254,0,32,204,35,14,0,24,199,24
90   DATA 100,124,230,240,15,15,15,15,22,34,24,34,124,230,15,22
100  DATA 27,24,27,125,230,240,15,15,15,15,22,16,24,16,124,230
110  DATA 15,22,9,24,9,209,62,211,146,221,119,13,24,255,213,245
120  DATA 17,0,60,26,254,32,40,3,19,24,248,123,254,4,32,10
130  DATA 62,32,27,18,27,18,27,18,27,18,241,254,10,48,4,198
140  DATA 48,24,2,198,55,18,197,6,255,16,254,193,209,62,201,146
150  DATA 221,119,68,24,255,221,229,221,229,225,209,122,238,32,87,213
160  DATA 6,128,43,27,16,252,1,255,0,237,176,221,225,62,32,221
170  DATA 174,162,221,119,162,62,240,221,174,172,221,119,172,221,229,225
180  DATA 46,17,233,68,69,78,78,73,83,32,75,73,84,83,90,0
```

Total program cycle: 12 minutes 24 seconds

Recommended number of cycles: 40 or more

Babyroot versions: Machine (top) and BASIC (bottom).

Develop a local business program—
but read this before you press ENTER!

Business Programming

Wes Clarke
Westech Corp.
P.O. Box 432
Sparta, NJ 07871

The arrival of the inexpensive microcomputer in the marketplace has opened an entirely new area of opportunity for the businessman who can program. With small business just crying out for some way to ease the burden of paperwork, a real need has developed for good business software.

This software, however, has been slow to emerge and many entrepreneurs have been disillusioned by what business systems have available. As a small businessman, who has developed a successful inventory control program for the TRS-80, I have insights into the needs of small business and the problems encountered writing and marketing good business software.

Maintaining Inventory

The story starts around mid 1967 where, as part owner of a Radio Shack dealership in Sparta, New Jersey, I had a problem: Maintaining my inventory. I had worked in the auto parts business and was using a card system popular in that industry, but the maintenance time was excessive.

The logical answer was a computer, but the prices for business systems were way over what I could afford. A hobby computer seemed like a way to solve my problem, but I knew little about them. I set out to learn. I was just starting my studies when Radio Shack announced the TRS-80—the perfect answer, a system that I could not only use, but sell.

I ordered one immediately and began studying in earnest. I devoured everything I could find on BASIC, the Z-80 and computers in general. I even studied assembly language. I also started looking closely at my problems in the store and planning what I wanted the computer to do and how I wanted it done.

This proved invaluable later on, for it is the consideration of the many minor details of store operation that has made my program a success both for me and for others.

My 16K Level I arrived three days after Christmas 1977, and I dragged it along on a ski vacation, much to the chagrin of my wife. The well-written Level I manual enabled me to quickly convert the general information that I had learned to the specifics of the TRS-80 and I started work on my problem.

It did not take long for me to discover that 16K would not suffice for the complex program that I had in mind. I had to expand to Level II and disk drives for a useful program.

I did not stop working, however, and developed a pilot program on the Level I to test the practicality of my ideas. I completed the program around April, 1978, and, as it turned out, it opened the door for me.

I took it to a dealer training meeting and demonstrated it to the Radio Shack Regional manager. He realized that it met the particular needs of the Dealer/Franchise network and contacted the VP of the Franchise Division who asked if I could have a demonstration version ready for the summer meetings. I agreed to try and, upon receipt of my 32K Level II disk system, was able to complete the program in time for a series of meetings in August, 1978.

. . . to ease the burden of paperwork, a real need has developed for good business software.

The program was well received. Several dealers offered valuable suggestions at that time which I incorporated in the program. I placed the program in operation in my store on September first for testing. It took about two months to get the bugs worked out and the documentation prepared. I started shipping to my first customers in November.

Documentation

So far, everything had gone smoothly, then problems began. It seemed that my documentation left something to be desired. I had written it in the form of other documentation that I had seen, complete with flow charts and detailed program descriptions.

Other users who were programmers were having no problem with it, but most of my customers were businessmen. They did not have time to read all about random files, backups, memory sizes, etc. They just wanted to use the program. It also turned out that my part entry routine was too slow and was causing some frustration.

I started work on revisions. I rewrote the manual as a programmed learning guide—one that would take a novice through the program step by step and teach him to use it. I rewrote the routines that were causing troubles. I developed an automatic loading routine that eliminated the need to set memory size and brought the user up to a master menu when the system was turned on.

I also sent out a lot of letters to correct various bugs that were found by an ever increasing number of users and tried to include in my planned modifications their suggestions for some new routines that made the program easier to use.

RONDURE COMPANY

7522 BUTLER ST
DALLAS, TEXAS 75235
214-630-4621

↙191

the computer room

SPECIAL
Printer for your Microcomputer

GE TERMINET 300 PRINTER

Pin feed—9" paper

- 80 Print positions
- Receive only
- ASCII code
- RS-232 interface
- 30 CPS
- Upper & lowercase
- Shipping wt. 75#

Shipping containers $15.00
(used)
(good working condition)

Will run on serial RS232 port of most micros including TRS-80. **$450.00**

WE HAVE FLAT-PACK ACOUSTIC

Modem pickup

$19.50

USED FANS

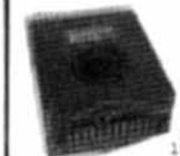
Muffin—8.00
Sprite—4.00

NEW POWER SUPPLY

$25.00

5V at 3 Amp
12V at 6 Amp
-12V at 3 Amp

USED POWER SUPPLY



$15.00

5V at 12 Amp
15V at 6 Amp
6V at 2 Amp

ASCII Keyboard

(used)
with enclosure

SALE $15.00

MICRO SWITCH KEYBOARD
USED BUT LOOKS VERY NICE

ASC II **$40.00**
(With Print)

USED OMNITEK

ORIG.
ONLY
95.00

ORDERING INFORMATION:
We ship the same day we receive a certified check or money order. Texas residents add 5% sales tax. Please call if you have a question. Write for our CATALOG of many parts, terminals, printers, etc. All items subject to availability. Your money returned if we are out of stock.

SHIPPING INFORMATION:
Modems: $2.50 each; Key Boards $3.50, Power Supply $5.00. Large items & Parts: Specify Freight or Air Freight Collect. Foreign Orders: Add appropriate freight or postage. We now take Master Charge and Visa orders. Specify full number, bank number and expiration date.

Technology plus Corporation

40-PIN EDGE CONNECTORS
(wirewrap style)

For Your TRS-80* or Superbrain.

$3.95—Nickel plated phosphor bronze contacts.

$7.95—Gold plated contacts.

OR—

Try our 3D40 edge card for your TRS-80*. The 3D40 is a precision double sided P.C. board with 40-pin connector in front, wired to 3 16-pin dip sockets in back. Use the 3D40 with dip ribbon cables and solderless breadboards to make TRS-80 interfacing experiments a snap! $15.95

ALSO—Coming soon for your TRS-80*

THE SCHEDULER

An inexpensive system of hardware and software that provides a real time clock and wireless remote control of 256 external devices for your TRS-80.

* TRS-80 is a trademark of the Radio Shack Division of Tandy Corporation.

P.O. Box 566
Westfield, MA.
01085

↙208

Disk Based Word Processor $7.50

A complete word processing system for your TRS-80 including full editing features such as paragraph move, line deletion, insertion & line correction. Store text on disk, print business and personal letters, reports with numbered pages and title pages! Text stored on disk as blocks are created so texts are not limited by the available memory. The PENSA-WRITE word processor is for 16K single-drive TRS-80s and comes complete with software to produce upper/lower case at printhead, and keyboard reverse. Full right/left justification and much more. Send cheque, money order or order by phone, 24 hours, 7 days a week. Mastercharge and Visa welcome. Manual only, $1.75-deductible from cost of software.

ORDER NOW! !

PENSADYNE
COMPUTER SERVICES
4441 WEST FIRST AVE.
VANCOUVER, B.C., V6R 4H9
604-224-3107

↙207

---

accounts payable TWICE as fast! with

the Spooler for your TRS-80*!

We have benchmarked voucher check processing and found it to run nearly twice as fast with **THE SPOOLER** installed.

THE SPOOLER requires no modification to your user application program.

Price: $100. Add $4. for credit card orders.
Add $5. for C.O.D. orders
Documentation available separately for $10.
Model II version now available.

Phone (513) 891-4496 or write:

Snapp Inc.

↙232

[illegible] Corporate Park Drive
Cincinnati, Ohio 45242

* TRS-80 is a trademark of the Radio Shack Division of Tandy Corporation.

In April, 1979, I informed my customers that a modification to the program was available at a nominal cost to cover shipping and the disk. The modification was very well received. The program had finally reached the professional level that I had hoped for.

Probably the most significant reason for the program's success is that I spent a great deal of time planning the type of information that needed to be stored and the manner in which it would be updated. The primary objective of the program was to improve the profitability of my business. All of my planning was directed toward this goal.

I made sure that each piece of information that was retained had a valid use in improving my business efficiency. I used machine language packing routines where necessary to store more information than could be handled with BASIC. To give you an idea of the packing density, the following information is stored on each part: part number (8 alphanumeric characters), description (15 alphanumeric chrs.), list price, sale price, regular cost, promotional cost, quantity on hand, quantity on order (two orders), items per box, reorder point, accumulated sales year to date and previous year, vendor code and three dates pertaining to items on sale. Even with all this information stored, I am able to get almost 5000 part numbers on a Model 1 TRS-80 system.

Good Programming

Storing information compactly is only part of the secret of good programming. Equally important is the manner in which it is updated. The program has to be easy to use by relatively untrained individuals and it has to save time. In most cases the owner or manager of a small business doubles as the data processing department and the entire justification for purchasing the computer is to free up more of his valuable time.

For this reason I chose point of sale as my primary means of updating inventory levels, with batch entry available when necessary. With a point of sale system, all of the following can be accomplished with a single entry, taking essentially the same time as writing a sales slip or invoice.

1. The inventory level is updated immediately. Therefore, stock levels are always current.

2. The descriptions and price match the product in hand, reducing the number of incorrect entries.

3. The computer calculates the extended cost, subtotal, sales tax and change, reducing mathematical errors.

4. The computer stores information on income, cost of sales and percent profit by category and reports the figures on a day, month and year basis eliminating the need for a cash register.

Documentation is probably the weakest area of small business programs that I have seen.

Additionally, the information is much more detailed.

Frequently I'm asked if point of sale is practical on a microcomputer and my answer is definitely, yes. By using a special machine-language search routine, I can access any part in the file in four to six seconds. Also, the computer can generate an invoice as rapidly as by hand.

This brings up another important point. Programs should be written in the normal high-level language of the computer, in this case BASIC, but employ machine-language subroutines wherever time or density is important. This allows you to modify most of the program, but gives you the flexibility of machine language.

When programming, don't neglect the human factor. An otherwise good program may be rejected because of operator frustration. Let me give some examples:

INKEY Routines written in BASIC. These routines which allow flashy screen formatting are great for demonstration, but I have yet to find one that is fast enough to keep up with a good typist. I used the normal input statement in my original program to avoid this problem.

Repetitive entries of the same information. I reviewed an accounts receivable program recently in which it was necessary to enter the date of the transaction for each entry. I suggested that the day's date be automatically entered with an option to change it. Multiple entries become highly frustrating with this method.

Methods of Updating

My initial program had a general change portion that updated the "part" information. I now have four separate methods to update: One for price changes; one for stock adjustment, one for sale items and the original general change.

Test your program in the real world and be prepared to modify it to make it easy for an operater to use. No one finds bugs faster than an inexperienced operator in an actual situation.

The likelihood of an inexperienced operator brings us to the next area of importance—documentation. Documentation is probably the weakest area of small business programs that I have seen. Generally, programmers want to write programs, not documentation. They tend to spend little time on it and direct it toward other programmers, not the novice. The following information is a must in any business program documentation:

1. An index.

2. A glossary of special terms including limits of fields used.

3. A description of the various program modules and how they interact with the stored information.

4. A step by step practice section that leads the user by the hand through all of the major portions of the program.

5. Instructions on how to make common modifications.

Assume that the user knows nothing about a computer and cover every detail. Most businessmen will not have read the disk operating system manual, the BASIC manual or anything else. These manuals are too technical and he doesn't want to waste time on them.

Forget flow charts for the same reason. I eliminated them from my second manual and no one's requested them. They confuse most non-programmers.

Once you have completed your program and documentation, copyright it. For additional protection, I also use a license agreement. My license form has space on it to keep track of the modifications and corrections that I have sent to the licensee, and undoubtedly, you will have corrections and modifications.

Even if you were to write a bug-free program, the dynamics of business and the computer industry will generate a need for changes and modifications.

For example, I have sent out mods for DOS 2.3 and have a version available for NEWDOS. I also added special routines based on user feedback such as a recently completed price tag printing routine.

Word of mouth advertising and recommendations are very important in this still small industry and your reputation will suffer if you do not correct your bugs, keep your program current and answer user questions. Be sure to price your program high enough to cover some of these extra costs, along with shipping, manual printing and some return phone calls. If you plan to use a dealer network, leave sufficient margin in your pricing for their markup. A dealer needs at least a 30 percent margin to cover his costs of advertising, customer support etc.

I hope that I have been able to give you some insights into the problems of developing a business program. There is a tremendous market potential out there, and if you can solve a business' problems, there is also great reward. ■

If you want fast and cheap storage, read on.

Disk Alternatives

Harley Dyk
1644 Grant
Grand Haven MI 49417

Most owners of TRS-80 16K Level II cassette based systems dream about disks and I am no exception. However, one must purchase the expansion interface, the disk, and more memory to fully utilize the system, and this costs nearly $1000. Some people seriously question spending this kind of money for only 79K of storage on line.

The Stringy Floppy by Exatron, and the BETA-80 by Meca, both California based companies provide alternative investments. Neither require the expansion interface and both will interest the hobbyist and small businessman.

ESF

When I purchased the Exatron Stringy Floppy (ESF), shipment was prompt and the unit worked well from the beginning. The system is designed for the non-hardware type programmer.

After plugging the ribbon cable into the keyboard and turning on the computer, you type SYSTEM, then /12345 and you are "talking" to the ESF. The ESF uses a continuous loop tape wafer which comes in 5, 10, 20, 50 and 75-ft. lengths and can store about 4K, 8K, 16K, 40K and 60K bytes respectively.

The ESF for the TRS-80 is mechanically simple. The tape moves at a constant speed of 10 in./sec. in one direction and is on only when the ESF is reading, writing or searching. These wafers seem quite reliable. I have had one wafer jam out of twenty-five and it was promptly replaced. They are available only from Exatron, which is the only negative point I can give the ESF system.

The speed of the ESF is impressive with an actual read/write speed of 800 bytes/sec. I use an 8" floppy when teaching and it loads BASIC programs at about 1300 bytes/sec. I timed the ESF and loaded a 14K program in eighteen seconds. The same program took four minutes with audio cassette.

The operating system resides in EPROM, but does not include data handling capabilities. Data capabilities are provided on tape taking up about 1K of RAM, leaving about 14.5K for the user.

The data handling capabilities are also very impressive. I was able to load the data for Adventure Land (a 21-minute task for a Radio Shack cassette) in 41 seconds, over 30 times as fast. Data is dumped onto tape in 256 byte blocks. As a test of the ESF's speed, I dumped 500 three digit numbers in ten seconds. It took fifteen seconds to read the numbers back and print them on the CRT.

Machine language programs can also be placed on the wafers. I place Space War & Sargon on wafer with a minimum of difficulty. Machine language and BASIC programs are saved on the wafer by number and one must maintain a good record of what is on each wafer or one can wipe out existing programs.

The ESF seems very reliable, each program is checked each time it is saved by an automatic verification feature. This increases save time since the tape must cycle around yielding a save time of 40-60 seconds for a 14K program on a 20-ft. wafer, and 80-115 seconds for the same program on a 50-ft. wafer. But compare that to over eight minutes to save and verify such a program on audio cassette! The verification feature can be bypassed by hitting the break key after the write light has gone off.

Beta-80

Meca sent me a BETA-80 to evaluate for this article. I have used it for about a month and have a fair knowledge of its capabilities. The unit plugs into the keyboard as does the ESF and with the system tape (digital cassette) installed, you merely turn on the power. The tape rewinds, loads the operating system, and reads the directory in about 20 seconds. The operating system resides in RAM and takes about 3½ K and the directory takes about ½K, so one is

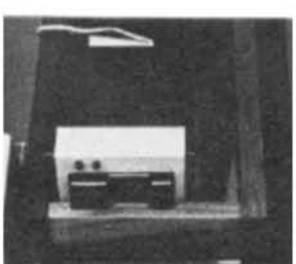


Photo 1. Exatron Stringy Floppy.

Photo 2. The TRS-80 16K Level II—BETA-80 turnkey system (the BETA-80 labels were on back order at the time of this photo).

left with 11719 bytes in a 16K system.

With some sacrifice in speed the BETA-80 offers you much of the convenience of a disk. As Meca advertises, the BETA-80 has six-seconds access to 80K, while the Radio Shack disk has two-seconds access, but the BETA-80 gives a full 250K per track with two tracks, worst case access time of 36 seconds per track with 10 seconds being average.

The actual data transfer rate on the BETA-80 is slower than the ESF with an advertised rate of 500 bytes/sec. The information is stored in 256 byte blocks with gaps in between giving an effective data rate of about 350 bytes/sec. Yet the BETA-80 can locate and load a 9K program (out of a 250K track) in an average time of 35 seconds. The BETA-80 has a read/write speed of 5 in./sec., but a search speed of 100 in./sec.

Once the cassette is inserted, it is totally software controlled. It sure is impressive to take all the software one has accumulated in ten months and put it on one cassette and be able to load any program at the stroke of a key.

Double density will be available for the BETA-80, making its actual transfer rate very close to that of the ESF. Meca recommends digital tape for the BETA-80, however, I have used top quality Maxell UDXL C-60 audio cassettes and have experienced no problems to date.

The BETA-80 can do a type of merge for programs, and it also can chain programs together using LOAD statements within the programs. Data is saved only by saving arrays and this works fine, except that one must convert strings to numeric data in BASIC before the string can be saved in an array. This slows down the saving of strings considerably.

The BETA-80 also saves machine language programs, although one must use a stand alone operating system in high memory. Since the operating systems in BASIC reside in the same area in which most machine language programs start, you cannot use the regular operating system to load such machine language programs. Meca recommends only the experienced assembly language programmers use the stand alone system, but this beginner was able to use it to transfer machine language programs from Radio Shack cassette to the BETA-80 format.

Comparisons

When comparing the two systems, you must keep in mind the price difference, which is substantial, and media handling differences.

The ESF requires frequent switching of wafers, but should be adequate for the hobbyist, average home computer enthusiast and some small businesses. If you want the luxury of the directory, universal media (digital quality cassette), lots of on line storage, and the proven Phi-deck, then the BETA-80 may be for you.

Tables 1 and 2 show a comparison of features and a summary of BASIC commands. Both units have performed very well for me. Both can give a very powerful system without an expansion interface or additional memory. Both units leave CLOAD & CSAVE totally functional, to your regular audio cassette can be used as a backup. This also allows you to transfer existing audio cassette software to the new media.

Face it, a disk is nice, but, quite frankly, not everyone needs the speed or price of a disk. However, most TRS-80 owners deserve more than audio cassette and such computerists should seriously consider the ESF and BETA-80. ■

Exatron now has available a speed-up kit for the TRS-80 (about $25) that doubles the speed of the computer and the ESF. Installing the kit means opening your computer and voiding the warranty, but the kit doubles the load and save speeds of the ESF, as well as the capacity of a wafer. This gives you, in effect, double density without touching the ESF: 1600 bytes/sec.

Beta

LOAD"	"	– load program (name up to 5 characters)
SAVE"	"	– save program
LOAD"	,R"	– load & run
LOAD"*	"	– load array
SAVE"*	"	– save array
MERGE"	"	– concatenate to BASIC program
MERGE"	,R"	– concatenate & run
NAME		– shows directory
KILL"	"	– kill program
KILL"*	"	– kill array

—plus 19 direct operating commands to mount tapes, initialize tapes, read & write blocks of memory, execute machine language programs, etc.

ESF

● NEW	– certifies & erases wafer (or part of wafer)
● SAVE X	– save BASIC or M.L. Program, where X is integer from 1-99
● LOAD X	– loads BASIC or M.L. Program, where X is integer from 1-99
● OPEN	– open data file
● CLOSE	– close data file
● PRINT	– writes data on wafer
● INPUT	– reads data from wafer

Table 2. BASIC Commands.

	ESF for TRS-80	Beta-80
Base Price	$249.50 Starter System @ $299.50, includes wafers, monitor, etc.	$499.00
Cost of Additional Drives	$199.50	$270.00
Drives per System	8	4
Tape	Wafer-continuous loop-1 direction travel-obtainable only from Exatron	Standard digital cassette
Dealers	None-mail order only	Yes – or mail order
Size	4¼ w x 6½d x 2⅝h	8⅝w x 10⅛d x 4¼h
Weight	Approx. 1 lb.	Approx. 6 lbs.
Read/Write Speed	800 bytes/sec., 10 in./sec., 720 bits/in.	500 bytes/sec., 5 in./sec., 800 bits/in.
Fast Search Speed	None	100 in./sec. both directions
Capacity per Drive	40-44K with 50 ft. wafer	500K with C-60 tape
Media Cost per M Byte Storage	$46 (23-50 ft. wafers)	$8.60
Compare to Cassette Speed	12.5 times faster at best than cassette	5.6 times faster at best than cassette
Warranty	Year/30 day money back	3 months mechanical 6 months electronic
Expansion Interface Needed	No	No
Programs/Unit	Max. 99 programs/wafer	Max. 126 programs/500K byte cassette
Total Software Control	Yes	Yes
Directory	No	Yes
Toll Free Number	Yes	No
Work with Level III BASIC	Yes	No
CLOAD & CSAVE Functional	Yes	Yes
Worst Case Across Time	60 seconds for 44K (50 ft. wafer)	36 seconds for 500K 10 second average
Verification	Automatic – can be bypassed	Automatic but can be turned off
Machine Lang. Cap.	Yes	Yes
Operating System	In EPROM	In RAM

Table 1. Comparison of features.

Try this inexpensive circuit, and meter your load level accurately.

CLOAD Micrometer

Bertram A. Thiel
159 West Main Street
Frostburg, MD 21532

The touchiness of the TRS-80 volume control setting is a persistent problem. If you are using commercial software and are frustrated with trial and error loading, try this simple hardware fix and save your hair-pulling for debugging.

Parts List
D1 – 1N34A or equivalent germanium diode
R1 – 5K trimmer resistor
M1 – 50 microampere DC meter, Lafayette 99-51146 or equivalent
P1 – 3.5mm mini-plug with cord
J1,2 – 3.5mm mini-jack, chassis mount

Parts List

Construction

The voltage level of a properly loading tape on my Level II machine is approximately 0.3 VAC, not very high, but with a sensitive enough meter the volume control of the tape recorder can be adjusted before CLOAD is entered. A sensitive meter will be in the microampere range, and all such meters are DC. We, however, are dealing with an AC signal from the tape deck.

The solution is to rectify the signal with a diode, but since silicone diodes only start to conduct at .7 volts, the older germanium diode that conducts at .3 volts will work best.

Did your eyebrow rise when you realized that the conduction voltage level of .3 volts is the same number as the .3 volts AC of the signal?

Don't worry. We have the square root of two working for us. Did your other eyebrow rise? Well then, just remember that AC is measured as RMS or root mean squared. To find the peak value, multiply the RMS value by 1.414 which gives us about .42 volts. This, minus the .3 volt detection level of the diode, leaves us with .12 volts to work with. The remaining voltage is fed to the meter and displayed (see Fig. 1.).

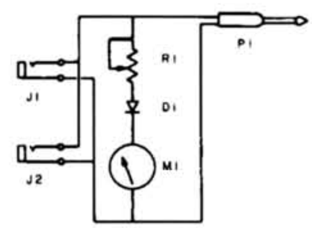


Fig. 1. CLOAD meter

The extra jack on the schematic is for an earphone if you wish to hear your CLOAD as well as see it.

Implementation

After you have constructed the project, initial adjustment is a snap. First, CLOAD a long program into your TRS-80, making sure it loads correctly. Disconnect the plug from the earphone jack of the tape deck and insert it in either of the meter jacks. Take the mini plug from the meter and insert it in the earphone jack. Now CLOAD the tape again and the meter should deflect when data begins to flow.

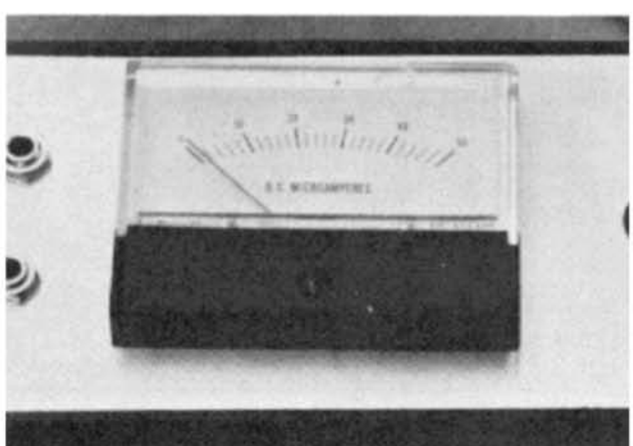


Photo 1. Finished project

Even if it doesn't, proceed to the next stage and turn the trimmer resister until the needle reads half scale or about 25 microamperes. As data is read the needle will rock back and forth; get used to this so that when you play a commercial tape you will know when the volume control of the tape deck is properly adjusted.

If the tape loads properly, plug in the earphone. The extra load on the output will change the voltage level but adjustment is now easy; just change the cassette volume control for the average reading of 25 microamperes. ■

Get a graphic display when CLOADing your Level I or II tapes.

CLOAD Monitor

Norman D. Whaland
430 E. 9 Street, Apt. 15
New York NY 10009

I've had no major problems with my Radio Shack TRS-80, except for one thing—loading tapes. Every tape I buy requires a different volume setting. Finding that setting often involves a half-hour of frustration.

The problem is this: the blinking stars that tell you the tape is loading are displayed only if the setting is correct at the very beginning of the file. Rewinding is needed for each new setting. Nor is that all. When the stars finally appear, BASIC may well respond with a WHAT?—indicating clearly that its little sign of encouragement was not to be construed as complete approval.

I have ended these problems. My Tape Monitor program displays continuous feedback on the quality of the input. In less than a minute I can find a volume level well within the acceptable range, so that I can be sure that it will work reliably.

The program was written for Level I, but it can be adapted for Level II with a few small modifications, as indicated on the listing. (No ROM subroutines are used.) I will assume Level I throughout most of the discussion. Except that the intervals between pulses are half as long, Level II tapes don't differ from Level I tapes in any way that affects use of the Monitor.

Tape Format

The Tape Monitor produces graphs that summarize the timing of the input signals. In most cases the guidelines at the end of this article will suffice for finding the correct volume setting. If something is wrong with a tape, however, you will want to know precisely what the graph means. Therefore we must first consider how data is represented on tape.

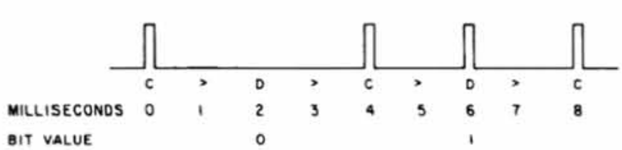


Fig. 1. The Level I tape code.

Fig. 1 shows what the bit sequence 01 looks like on a Level I tape. Clock pulses, marked with a *C* in the figure, appear at intervals of four milliseconds. Each clock pulse marks the beginning of a bit. Data pulses appear midway between the clock pulses at the positions marked with a *D*. The value of the bit is one or zero depending on whether the data pulse is present or absent.

When the cassette interface detects a pulse, it sets a flip-flop that the program can test. It remains set until the program resets it. The CLOAD routine reads the bits by testing and resetting the flip-flop on a precise schedule.

CLOAD first locates the clock pulse by repeatedly testing the input. The moment when the pulse appears is established as time zero of the read cycle. After a delay of one millisecond, the flip-flop is reset. (The times when the flip-flop is reset are marked with a > in Fig. 1.) After a further delay of two milliseconds, CLOAD tests the flip-flop and saves the result as the data bit. The flip-flop is immediately reset and the wait for the next clock pulse begins.

Thus, if a pulse is detected at any time from one to three milliseconds after the appearance of the clock pulse, the bit is read as a one; otherwise it is read as zero.

Clock and data pulses are physically identical. The CLOAD routine distinguishes them by their timing. Each file begins with a leader consisting of 1024 zero bits. Since all of the pulses in the leader are clock pulses, CLOAD is able to synchronize its timing with the tape.

All of the bits in a file are written as one continuous stream; byte boundaries aren't marked. To enable CLOAD to synchronize with the tape at the byte level, the special character A5H appears at the end of the leader. When CLOAD encounters this character, it begins assembling sequences of eight bits into bytes. The first four bytes specify the beginning and ending addresses of the area where the data will be stored. After them come the data bytes, followed by the checksum byte.

The two stars are displayed when the A5H character is encountered. Whenever a carriage-return character is read, the second star is removed or replaced.

If the A5H character isn't read properly, CLOAD continues reading a bit at a time until it finds one. Since A5H bytes don't normally occur in BASIC programs, what CLOAD finds will actually be the end of one byte combined with the start of the next. The stars appear, but don't blink very often, because the carriage-return characters are read out of synchronization and aren't recognized.

A particularly unpleasant consequence of missing the A5H character is that the addresses eventually used to load the data are incorrect. As a result, the areas of RAM used by BASIC could be destroyed. That's why it's a good idea to turn power off when CLOAD fails; some of the data areas aren't initialized when you push the RESET button. However, if you stop the recorder before the stars appear, there is no need to power off or even push RESET. You can simply rewind, adjust the volume, and try again.

If the beginning of the file is read normally, synchronization may still be lost if errors occur farther on. Synchronization at the bit level is restored as soon as a zero bit is read, but synchronization at the byte level may be lost permanently. When this happens, the stars stop flashing.

It's also possible for errors to occur without loss of synchronization. In that case, the checksum computed by CLOAD doesn't match the checksum on the tape, and the WHAT? message is displayed.

It's obvious what happens if the volume control is set too low. Either the pulses can't be read at all, or pulses are missed occasionally. Usually, the latter occurs only in a very narrow range of volume.

What happens at too high a volume came as a surprise to me. I assumed that noise on the tape would show up as spurious pulses. This does happen sometimes, but the usual problem is that the pulses lengthen as the volume increases. When a pulse continues to give an input signal for more than a millisecond, it is read twice. A data pulse, for example, may be read once as a data pulse and again as a clock pulse.

It might seem that byte synchronization would surely be lost when the volume is too high. My experiments with the Tape Monitor have shown that this isn't so. If the volume is just a little too high, errors due to long pulses are sporadic. Bit synchronization is lost when a data pulse is too long, but complete synchronization is restored if no further error occurs before a zero bit is reached. That bit is converted to a one. The stars flash normally right to the end, but the WHAT? message appears.

When the volume is too low, on the other hand, byte synchronization is almost sure to be lost. As soon as a clock pulse is missed, synchronization is lost and not regained.

Program Description

The Tape Monitor reads bits in groups of 255. At the end of each group, it displays the timing of the pulses as a bar graph. Each graph remains on the screen for approximately one second, while the next group of bits is being read. Some typical graphs are shown in Fig. 2. Their significance will be discussed in the next section.

Time is represented horizontally on the screen and is reckoned from the moment the clock pulse is first detected. Each bar in the graph represents an interval of 0.1 millisecond. The length of the bar represents the number of bits for which a pulse was detected during the interval.

The vertical scale is logarithmic, using base 2. A bar of length one indicates a count of one; a bar of length two indicates a count of 2-3; a bar of length three, a count of 4-7 and so on.

Processing centers around an array of 64 counters, one for each bar in the graph. (In the Level II version, 32 counters are used.) The counters are addressed through the HL register. When thinking about timings, it is convenient to number the counters from one to 64, indicating how many tenths of a millisecond have elapsed since the clock pulse began. For programming reasons, the corresponding L register values are two to 65.

When a clock pulse hasn't been detected by the time counter 63 is reached, counter 64 is incremented. Thus, the last bar in the graph shows how many times the interval between clock pulses was too long to display.

The program has four sections. NEW clears the counters and waits for a zero bit, in order to establish synchronization. READ reads the tape and tallies its findings in the counters. GRAPH creates the bar graph. BREAK handles input from the keyboard, as described in the operating instructions.

Most of the program is quite straightforward and needs no explanation beyond the comments in the listing. The READ routine, however, is a bit tortuous, because of the stringent timing requirements. (I ended up with only 13 cycles to spare.)

READ is a loop, executed once every 170 machine cycles or approximately once every 0.1 millisecond. At the beginning of the loop, the status of the input flip-flop is transferred to the carry flag, and the flip-flop is reset if necessary. After that, the course of processing depends on the carry flag and on the contents of the L register. The carry flag remains undisturbed for several instructions, until it is finally altered by a CP instruction.

If L lies between one and 63, inclusive, it is incremented. If a pulse was detected, the counter that HL then addresses is incremented. In addition, if the new value of L is greater than 32, the pulse is assumed to be a clock pulse, and L is set to one.

If L contains 64, it is incremented, and the last counter is incremented whether or not a pulse was detected. If a pulse was detected, L is set to one.

If L contains 65 and no pulse was detected, both L and the counters are left unchanged. If a pulse was detected, L is set to

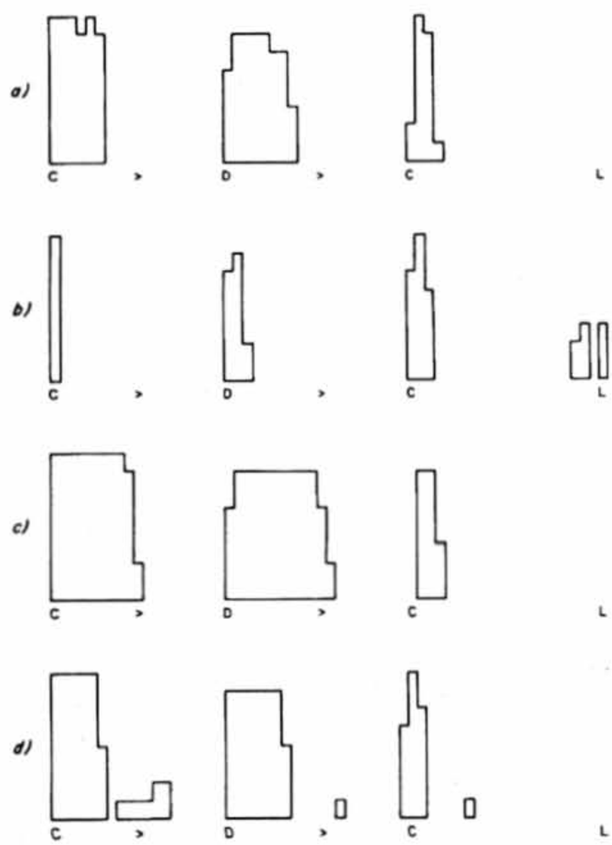


Fig. 2. Graphs displayed by the Tape Monitor. (a) acceptable volume. (b) Volume too low. (c) Volume too high. (d) Stray pulses.

one. The **L** register contains 65 when READ is entered from NEW.

Whenever L is set to one, the completion of the bit is tallied in register D. The routine branches to GRAPH when 255 bits have been processed.

At the end of each pass through the loop, the routine tests the BREAK key and branches to BREAK if it is depressed.

Using the Program

The program is entered at location 4E00H. It turns on the cassette at once and begins

Program Listing.: Alternate instructions for Level II are marked with an asterisk.

```
4EØØ   31 ØØ 4F     START  LD   SP,4FØØH
4EØ3   3E Ø4               LD   A,Ø4H     Turn on cassette
4EØ5   D3 FF               OUT  FFH,A
4EØ7   21 ØØ 4F     NEW    LD   HL,4FØØH  Clear pulse counters
4EØA   Ø6 41               LD   B,41H
*4EØA  Ø6 21               LD   B,21H          *** Level II
4EØC   23           NEW1   INC  HL
4EØD   36 ØØ               LD   (HL),ØØH
4EØF   1Ø FB               DJNZ NEW1
4E11   16 FF               LD   D,FFH     Bit counter := 255
4E13   DB FF        NEW2   IN   A,FFH     Wait for a pulse
4E15   Ø7                  RLCA
4E16   3Ø FB               JR   NC,NEW2
4E18   Ø6 DØ               LD   B,DØH     Timing delay
*4E18  Ø6 68               LD   B,68H          *** Level II
4E1A   1Ø FE               DJNZ $
4E1C   3E Ø4               LD   A,Ø4H     Reset input flip-flop
4E1E   D3 FF               OUT  FFH,A
4E2Ø   Ø6 DØ               LD   B,DØH     Timing delay
*4E2Ø  Ø6 68               LD   B,68H          *** Level II
4E22   1Ø FE               DJNZ $
4E24   DB FF               IN   A,FFH     Move input to carry
4E26   Ø7                  RLCA
4E27   3E Ø4               LD   A,Ø4H     Reset input flip-flop
4E29   D3 FF               OUT  FFH,A
4E28   38 E6               JR   C,NEW2    Repeat if not a Ø bit
4E2D   DB FF        READ   IN   A,FFH     Move input to carry
4E2F   Ø7                  RLCA
4E3Ø   3Ø 18               JR   NC,READ1  If input is present,
4E32   3E Ø4               LD   A,Ø4H       reset flip-flop
4E34   D3 FF               OUT  FFH,A
4E36   CB 75        READ2  BIT  6,L       Long interval?
*4E36  CB 6D        READ2  BIT  5,L            *** Level II
4E38   2Ø 12               JR   NZ,READ3  Yes: branch
4E3A   2C                  INC  L         No: next counter
4E38   3Ø 19               JR   NC,TIME3  Through if no pulse
4E3D   34                  INC  (HL)      Tally pulse
4E3E   3E 2Ø               LD   A,2ØH     If it is a clock pulse,
*4E3E  3E 1Ø               LD   A,1ØH          *** Level II
4E4Ø   BD                  CP   L
4E41   3Ø 19               JR   NC,TIME2
4E43   2E Ø1        READ5  LD   L,Ø1H       go to first counter
4E45   15                  DEC  D           tally bit
4E46   2Ø 18               JR   NZ,TIME1  If bit counter is Ø,
4E48   18 33               JR   GRAPH       display graph
4E4A   18 EA        READ1  JR   READ2     Adjust timing
4E4C   CB 45        READ3  BIT  Ø,L       Interval tallied?
4E4E   2Ø Ø2               JR   NZ,READ4
4E5Ø   2C                  INC  L         No: tally interval
4E51   34                  INC  (HL)
4E52   3Ø ØC        READ4  JR   NC,TIME1  If pulse present,
4E54   18 ED               JR   READ5       go to first counter
4E56   FD 2A ØØ ØØ  TIME3  LD   IY,(ØØØØH) Timing delay
4E5A   FD 23               INC  IY
4E5C   ED 5F        TIME2  LD   A,R       Timing delay
4E5E   ED 5F               LD   A,R
4E6Ø   3A 4Ø 38     TIME1  LD   A,(384ØH) If BREAK key pressed,
4E63   FE Ø4               CP   Ø4H
4E65   28 Ø5               JR   Z,BREAK     exit loop
4E67   3A ØØ ØØ            LD   A,(ØØØØH) Timing delay
4E6A   18 C1               JR   READ
4E6C   3A 4Ø 38     BREAK  LD   A,(384ØH) Wait for key press
4E6F   FE Ø1               CP   Ø1H       If ENTER key,
4E71   28 94               JR   Z,NEW       resume scanning
4E73   FE Ø2               CP   Ø2H       If CLEAR key,
4E75   2Ø F5               JR   NZ,BREAK
4E77   AF                  XOR  A           turn off cassette
4E78   D3 FF               OUT  FFH,A
4E7A   C3 66 ØØ            JP   ØØ66H       system reset
4E70   Ø1 FF Ø3     GRAPH  LD   BC,Ø3FFH  Clear screen
4E8Ø   11 Ø1 3C            LD   DE,3CØ1H
4E83   21 ØØ 3C            LD   HL,3CØØH
4E86   36 2Ø               LD   (HL),2ØH
4E88   ED BØ               LDIR
4E8A   3E 43               LD   A,43H     Display C's
4E8C   32 CØ 3F            LD   (3FCØH),A
4E8F   32 E7 3F            LD   (3FE7H),A
4E92   3E 3E               LD   A,3EH     Display >'s
4E94   32 C9 3F            LD   (3FC9H),A
4E97   32 DD 3F            LD   (3FDDH),A
4E9A   3E 44               LD   A,44H     Display D
4E9C   32 D3 3F            LD   (3FD3H),A
4E9F   3E 4C               LD   A,4CH     Display L
4EA1   32 FF 3F            LD   (3FFFH),A
4EA4   Ø1 CØ FF            LD   BC,FFCØH  Load -64
4EA7   11 Ø2 4F            LD   DE,4FØ2H  Load counter pointer
4EAA   21 CØ 3F            LD   HL,3FCØH  Load screen pointer
4EAD   1A           GRPH3  LD   A,(DE)    If counter = Ø,
4EAE   B7                  OR   A
4EAF   28 Ø9               JR   Z,GRPH1     omit write loop
4EB1   E5                  PUSH HL        Save screen pointer
4EB2   Ø9           GRPH2  ADD  HL,BC     Raise screen pointer
4EB3   36 BF               LD   (HL),BFH  Display rectangle
4EB5   CB 3F               SRL  A         Divide counter by 2
4EB7   2Ø F9               JR   NZ,GRPH2  Loop if counter ≠ Ø
4EB9   E1                  POP  HL        Restore screen pointer
4EBA   13           GRPH1  INC  DE        Next counter
4E8B   23                  INC  HL        Next bar position
4E8C   ØØ                  NOP
*4E8C  23                  INC  HL             *** Level II
4EBD   7B                  LD   A,E       Last counter?
4EBE   FE 42               CP   42H
*4EBE  FE 22               CP   22H            *** Level II
4ECØ   38 EB               JR   C,GRPH3   No: repeat
4EC2   C3 Ø7 4E            JP   NEW       Yes: initialize
```

displaying graphs as soon as it finds information on the tape. The first graph produced by the file leader has an irregular appearance, presumably because CSAVE starts to write before the tape reaches running speed. It is normal for the data pulse to be missing in the next couple of graphs.

A graph can be held on the screen for close inspection by pressing the BREAK key. The tape continues to move, but it isn't read by the program. Press ENTER to resume the displays, or press CLEAR to return to BASIC. The program returns to the address accessed by the RESET button. Since the Tape Monitor doesn't actually load any data, there is no need for a power-on reset.

Fig. 2a shows what the graphs look like when the volume is properly adjusted. The pulses don't overlap the markers, and there are no stray pulses. As in Fig. 1, *C* and *D* mark the position of the clock and data pulses respectively. The clock pulses are split, with their first appearance shown at the right and the continuation shown at the left. It is normal for the portion at the right to be three bars wide, because of timing irregularities in CSAVE and in the Tape Monitor.

If the volume setting is too low, graphs like Fig. 2b will result. The bars at the extreme right indicate that pulses were missed, resulting in long intervals between pulses. The bar marked with *L* indicates intervals too long to graph, undoubtedly caused by the loss of a clock pulse between two zero bits.

When the volume is too high, the pulses spread beyond the markers, as shown in Fig. 2c. At still higher volume, the program may lose synchronization, resulting in a sudden deterioration of the graphs. Figs. 2a-c were all prepared from the same tape.

Stray pulses produce a graph like Fig. 2d. This is a graph from a program recorded over another program and read at a volume much higher than I usually use. At lower volume the stray pulses don't appear. Although Radio Shack recommends using a bulk eraser on tapes before re-using them, I routinely write revised programs over the original. This practice has caused no difficulty at low volume.

There is no need for the volume to be adjusted so that the pulses are as narrow as possible. Some tapes have sporadic episodes of missed pulses, unless the pulses are rather wide. A one-interval gap between the pulses and the markers should be quite sufficient. A lower volume is necessary only when stray pulses may be present.

The graphs produced by the Level II version have 32 bar positions, spaced out across the screen so that the markers have the same significance. As in the Level I version, each bar represents an interval of 0.1 millisecond.

A Level I tape monitored with the Level II version can be recognized by the absence of data pulses and the presence of a long bar over the *L* marker. It is normal to see no data bar when reading zeros, but in that case there should be no bar over the *L* marker.

If you don't get a graph at a reasonably high volume, you may be scanning a Level II tape with the Level I version. To test for this possibility, slowly increase the volume from zero. This procedure should produce a graph if there is anything at all on the tape. If it is a Level II tape, pulses will appear over the markers as well as over the *C* and *D* markers.

Conclusion

Besides making it easier to adjust the volume, the Tape Monitor may be useful for other purposes, such as the design of tape input-output routines and the assessment of tape quality. Certainly, the process of reading tape seems less mysterious when you can see the actual input. ■

An algorithm for searching your data base character by character!

Free Format Search

Henry G. Riekers
208 Phelps Ave.
Glen Burnie MD 21061

This article describes one of the most convenient subroutines that I have written. It allows users of BASIC to search through a free-formatted data base to identify items of interest and to apply these items to a variety of tasks. Example 1 shows only one of a wide variety of applications of this algorithm.

The Algorithm

The algorithm accepts any string of alphanumerics and then searches through the entire data base, character-by-character, in an attempt to make a match. If a match is achieved, then the data line will be printed.

The algorithm is constructed so that it will search through data lines of up to 70 characters. If the data lines are shorter, then you should modify lines 50 and 80 accordingly. If you are dealing with a large data base and matching strings of more than just a few characters, you should rewrite lines 50 and 80 such that the length "x" of the string is subtracted from 70.

In Example 1, I have constructed a data base of my home library. The data base consists of a call number for the book, the title and the author. Obviously additional in-

```
5 CLS
10 INPUT A$
20 X=LEN(A$)
30 RESTORE
40 READ W$
43 IF W$="Z"THENPRINT:INPUT"CONTINUE";P$
45 IF W$="Z" THEN GOTO 5
50 FOR N=1 TO 70
60 IF A$=MID$(W$,N,X) THEN 100
70 NEXT N
80 IF N>70 THEN 40
100 PRINTW$
110 GOTO 40
1000 DATA 529.19 PHYSICS HAUSMANN SLACK
1005 DATA 529.33 ELEMENTARY PHYSICS KRICHT
1010 DATA 726.2  ELECTRONICS WEIDNER SELLS
9999 DATA Z
```

Data Search algorithm.

Example 1. Library application.

```
?SLACK
529.19 PHYSICS HAUSMANN SLACK

CONTINUE ?
?PHYSICS
529.19 PHYSICS HAUSMANN SLACK
529.33 ELEMENTARY PHYSICS KRICHT

CONTINUE ?
?ELECTRONICS
726.2  ELECTRONICS WEIDNER SELLS

CONTINUE ?
?ELECTRONIC
726.2  ELECTRONICS WEIDNER SELLS

CONTINUE ?
?529.19
529.19 PHYSICS HAUSMANN SLACK

CONTINUE ?
?529
529.19 PHYSICS HAUSMANN SLACK
529.33 ELEMENTARY PHYSICS KRICHT

CONTINUE ?
```

```
?PHYSICS
529.19 PHYSICS HAUSMANN SLACK
529.33 ELEMENTARY PHYSICS KRICHT

CONTINUE ?
?PHY
529.19 PHYSICS HAUSMANN SLACK
529.33 ELEMENTARY PHYSICS KRICHT



CONTINUE ?
?S
529.19 PHYSICS HAUSMANN SLACK
529.33 ELEMENTARY PHYSICS KRICHT
726.2  ELECTRONICS WEIDNER SELLS

CONTINUE ?
```

formation, such as a "key word" list, could be added. When the algorithm is run, a question mark appears at the top of the CRT. You then enter the string you are interested in matching.

The first entry shows the author's name, SLACK, used to locate the book, *Physics*. In the second entry the title, Electronics, is used to find that book. As a further extension of the second entry, the word Electronic (note—no "s" at the end) is used and still locates the book correctly. In the next entry the call number 529.19 is used to identify the book, *Physics*. If the call number 529 is used without the decimal portion, then all the 529 series of books will be printed out. Similarly, with the word PHYSICS and PHY, the same results can be obtained. Finally, if the letter S is input, then each data line with that letter in it will be printed out.

This implementation was done on a TRS-80 Level II machine. For large data bases the execution time can become as long as a minute or more, so don't put any more in your data base than absolutely necessary. There are several ways to make the algorithm more efficient, and I will leave that to the reader's ingenuity. ■

Use this AL program to backup your 500 baud tapes and display the contents in both ASCII and Hex.

Backup/Display

Craig A. Lindley
320 W. Alameda # 203
Burbank CA 91506

Have you ever had an expensive piece of software eaten by a hungry tape deck?

Have you ever accidentally written over your most popular software package?

Have you ever had a cassette tape develop a short dropout that caused the program you have been working with for a month and a half to bomb continually?

If your answer to any to the above questions is yes, then you know the frustrations that those episodes can cause. Either you send your damaged tape back to the place where you bought it, or you swallow your pride, dig deep into your pocket and purchase the same software package all over again. (The editor/assembler for $30.00 Yek!)

Having experienced this numerous times I've developed a rule to live by: Never run expensive software longer than necessary to make a backup copy. From that day forth always run with the backup copy until it dies. When it does, just generate a new one.

Be warned, however, that a program used to make backup tapes can also be used to pirate other people's software. As an author of software myself, I in no way condone this action. Anyone caught distributing software "under the table" so to speak, should be stopped. It is not fair to the people who worked so hard to write it.

The Program

The Backup-Display program was originally intended to make backup copies of any TRS-80 Level II - 500 baud tape, including BASIC, source code or object code formats. Since then an additional function has been added to display all the data on a given tape on the screen and/or line printer in both ASCII and hex codes. This allows the tape format used in the TRS-80 to be studied closely.

The ASCII-hex display also allows one to ascertain the program type, the program name, starting address and entry point. Thus one can identify what type of tape it is without actually loading it.

The results of my effort are shown in the listings. The message located at the start of the listing indicates that the program contains all the user prompts necessary for ease of operation. In other words, the program contains quite a few bells and whistles for those of us who like that sort of thing.

For convenience, the program also includes a menu of selectable options, line printer output for the memory dump and a built-in leader delay to jump over any tape leader when making a copy or separate multiple copies on the same tape.

Using the Program

SYSTEM loads the object tape. Type BACKUP at the first asterisk prompt. After loading, type a / [enter] to execute.

After the initial messages, the computer will wait for you to load the tape you want copied or examined. Put the tape into the cassette deck and press enter. The double asterisks will appear in the upper right hand corner of the screen and will blink (quite irregularly) as the tape is being copied into memory. When the tape is completely loaded in, the program will then display the menu of all the selectable options and await a response. Your choices are:

R—RETURN TO BASIC
M—LOAD NEW MASTER TAPE
C—MAKE ANOTHER COPY
S—SYMBOLIC TAPE DUMP

If any other choices are made than the ones listed, the menu will be repeated.

R returns the computer to BASIC just as if you turned the power on. M repeats the sequence of events just completed as if starting the program over.

C clears the screen and displays the message "When the tape is ready, press enter." This is your cue to record. (It is not necessary to run the tape past the leader as the program will do that for you.) When you press the enter key again, the tape copy will be made and then return to the menu. If you want another copy of the same cassette, select C again and follow the same procedure. You'll notice that as the tape is being recorded a single asterisk will flash in the right hand corner of the screen to indicate that a recording is being made.

The last selection on the menu is S, or symbolic memory dump. This displays on the screen the contents of the master tape just loaded in both ASCII and hex formats. (The printable ASCII characters are $21^H \rightarrow 7E^H$. Non printable characters are printed as blanks.)

If a printer is on-line, each complete screen (128 bytes) of data will be listed just as it appears on the screen. After each complete screen of data is written, the user has two choices, either hit any key (other than Break) to display the next 128 bytes of data or hit the Break key to return to the menu. After the last data is written on the screen, hitting *any* key will return you to the menu. At this time all of the options are again available.

Hints for Using the Backup/Display Program

When loading the backup copy into the TRS-80 using either the SYSTEM command for object tapes or CLOAD for BASIC tapes, use the same procedure as you would in loading the original. The copy is exactly the same as the original in every way including the program

```
1 ASCII- U B A C K U P <     J C O P Y R  TAPE     1 ASCII-   N     L           M  3 N   X     TAPE
   HEX- 55 42 41 43 4B 55 50 3C 00 00 4A 43 4F 50 59 52      HEX- CF 4E CD 15 4C 18 CD CD 9D 4D C3 33 4E 9D 78 C8
2 ASCII- I G H T   1 9 7 9 - - C L O C K  DUMP     2 ASCII- M                                      DUMP
   HEX- 49 47 48 54 20 31 39 37 39 2D 2D 43 4C 4F 43 4B      HEX- 4D
3 ASCII- W O R K S   S O F T W A R E - -  HIT      3 ASCII-                                        HIT
   HEX- 57 4F 52 4B 53 20 53 4F 46 54 57 41 52 45 2D 2D      HEX-
4 ASCII-   T A P E   B A C K U P / D I S  ANY      4 ASCII-                                        ANY
   HEX- 00 54 41 50 45 20 42 41 43 4B 55 50 2F 44 49 53      HEX-
5 ASCII- P L A Y   P R O G R A M   B Y    KEY      5 ASCII-                                        KEY
   HEX- 50 4C 41 59 20 50 52 4F 47 52 41 4D 20 42 59 20      HEX-
6 ASCII- C .   L I N D L E Y   T H I S    TO       6 ASCII-                                        TO
   HEX- 43 2E 20 4C 49 4E 44 4C 45 59 00 54 48 49 53 20      HEX-
7 ASCII- P R O G R A M   W I L L   C O P  ADV      7 ASCII-                                        ADV
   HEX- 50 52 4F 47 52 41 4D 20 57 49 4C 4C 20 43 4F 50      HEX-
8 ASCII- Y   A N Y   5 0 0   B A U D   T           8 ASCII-
   HEX- 59 20 41 4E 59 20 35 30 30 20 42 41 55 44 20 54      HEX-
```

Program Listing 2: An example of the first and last block of Backup/Display program, exactly like screen display.

name. The tape copy produced by this program is no better than the data it received. There is no error checking in this program whatsoever. If a bad load occurs from the original tape, the backup copy will be bad also.

When using this backup program it is necessary that the cassette you are copying have a short blank section (≈1.5s) immediately following the program on the tape. This blank area allows the backup program to correctly interpret the end of the tape you are trying to copy. If this blank area is missing, the program will consider any following data as valid and append it to the program you are trying to backup. This is of no consequence however, as the computer will read in only the backup copy and stop, ignoring all that additional garbage appended to the end of the tape.

Also, certain machine language tapes have a small loader at the beginning followed by the main program text. Each of these segments must be treated as individual programs and copied one after another on the same cassette.

An object code copy of this program can be had in one of two ways. The listing can be copied into the computer using the editor/assembler package and an assembly can be done, or the Backup/Display program can be purchased from Clockworks Software 320 W. Alameda, #203, Burbank, CA., 91506 for $12.95. I hope the program will save you one hundred times that amount in frustration and lost time, while allowing you to become more familiar with your TRS-80 tape format.■

Program Listing 1

```
4A00          00100         ORG     4A00H
              00110 ;BACKUP/DISPLAY PROGRAM
0029          00120 INPUT   EQU     2BH
000D          00130 CR      EQU     0DH
0001          00140 BREAK   EQU     01H
0020          00150 SUBST   EQU     ' '      ;PRINTED IN PLACE OF NONPRNT ASCII
00CD          00160 BCHR    EQU     0CDH     ;BLINK CHAR. FOR REC. AND PBLK.
0000          00170 BASIC   EQU     0000H
37E8          00180 LPRNT   EQU     37E8H    ;LINE PRINTER STATUS-DATA PORT
4A00 43       00190 CMES    DEFM    'COPYRIGHT 1979--CLOCKWORKS SOFTWARE--'
4A01 4F       4A02 50       4A03 59       4A04 52       4A05 49       4A06 47
   4A07 48       4A08 54       4A09 20       4A0A 31       4A0B 39       4A0C 3
7     4A0D 39       4A0E 2D       4A0F 2D       4A10 43       4A11 4C       4A
12 4F       4A13 43       4A14 4B       4A15 57       4A16 4F       4A17 52
 4A18 4B       4A19 53       4A1A 20       4A1B 53       4A1C 4F       4A1D 46
   4A1E 54       4A1F 57       4A20 41       4A21 52       4A22 45       4A23
 2D     4A24 2D       4A25 00       00200         DEFB    00H
4A26 54       00210 MES1    DEFM    'TAPE BACKUP/DISPLAY PROGRAM BY C. LINDLEY'
4A27 41       4A28 50       4A29 45       4A2A 20       4A2B 42       4A2C 41
   4A2D 43       4A2E 4B       4A2F 55       4A30 50       4A31 2F       4A32 4
4     4A33 49       4A34 53       4A35 50       4A36 4C       4A37 41       4A
38 59       4A39 20       4A3A 50       4A3B 52       4A3C 4F       4A3D 47
 4A3E 52       4A3F 41       4A40 4D       4A41 20       4A42 42       4A43 59
   4A44 20       4A45 43       4A46 2E       4A47 20       4A48 4C       4A49
 49     4A4A 4E       4A4B 44       4A4C 4C       4A4D 45       4A4E 59
4A4F 00       00220         DEFB    00H
4A50 54       00230 MES2    DEFM    'THIS PROGRAM WILL COPY ANY 500 BAUD TAPE'
4A51 48       4A52 49       4A53 53       4A54 20       4A55 50       4A56 52
   4A57 4F       4A58 47       4A59 52       4A5A 41       4A5B 4D       4A5C 2
0     4A5D 57       4A5E 49       4A5F 4C       4A60 4C       4A61 20       4A
62 43       4A63 4F       4A64 50       4A65 59       4A66 20       4A67 41
 4A68 4E       4A69 59       4A6A 20       4A6B 35       4A6C 30       4A6D 30
   4A6E 20       4A6F 42       4A70 41       4A71 55       4A72 44       4A73
 20     4A74 54       4A75 41       4A76 50       4A77 45       4A78 00
00240         DEFB    00H
4A79 57       00250 MES3    DEFM    'WHEN THE TAPE IS READY, PRESS ENTER'
4A7A 48       4A7B 45       4A7C 4E       4A7D 20       4A7E 54       4A7F 48
   4A80 45       4A81 20       4A82 54       4A83 41       4A84 50       4A85 4
5     4A86 20       4A87 49       4A88 53       4A89 20       4A8A 52       4A
8B 45       4A8C 41       4A8D 44       4A8E 59       4A8F 2C       4A90 20
 4A91 50       4A92 52       4A93 45       4A94 53       4A95 53       4A96 20
   4A97 45       4A98 4E       4A99 54       4A9A 45       4A9B 52       4A9C
 00     00260         DEFB    00H
4A9D 43       00270 MES4    DEFM    'COMPLETE TAPE COPIED INTO MEMORY'
4A9E 4F       4A9F 4D       4AA0 50       4AA1 4C       4AA2 45       4AA3 54
   4AA4 45       4AA5 20       4AA6 54       4AA7 41       4AA8 50       4AA9 4
```

Medoffice™

The state of the art in small systems medical office programs. Pascal software for the TRS-80*, PDP-11†, and many others.

From the medical software specialists:

PATIENT CARE DATA SYSTEMS
418 North Main St., Penn Yan, New York 14527
315-536-3734

*TRS-80 is a trademark of Radio Shack, a division of Tandy Corporation. ✓98
†PDP-11 is a trademark of Digital Equipment Corporation.

Computers & Gambling Magazine**

PRESENTS:
PROBABILITY HANDICAPPING DEVICE I
A TRS-80* LEVEL II BASIC PROGRAM FOR
HORSE RACE HANDICAPPING!

**"BE A WINNER: READ COMPUTERS & GAMBLING MAGAZINE SAMPLE ISSUES $1.00

This amazing program was written by a professional software consultant to TRW Space Systems, and is being introduced by the publishers of Computers & Gambling Magazine. "PHD-I" is a large complex level II program requiring a full 16K. It is carefully human factored for easy use. PHD-I is a comprehensive horse racing system for picking overlays in thoroughbred sprint races (less than 1 mile). This system has been tested using hundreds of races from the daily racing form for both major and minor tracks in the U.S. It consistently yields a **large positive return.** Detailed printouts of our statistics are available. This program features:

- Automatic keyboard debounce
- Verification display of each horse's parameters prior to entry for easy error correction
- The win probability and correct odds for each horse
- Bubble-sort routine for final display
- Line printer output option
- Complete users manual with examples and tips on betting and money management

Sit down with your TRS-80* and the daily racing form the night before the race and answer 5 or 6 questions about each horse's past performance. The computer then accurately predicts the win probability and odds-line for each horse allowing you to spot overlaid horses while at the track. The users manual contains a complete explanation of overlay betting.

ORDER NOW AT THIS SPECIAL INTRODUCTORY PRICE AND RECEIVE A COPY OF OUR MAGAZINE ABSOLUTELY FREE!
PHD-I 16k Level II Cassette **$19.95**

Make checks payable to:
Joe Compuder
22713 Ventura Boulevard, Suite F ✓193
Woodland Hills, Calif. 91364

*TRS-80 is a registered trademark of Tandy Corporation

— T R S 80 USERS —

Preserve — Protect — Display
your equipment with

CRYSTAL CLEAR PLASTIC COVERS

Special offer: Buy both covers & save

- Keyboard, interface & CRT ..... **$12.95 ea.**
- Line printer .... **$12.95 ea.**

[***Combination price $21.95***
plus $1.50 postage]

(Indiana residents add 4% sales tax)

Crown Plastic Co. ✓119
3746 N. College 317-925-5566
Indianapolis, IN 46225

---

announcing the ***new***

PC80 INSTRUMENTATION INTERFACE

designed for the TRS-80*

Applied Micro Technology introduces the first truly powerful instrumentation interface designed for use with TRS-80* microsystems.

The **PC80** connects directly to the Expansion Port Edge Card, and accommodates up to 4 of Applied Micro's Z-80* based special function circuit cards.

Circuit cards presently available are:

- **32-Channel A/D Converter**
- **8-Channel D/A Converter**
- **Programmable Clock**
- **Floating Point Arithmetic Processor**
- **2-Channel Serial I/O**
- **48-Bit Parallel I/O**

For complete information, contact:

APPLIED MICRO technology ✓236

* TRS-80 is a registered trademark of Tandy Corp.
* Z-80 is a registered trademark of Zilog Corp.

WE MEAN BUSINESS!
BUSINESS SOFTWARE, THAT IS

USE YOUR TRS-80* FOR MORE THAN FUN AND GAMES

THE DATA DUBBER **$49.95**
Duplicates any program tape to TRS-80 quality. Reconstructs data pulses to ensure accurate CLOADs. Permits easy loading of even poor quality commercial tapes without constant volume adjusting. Money back guarantee if not satisfied.

THE ELECTRIC SECRETARY **$75.00**
A powerful word processor to turn your TRS-80 into an automatic typewriter. Features page numbering, movable margins, headers, variable page length, and title centering. Enter text, revise, correct, and output to printer page formatted, justified, even hyphenated as required. Cross coupling files permits individually addressed form letters. Complete with upper/lower case conversion information on diskette. Specify if RS-232 adapter is installed in interface.

MAILROOM PLUS **$75.00**
A versatile and powerful mailing program to print labels by sequential coding: zip, city, state, customer ID code, even last name. Sorts by any code in minutes and stores sequentially in a single string (approx. 1500 records per diskette). Includes AUTOPRINT. Supplied on diskette.

MINIMAIL **$50.00**
A compact version of MAILROOM PLUS but without customer coding. Features alpha-lookahead for duplicates. Supplied on diskette.

FORMLET **$35.00**
Generates form letters from MINIMAIL records. Prepare your letter, bulletin, notice, advertisement, etc., then load the MINIMAIL files. Your printer will print the inside address, letter, and repeat for each name in the file—all properly spaced and justified. Supplied on cassette.

AUTOBOOT **$15.00**
Simplifies automatic BASIC program loading from your DOS. Permits sequencing through your choice of DOS commands, selects files and memory size you specify and loads or runs selected program. Allows user to see directory and free space before program runs automatically. Supplied on cassette.

SIR ECHO **$10.00**
A handy program to make your printer work like an electric typewriter. Use alone or merge with your programs to make what appears on the screen echo to the printer. Supplied on cassette.

TELEFON **$20.00**
Make your TRS-80 a smart terminal. Communicate with time-share and other computers, bulletin boards, etc. Transfer programs over the phone. For disk systems with modem.

UPPER/LOWER CASE CONVERSION **$20.00**
Reprint of KILOBAUD article explaining how to modify the TRS-80 to display both upper and lower case characters. Kit contains step-by-step instructions, parts, and necessary software on cassette for case reversal, echo, and automatic line feed routines.

User group discounts available
Dealer inquiries invited

*TRS-80 is a trademark of the Tandy Corp.

TERMS: Check, money order, Visa, Mastercharge. Washington residents add 5.3% for tax.

THE PERIPHERAL PEOPLE
P.O. Box 524, Dep't. M
Mercer Island, WA 98040
✓43 **(206) 232-4505**

```
5       4AAA 20     4AAB 43     4AAC 4F     4AAD 50     4AAE 49     4A
AF 45     4AB0 44     4AB1 20     4AB2 49     4AB3 4E     4AB4 54
  4AB5 4F     4AB6 20     4AB7 4D     4AB8 45     4AB9 4D     4ABA 4F
    4ABB 52     4ABC 59     4ABD 00     00280       DEFB    00H
4ABE 43      00290 MES5   DEFM   'COPY OPTIONS'
4ABF 4F      4AC0 50     4AC1 59     4AC2 20     4AC3 4F     4AC4 50
   4AC5 54     4AC6 49     4AC7 4F     4AC8 4E     4AC9 53     4ACA 0
0      00300        DEFB   00H
4ACB 4D      00310 MES6   DEFM   'M-LOAD NEW MASTER TAPE'
4ACC 2D      4ACD 4C     4ACE 4F     4ACF 41     4AD0 44     4AD1 20
   4AD2 4E     4AD3 45     4AD4 57     4AD5 20     4AD6 4D     4AD7 4
1      4AD8 53     4AD9 54     4ADA 45     4ADB 52     4ADC 20     4A
DD 54     4ADE 41     4ADF 50     4AE0 45     4AE1 00     00320
  DEFB   00H
4AE2 43      00330 MES7   DEFM   'C-MAKE ANOTHER COPY'
4AE3 2D      4AE4 4D     4AE5 41     4AE6 4B     4AE7 45     4AE8 20
   4AE9 41     4AEA 4E     4AEB 4F     4AEC 54     4AED 48     4AEE 4
5      4AEF 52     4AF0 20     4AF1 43     4AF2 4F     4AF3 50     4A
F4 59     4AF5 00     00340       DEFB   00H
4AF6 52      00350 MES8   DEFM   'R-RETURN TO BASIC'
4AF7 2D      4AF8 52     4AF9 45     4AFA 54     4AFB 55     4AFC 52
   4AFD 4E     4AFE 20     4AFF 54     4B00 4F     4B01 20     4B02 4
2      4B03 41     4B04 53     4B05 49     4B06 43     4B07 00     00
360       DEFB   00H
4B08 53      00370 MES9   DEFM   'S-SYMBOLIC TAPE DUMP'
4B09 2D      4B0A 53     4B0B 59     4B0C 4D     4B0D 42     4B0E 4F
   4B0F 4C     4B10 49     4B11 43     4B12 20     4B13 54     4B14 4
1      4B15 50     4B16 45     4B17 20     4B18 44     4B19 55     4B
1A 4D     4B1B 50     4B1C 00     00380       DEFB   00H
4B1D 20      00390 MES10  DEFM   ' ASCII-'
4B1E 41      4B1F 53     4B20 43     4B21 49     4B22 49     4B23 2D
   4B24 00     00400       DEFB   00H
4B25 20      00410 MES11  DEFM   '    HEX-'
4B26 20      4B27 20     4B28 20     4B29 48     4B2A 45     4B2B 58
   4B2C 2D     4B2D 00     00420       DEFB   00H
4B2E 54      00430 MES12  DEFM   'TAPE'
4B2F 41      4B30 50     4B31 45     4B32 00     00440       DEFB   00
H
4B33 44      00450 MES13  DEFM   'DUMP'
4B34 55      4B35 4D     4B36 50     4B37 00     00460       DEFB   00
H
4B38 48      00470 MES14  DEFM   'HIT'
4B39 49      4B3A 54     4B3B 00     00480       DEFB   00H
4B3C 41      00490 MES15  DEFM   'ANY'
4B3D 4E      4B3E 59     4B3F 00     00500       DEFB   00H
4B40 4B      00510 MES16  DEFM   'KEY'
4B41 45      4B42 59     4B43 00     00520       DEFB   00H
4B44 54      00530 MES17  DEFM   'TO'
4B45 4F      4B46 00     00540       DEFB   00H
4B47 41      00550 MES18  DEFM   'ADV'
4B48 44      4B49 56     4B4A 00     00560       DEFB   00H
4B4B 42      00570 MES19  DEFM   'BUFFER AREA TO SMALL IN 16K RAM'
4B4C 55      4B4D 46     4B4E 46     4B4F 45     4B50 52     4B51 20
   4B52 41     4B53 52     4B54 45     4B55 41     4B56 20     4B57 5
4      4B58 4F     4B59 20     4B5A 53     4B5B 4D     4B5C 41     4B
5D 4C     4B5E 4C     4B5F 20     4B60 49     4B61 4E     4B62 20
  4B63 31     4B64 36     4B65 4B     4B66 20     4B67 52     4B68 41
    4B69 4D     4B6A 00     00580       DEFB   00H
4B6B 30      00590 TABLE  DEFM   '0123456789ABCDEF'      ;ASCII LOOK UP TABLE
4B6C 31      4B6D 32     4B6E 33     4B6F 34     4B70 35     4B71 36
   4B72 37     4B73 38     4B74 39     4B75 41     4B76 42     4B77 4
3      4B78 44     4B79 45     4B7A 46     4B7B CD9A4D   00600 TOMUCH  CA
LL     ESCRN   ;IF BUFFER AREA EXCEEDED
```

```
4B7E 21103E  00610         LD    HL,3E10H      ;CLEAR SCREEN
4B81 222040  00620         LD    (4020H),HL
4B84 214B4B  00630         LD    HL,MES19      ;SETS CURSOR TO MID, THEN OU
TPUTS MES
4B87 CD804D  00640         CALL  BLKW          ;START PROGRAM OVER
4B8A CD024D  00650         CALL  BLANK         ;AFTER DELAY
4B8D C3BD40  00660         JP    START
4B90 0610    00670 OTPUT   LD    B,10H   ;COUNT ACROSS DISPLY LINE
4B92 CDE24B  00680         CALL  OUTPT1
4B95 E5      00690         PUSH  HL
4B96 7C      00700         LD    A,H
4B97 B5      00710         OR    L
4B98 2805    00720         JR    Z,OT2   ;CHECK IF LENGTH=0
4B9A CDAB4B  00730         CALL  OUTPT2
4B9D E1      00740         POP   HL
4B9E C9      00750         RET
4B9F 318049  00760 OT2     LD    SP,4980H      ;RETURNS TO MENU
4BA2 CD1F4D  00770         CALL  KBIN          ;AFTER ANY KEY IS PRESSED
4BA5 CD9A4D  00780         CALL  ESCRN
4BA8 C3304E  00790         JP    ASK
4BAB 2A2040  00800 OUTPT2  LD    HL,(4020H)    ;UPDATE CURSOR POSITION
4BAE 19      00810         ADD   HL,DE         ;LIKE LFCR WITHOUT ERASING
4BAF 222040  00820         LD    (4020H),HL
4BB2 C9      00830         RET
4BB3 DD23    00840 CHECK   INC   IX      ;CHECKS IF LENGTH=0
4BB5 2B      00850         DEC   HL      ;IF SO, REDUCES LOOP COUNT B
4BB6 7C      00860         LD    A,H     ;TO ZERO ○ ENDS DISPLY LINE
4BB7 B5      00870         OR    L
4BB8 2801    00880         JR    Z,ENDIT
4BBA C9      00890 OT1     RET
4BBB 0601    00900 ENDIT   LD    B,01H   ;ROUTINE FIXS CURSOR AT EOT
4BBD 2A2040  00910         LD    HL,(4020H)
4BC0 CB7D    00920         BIT   7,L
4BC2 2008    00930         JR    NZ,MTEST
4BC4 CB75    00940         BIT   6,L
4BC6 200C    00950         JR    NZ,MIDTST
4BC8 2E38    00960         LD    L,38H
4BCA 180E    00970         JR    OUTAGN
4BCC CB75    00980 MTEST   BIT   6,L
4BCE 2006    00990         JR    NZ,HTEST
4BD0 2EB8    01000         LD    L,0B8H
4BD2 1806    01010         JR    OUTAGN
4BD4 2E78    01020 MIDTST  LD    L,78H
4BD6 1802    01030         JR    OUTAGN
4BD8 2EF8    01040 HTEST   LD    L,0F8H
4BDA 222040  01050 OUTAGN  LD    (4020H),HL
4BDD 210000  01060         LD    HL,0000H
4BE0 18D8    01070         JR    OT1
4BE2 C5      01080 OUTPT1  PUSH  BC      ;LINE CHAR. COUNT
4BE3 DDE5    01090         PUSH  IX      ;CURRENT DATA POINTER
4BE5 E5      01100         PUSH  HL      ;CURRENT LENGTH VALUE
4BE6 DD7E00  01110 NCHR1   LD    A,(IX)  ;WRITES FULL LINE OF ASCII
4BE9 CD054C  01120         CALL  WRASC
4BEC CDB34B  01130         CALL  CHECK
4BEF 10F5    01140         DJNZ  NCHR1
4BF1 CDAB4B  01150         CALL  OUTPT2
4BF4 E1      01160         POP   HL
4BF5 DDE1    01170         POP   IX
4BF7 C1      01180         POP   BC
4BF8 DD7E00  01190 NCHR2   LD    A,(IX)  ;WRITES FULL LINE OF HEX CHARS.
4BFB CDAF4C  01200         CALL  WRHEX
4BFE CDB34B  01210         CALL  CHECK
4C01 10F5    01220         DJNZ  NCHR2
```


✓Reader Service—see page 162

```
4C03 C9      01230         RET
4C04 3AE837  01240 PRTDVR  LD      A,(LPRNT)      ;LINE PRINTER DRIVER ROUTINE
4C07 E6F0    01250         AND     0F0H
4C09 FE30    01260         CP      30H
4C0B 20F7    01270         JR      NZ,PRTDVR
4C0D 79      01280         LD      A,C
4C0E 32E837  01290         LD      (LPRNT),A
4C11 C9      01300         RET
4C12 3AE837  01310 LIST    LD      A,(LPRNT)      ;LINE PRINTER ROUTINE
4C15 FE3F    01320         CP      3FH
4C17 C0      01330         RET     NZ         ;CHECKS IF PRINTER ON LINE
4C18 E5      01340         PUSH    HL         ;IF IT IS IT PRINTS IF NOT RETURNS
4C19 C5      01350         PUSH    BC
4C1A 21003C  01360         LD      HL,3C00H
4C1D 0640    01370 LB      LD      B,40H
4C1F CB74    01380         BIT     6,H
4C21 200E    01390         JR      NZ,END1
4C23 4E      01400 LDATA   LD      C,(HL)
4C24 CD044C  01410         CALL    PRTDVR
4C27 23      01420         INC     HL
4C28 10F9    01430         DJNZ    LDATA
4C2A 0E0D    01440         LD      C,CR
4C2C CD044C  01450         CALL    PRTDVR
4C2F 18EC    01460         JR      LB
4C31 0E0D    01470 END1    LD      C,CR
4C33 CD044C  01480         CALL    PRTDVR
4C36 CD044C  01490         CALL    PRTDVR
4C39 C1      01500         POP     BC
4C3A E1      01510         POP     HL
4C3B C9      01520         RET
4C3C CDF94C  01530 ENTER1  CALL    OUTPUT
4C3F 211D4B  01540         LD      HL,MES10
4C42 CDB04D  01550         CALL    BLKW
4C45 2A2040  01560         LD      HL,(4020H)
4C48 09      01570         ADD     HL,BC
4C49 222040  01580         LD      (4020H),HL
4C4C C9      01590         RET
4C4D CDB04D  01600 ENTER2  CALL    BLKW
4C50 CDA74C  01610         CALL    LFCR
4C53 21254B  01620         LD      HL,MES11
4C56 CDB04D  01630         CALL    BLKW
4C59 CDA74C  01640         CALL    LFCR
4C5C 3C      01650         INC     A
4C5D C9      01660         RET
4C5E CD4D4C  01670 ENTERB  CALL    ENTER2
4C61 CD3C4C  01680         CALL    ENTER1
4C64 C9      01690         RET
4C65 013300  01700 BOR     LD      BC,33H  ;PUTS ASCII-HEX BORDER AND
4C68 21003C  01710         LD      HL,3C00H       ;ADV MESSAGE ON SCREEN
4C6B 222040  01720         LD      (4020H),HL
4C6E 3E31    01730         LD      A,31H
4C70 CD3C4C  01740         CALL    ENTER1
4C73 212E4B  01750         LD      HL,MES12
4C76 CD5E4C  01760         CALL    ENTERB
4C79 21334B  01770         LD      HL,MES13
4C7C CD5E4C  01780         CALL    ENTERB
4C7F 21384B  01790         LD      HL,MES14
4C82 CD5E4C  01800         CALL    ENTERB
4C85 213C4B  01810         LD      HL,MES15
4C88 CD5E4C  01820         CALL    ENTERB
4C8B 21404B  01830         LD      HL,MES16
4C8E CD5E4C  01840         CALL    ENTERB
4C91 21444B  01850         LD      HL,MES17
```

```
4C94 CD5E4C   01860        CALL   ENTREB
4C97 21474B   01870        LD     HL,MES10
4C9A CD5E4C   01880        CALL   ENTEPB
4C9D CDA74C   01890        CALL   LFCR
4CA0 21254B   01900        LD     HL,MES11
4CA3 CD004D   01910        CALL   BLKW
4CA6 C9       01920        RET
4CA7 F5       01930 LFCR   PUSH   AF        ;LINE FEED-CARRAGE RETURN
4CA8 3E0D     01940        LD     A,CR
4CAA CDF94C   01950        CALL   OUTPUT
4CAD F1       01960        POP    AF
4CAE C9       01970        RET
4CAF C5       01980 WRHEX  PUSH   BC        ;PRINTS A SPACE THEN 2 HEX CHRS.
4CB0 F5       01990        PUSH   AF
4CB1 0602     02000        LD     B,02H
4CB3 4F       02010        LD     C,A
4CB4 3E20     02020        LD     A,' '
4CB6 CDF94C   02030        CALL   OUTPUT
4CB9 79       02040        LD     A,C
4CBA 0F       02050        RRCA
4CBB 0F       02060        RRCA
4CBC 0F       02070        RRCA
4CBD 0F       02080        RRCA
4CBE E60F     02090 AGIN   AND    0FH
4CC0 CDCC4C   02100        CALL   LKUP
4CC3 CDF94C   02110        CALL   OUTPUT
4CC6 79       02120        LD     A,C
4CC7 10F5     02130        DJNZ   AGIN
4CC9 F1       02140        POP    AF
4CCA C1       02150        POP    BC
4CCB C9       02160        RET
4CCC E5       02170 LKUP   PUSH   HL        ;LOOKS UP ASCII VALUE OF CHRS.
4CCD 21604B   02180        LD     HL,TABLE
4CD0 85       02190        ADD    A,L
4CD1 6F       02200        LD     L,A
4CD2 7E       02210        LD     A,(HL)
4CD3 E1       02220        POP    HL
4CD4 C9       02230        RET
4CD5 C5       02240 WRASC  PUSH   BC        ;WRITES 2 SPACES AND THEN ASCII
4CD6 F5       02250        PUSH   AF
4CD7 47       02260        LD     B,A
4CD8 3E20     02270        LD     A,' '
4CDA CDF94C   02280        CALL   OUTPUT
4CDD 3E20     02290        LD     A,' '
4CDF CDF94C   02300        CALL   OUTPUT
4CE2 78       02310        LD     A,B
4CE3 D620     02320        SUB    20H
4CE5 FAF54C   02330        JP     M,NPNT
4CE8 78       02340        LD     A,B
4CE9 D67E     02350        SUB    7EH
4CEB F2F54C   02360        JP     P,NPNT
4CEE 78       02370        LD     A,B
4CEF CDF94C   02380 BK     CALL   OUTPUT
4CF2 F1       02390        POP    AF
4CF3 C1       02400        POP    BC
4CF4 C9       02410        RET
4CF5 3E20     02420 NPNT   LD     A,SUBST
4CF7 18F6     02430        JR     BK
4CF9 D5       02440 OUTPUT PUSH   DE        ;WRITES A AT CURSOR POSITION
4CFA FDE5     02450        PUSH   IY
4CFC CD3300   02460        CALL   33H
4CFF FDE1     02470        POP    IY
4D01 D1       02480        POP    DE
```


✓Reader Service—see page 162

```
4002 C9       02490        RET
4003 C5       02500 BLANK  PUSH   BC       ;JUMPS OVER TAPE HEADER
4004 E5       02510        PUSH   HL       ;ALSO PUTS SPACE BET. COPYS
4005 F5       02520        PUSH   AF
4006 0600     02530        LD     B,00H
4008 210000   02540 L6     LD     HL,0000H
400B 2B       02550 L5     DEC    HL
400C 7C       02560        LD     A,H
400D B5       02570        OR     L
400E 20FB     02580        JR     NZ,L5
4010 10F6     02590        DJNZ   L6
4012 F1       02600        POP    AF
4013 E1       02610        POP    HL
4014 C1       02620        POP    BC
4015 C9       02630        RET
4016 3A3F3C   02640 BLINK  LD     A,(3C3FH);BLINKS * WHEN LOADING
4019 EE0A     02650        XOR    0AH
401B 323F3C   02660        LD     (3C3FH),A
401E C9       02670        RET
401F D5       02680 KBIN   PUSH   DE
4020 CD2B00   02690        CALL   INPUT    ;RETURNS ON KEYBOARD INPUT
4023 B7       02700        OR     A        ;INPUT CHAR IN A REGISTER
4024 28FA     02710        JR     Z,KBIN+1
4026 D1       02720        POP    DE
4027 C9       02730        RET
4028 21403C   02740 MENU   LD     HL,3C40H;DISPLAYS ALL OPTIONS
402B 222040   02750        LD     (4020H),HL
402E 21BE4A   02760        LD     HL,MES5
4031 CD6840   02770        CALL   BLKW
4034 21CB3C   02780        LD     HL,3CCBH
4037 222040   02790        LD     (4020H),HL
403A 21CB4A   02800        LD     HL,MES6
403D CD6840   02810        CALL   BLKW
4040 21083D   02820        LD     HL,3D08H
4043 222040   02830        LD     (4020H),HL
4046 21E24A   02840        LD     HL,MES7
4049 CD6840   02850        CALL   BLKW
404C 21483D   02860        LD     HL,3D48H
404F 222040   02870        LD     (4020H),HL
4052 21F64A   02880        LD     HL,MES8
4055 CD6840   02890        CALL   BLKW
4058 21883D   02900        LD     HL,3D88H
405B 222040   02910        LD     (4020H),HL
405E 21084B   02920        LD     HL,MES9
4061 CD6840   02930        CALL   BLKW
4064 C9       02940        RET
4065 0608     02950 TAPRD  LD     B,08H    ;ASSEMBLES FULL BYTES FROM TAPE
4067 110000   02960 L1     LD     DE,0000H
406A CD7040   02970        CALL   BIN
406D 10F8     02980        DJNZ   L1
406F C9       02990        RET
4070 C5       03000 BIN    PUSH   BC
4071 F5       03010        PUSH   AF
4072 1B       03020 L2     DEC    DE       ;TIMEOUT ROUTINE
4073 7A       03030        LD     A,D      ;JUMPS TO RETN WHEN TAPE IS
4074 B3       03040        OR     E        ;DONE LOADING. NO MORE BITS
4075 CA124E   03050        JP     Z,RETN
4078 DBFF     03060        IN     A,(0FFH)
407A 17       03070        RLA
407B 30F5     03080        JR     NC,L2
407D 0641     03090        LD     B,41H
407F 10FE     03100 L3     DJNZ   L3
4081 E5       03110        PUSH   HL
```

```
4D82 CD1E02        CALL   021EH
4D85 E1            POP    HL
4D86 0676          LD     B,76H
4D88 10FE    L4    DJNZ   L4
4D8A DBFF          IN     A,(0FFH)
4D8C 47            LD     B,A
4D8D F1            POP    AF
4D8E CB10          RL     B
4D90 17            RLA
4D91 F5            PUSH   AF
4D92 E5            PUSH   HL
4D93 CD1E02        CALL   021EH
4D96 E1            POP    HL
4D97 F1            POP    AF
4D98 C1            POP    BC
4D99 C9            RET
4D9A E5      ESCRN PUSH   HL        ;ERASES COMPLETE SCREEN
4D9B C5            PUSH   BC
4D9C F5            PUSH   AF
4D9D 21003C        LD     HL,3C00H
4DA0 01FF03        LD     BC,3FFH
4DA3 3E20    NEXT  LD     A,' '
4DA5 77            LD     (HL),A
4DA6 23            INC    HL
4DA7 0B            DEC    BC
4DA8 78            LD     A,B
4DA9 B1            OR     C
4DAA 20F7          JR     NZ,NEXT
4DAC F1            POP    AF
4DAD C1            POP    BC
4DAE E1            POP    HL
4DAF C9            RET
4DB0 F5      BLKW  PUSH   AF
4DB1 7E      BLK   LD     A,(HL)    ;WRITES MESSAGES POINTED TO BY
4DB2 B7            OR     A         ;HL, UNTIL IT READS A 00H
4DB3 2806          JR     Z,DNE
4DB5 CDF94C        CALL   OUTPUT
4DB8 23            INC    HL
4DB9 18F6          JR     BLK
4DBB F1      DNE   POP    AF
4DBC C9            RET
4DBD 310049  START LD     SP,4900H;START OF MAIN PROGRAM HERE
4DC0 CD9A4D        CALL   ESCRN
4DC3 21003C        LD     HL,3C00H
4DC6 222040        LD     (4020H),HL

4DC9 21004A        LD     HL,CMES
4DCC CDB04D        CALL   BLKW
4DCF CDA74C        CALL   LFCR
4Dv2 21264A        LD     HL,MES1
4DD5 CDB04D        CALL   BLKW
4DD8 CDA74C        CALL   LFCR
4DDB 21504A        LD     HL,MES2
4DDE CDB04D        CALL   BLKW
4DE1 CDA74C        CALL   LFCR
4DE4 21794A        LD     HL,MES3
4DE7 CDB04D        CALL   BLKW
4DEA CD1F4D  AGN   CALL   KBIN
4DED FE0D          CP     CR
4DEF 20F9          JR     NZ,AGN
4DF1 21094E        LD     HL,BUFFER
4DF4 3E00          LD     A,00H
4DF6 CD1202        CALL   212H
```


✓Reader Service—see page 162

```
4DF9 CD9602   03750        CALL   296H
4DFC CD654D   03760 LOOP   CALL   TAPRD    ;READS BYTES FROM TAPE AND STORES
4DFF 77       03770        LD     (HL),A   ;THEM INTO MEMORY POINTED TO BY
4E00 FE0D     03780        CP     CR       ;THE HL REGISTER PAIR
4E02 CC164D   03790        CALL   Z,BLINK
4E05 FECD     03800        CP     BCHR
4E07 CC164D   03810        CALL   Z,BLINK
4E0A 23       03820        INC    HL
4E0B CB7C     03830        BIT    7,H      ;TESTS FOR BUFFER OVERFLOW
4E0D C27B4B   03840        JP     NZ,TOMUCH       ;8000H FOR 16K MACHINE
4E10 18EA     03850        JR     LOOP
4E12 310049   03860 RETN   LD     SP,4900H;PROGRAM JUMPS HERE AFTER IT
4E15 01094E   03870        LD     BC,BUFFER;TIMES OUT
4E18 B7       03880        OR     A
4E19 ED42     03890        SBC    HL,BC
4E1B 22074E   03900        LD     (LENGTH),HL
4E1E CD9A4D   03910        CALL   ESCRN
4E21 21003C   03920        LD     HL,3C00H
4E24 222040   03930        LD     (4020H),HL
4E27 21904A   03940        LD     HL,MES4
4E2A CD804D   03950        CALL   BLKW
4E2D CDF801   03960        CALL   1F8H
4E30 CD284D   03970 ASK    CALL   MENU     ;DISPLAYS MENU OF ALL POSSIBLE
4E33 CD1F4D   03980        CALL   KBIN     ;OPTIONS
4E36 FE4D     03990        CP     'M'
4E38 CAB04D   04000        JP     Z,START
4E3B FE52     04010        CP     'R'
4E3D CA0000   04020        JP     Z,BASIC
4E40 FE43     04030        CP     'C'
4E42 280A     04040        JR     Z,COPY
4E44 FE53     04050        CP     'S'
4E46 CA914E   04060        JP     Z,DISPLY
4E49 CD9A4D   04070        CALL   ESCRN
4E4C 18E2     04080        JR     ASK
4E4E CD9A4D   04090 COPY   CALL   ESCRN    ;START OF COPY ROUTINE
4E51 21003C   04100        LD     HL,3C00H
4E54 222040   04110        LD     (4020H),HL
4E57 21794A   04120        LD     HL,MES3
4E5A CD804D   04130        CALL   BLKW
4E5D CD1F4D   04140        CALL   KBIN
4E60 21094E   04150        LD     HL,BUFFER
4E63 3E00     04160        LD     A,00H
4E65 CD1202   04170        CALL   212H
4E68 CD034D   04180        CALL   BLANK
4E6B CD8702   04190        CALL   287H
4E6E ED4B074E 04200        LD     BC,(LENGTH)
4E72 7E       04210 NEXT1  LD     A,(HL)
4E73 CD6402   04220        CALL   264H
4E76 23       04230        INC    HL
4E77 FECD     04240        CP     BCHR
4E79 CC164D   04250        CALL   Z,BLINK
4E7C FE0D     04260        CP     CR
4E7E CC164D   04270        CALL   Z,BLINK
4E81 0B       04280        DEC    BC
4E82 78       04290        LD     A,B
4E83 B1       04300        OR     C
4E84 2802     04310        JR     Z,DEL
4E86 18EA     04320        JR     NEXT1
4E88 06FF     04330 DEL    LD     B,0FFH
4E8A 10FE     04340 DEL1   DJNZ   DEL1
4E8C CDF801   04350        CALL   1F8H
4E8F 189F     04360        JR     ASK
4E91 310049   04370 DISPLY LD     SP,4900H;ASCII-HEX DISPLAY PROGRAM
4E94 111000   04380        LD     DE,0010H
4E97 2A074E   04390        LD     HL,(LENGTH)     ;INITIAL LENNGTH OF TAPE
4E9A DD21094E 04400        LD     IX,BUFFER       ;INITIAL DATA BUF POINTER
4E9E E5       04410 BK1    PUSH   HL
4E9F CD9A4D   04420        CALL   ESCRN
4EA2 CD654C   04430        CALL   BOR      ;WRITES ASCII-HEX LABELS ON SCREEN
4EA5 21083C   04440        LD     HL,3C08H
4EA8 222040   04450        LD     (4020H),HL
4EAB E1       04460        POP    HL
4EAC CD904B   04470        CALL   OTPUT
4EAF CD904B   04480        CALL   OTPUT
4EB2 CD904B   04490        CALL   OTPUT
4EB5 CD904B   04500        CALL   OTPUT
4EB8 CD904B   04510        CALL   OTPUT
4EBB CD904B   04520        CALL   OTPUT
4EBE CD904B   04530        CALL   OTPUT
4EC1 CD904B   04540        CALL   OTPUT
4EC4 CD1F4D   04550        CALL   KBIN     ;CHECKS IF TO GO ON OR QUIT
4EC7 FE01     04560        CP     BREAK    ;BREAK KEY=RET. TO MENU
4EC9 CAD14E   04570        JP     Z,BRK    ;OTHER KEY=DISPLAY MORE
4ECC CD124C   04580        CALL   LIST
4ECF 18CD     04590        JR     BK1
4ED1 CD9A4D   04600 BRK    CALL   ESCRN
4ED4 C3304E   04610        JP     ASK
0002          04620 LENGTH DEFS   2        ;HOLDS LENGTH OF TAPE TO BE COPIED
4ED9          04630 BUFFER EQU    $        ;BEGINNING OF BUFFER AREA
4DB0          04640        END    START
00000 TOTAL ERRORS
```

Everyone had their own reasons for purchasing an 80—here is one owner's story.

An Owner's Tale

John Dilbeck
PO Box 187
Smyrna GA 30081

Eight years ago as a student at Georgia Tech, I was exposed to computer programming for the first time. As a part of my curriculum, I studied the basics of programming in FORTRAN and ALGOL. I was interested and enjoyed the experience, but I never expected to purchase one of those multi-million dollar electronic monsters.

I had not even heard of the BASIC language at that time and, since I chose to discontinue my engineering studies, I was not exposed to programming or operating computers until one day about eighteen months ago.

Then, the Radio Shack TRS-80 came galloping over the horizon. I was reading a Radio Shack catalog and there was a fully assembled, honest-to-goodness computer that could almost understand English.

During the time I was kept waiting for the computer to arrive, I spent my time reading everything I could lay my hands on concerning BASIC, micros, programming, logic, etc.

It only took about six weeks before I felt that I had control over most of the functions of the Level I system. I was able to write my own programs with ease and had begun to grow tired of hundreds of games of backgammon and blackjack.

Expanding

I eased on down to my Radio Shack store and asked, "How about getting me the Level II and 16K mods for my 80?" (You can see how fast I was picking up the lingo.)

A few weeks later, I got a call to bring my keyboard to the store. (This was at the height of the Level II backorder situation.) It was a week later that I tried out my new system and I realized that I had a lot of relearning to do.

The first thing I learned was to convert Level I programs into Level II.

I found that the conversion software took care of a lot of the changes, but not everything. For instance, there are not wrap-around graphics in Level II. I had been doing a number of experiments with graphics and had to rewrite extensively some of those programs.

Also, you must DIMension the arrays in Level II. In Level I there is only the A(x) array, but it is limited only by the size of the memory. The arrays in Level II are also only limited by the size of the memory, if you first set the maximum size of the array by the use of a DIMension statement.

I had just about mastered the Level II editing features (a great improvement over retyping the whole line), when the unexpected occurred.

I went into my office one morning and turned on the computer, and the whole screen filled up with letters, numbers, punctuation and graphics blocks in a random pattern (just as it does if you power off and then back on very quickly). I pressed the reset button and nothing happened. Then, I calmly turned the computer off, counted slowly to ten and powered back on . . . several times! It wouldn't work!

I grabbed the ailing machine and headed for the Radio Shack computer doctor. It was sent to the regional repair center in Atlanta, where they replaced a ROM.

More Memory

I converted games to the Level II dialect and I credit the problems encountered when translating these games with developing the confidence I now feel when writing my own programs.

While modifying one of my versions of Star Trek, I happened upon another example of the Fifth Law of Computer Programming, as stated in Terry Kepner's article in the August issue of *Kilobaud Microcomputing*. Any given program will expand to fill all available memory.

By now, I was a raving computer freak. So, I went out and signed some more promises and ordered an expansion interface with 32K of RAM, a line printer and two disk drives.

I ordered these additions in September, 1978; they began arriving in late November. However, Murphy was right! Everything that could go wrong, did. And at the worst possible times.

The directions for connecting two recorders were inaccurate. I made sure that I followed them correctly, stood back, scratched my head a few times, proceeded to reverse the order of the plugs and everything worked fine.

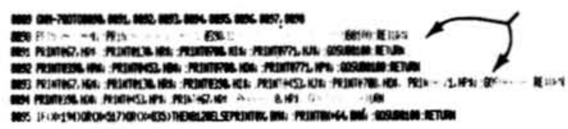
This is a risky thing to do; I really don't recommend this approach.

I bought a recorder controller from Micro-Mega and I am pleased with the product and the response time of this company. I would *not* use a recorder with the TRS-80 without one of these controllers.

The Printer

It is a wonderful feeling to sit back and watch the computer list its own programs or do reports, graphs, pictures (Snoopy, for example), calendars, etc. The only thing this requires is a printer that works properly. Mine didn't!

My printer is the Centronics 779 that is modified slightly for Radio Shack and has been nothing but trouble, though everyone, including other owners agreed my problems were unique.



Example 1: My ribbon occasionally hangs up causing the printer to strike the same place.

I had two problems with the printer. As the print head moved across the platen, occasionally the ribbon would hang up causing pins to strike the same place on the ribbon. The print became progressively fainter until I noticed and corrected the problem (see Example 1).

Secondly, as the printer was listing reports, the head assembly might go awry. It sounded like both of the solenoids had activated at once, causing the head to try to go both forward and return at the same time.

As a result of this malfunction, the printer thought it had already printed part of a line, and would continue across the page. It then printed whatever was still left in the buffer resulting in only partial lines being printed (see Example 2).

This could be very frustrating, especially at the end of a 15-minute printout.

I sent the printer to the regional repair center in Atlanta, along with samples of the printouts it had produced. They sent it back in a few days, after cleaning it, and said that nothing was wrong with the printer and that the problem must be in the software.

I then went through the software and to be absolutely sure, I rewrote each line of the program that generated output to the printer. Despite Radio Shack's assurances the printer continued to malfunction.

I sent the printer off again and it was returned with a note saying that there was nothing wrong with it.

The next time I returned the printer, I also sent the keyboard module and the expansion interface. Again, it was returned with the implication that my software was to blame.

I was starting to get upset.

It was about this time that both of the disk drives arrived and I temporarily forgot about the printer.

Disk Drives

The mini-disk method of storing and retrieving data is much faster than cassette and this is especially noticeable when loading 14-18K programs.

I was annoyed by the occasional LOST DATA DURING READ errors, until one day, during a write-to-disk, the computer "froze-up." I waited, in case it might be doing some operation I wasn't aware of such as housekeeping, rearranging the data, etc.

After five minutes I came back and pressed reset. That didn't work, so I turned off the power, waited for a few seconds and then turned it on again.

This was the procedure I was used to and I didn't think too much about it. However, this time it erased both diskettes in the process. At least I had a backup of the TRSDOS disk and didn't lose anything on #1 that couldn't be easily replaced.

I'm still not sure what happened, but since that day, I have been very careful to use the following procedure:

1. Turn on video.
2. Turn on the expansion interface.
3. Turn on the disk drives.
4. Turn on printer.
5. Turn on the keyboard.
6. Insert disks into drives.
7. Press RESET to load DOS.

If I have to power off any time after the disks are in, I remove the diskettes first. Also, it is important to be sure that a backup is made of each diskette, in case it is accidentally erased or damaged.

Using TRSDOS 2.1, I had a terrible time with lost data and other disk related errors. One of the salesmen at the Atlanta Computer Center suggested I move my disk drives away from the left side of the video. It seems that there is a magnetic field there which wreaks havoc with disk READ/WRITE operations. I moved the disk drives several inches to the right side of the keyboard and this resulted in some improvement.

But, I was still getting errors and garbage inserted into my programs. Sometimes this garbage would change commands, mess up the numbering, lose whole sections of the programs and other insidious actions. This was very frustrating.

I returned the disk drives to the Atlanta Radio Shack Computer Center for repair. The repairman cleaned both drives but found nothing wrong. When I hooked them up I still had the same problems.

I Return Everything!

Once more, I took the printer and both of the drives back to the repair center. One technician said that the stepper motor in drive 0 was defective and ordered a new one. Another technician finally found the real problem, one of the screws was loose and he tightened it. It worked much better, but still my machine was not cured.

When I returned the keyboard and interface to the computer center, to install the buffered cable, performance improved. I was no longer losing programs or having them altered by the machine.

I was still trying to get the printer functional. Extremely frustrated, I had consulted with a lawyer about filing suit against Radio Shack. I sent the printer back one more time, informing Radio Shack that I was ready to institute court action. When I received it this time, the printer worked fine. I still don't know what they did.

Reflecting back on these events, I have some thoughts I would like to pass on.

1. READ! Read the instructions and information about all your hardware and software. Read periodicals devoted to microcomputers and keep up to date.

2. PERSIST! Though it took a number of trips and several false starts, my Radio Shack system is working very well now.

3. I have learned from the Radio Shack newsletter that the upper six bytes of a 48K system should not be written into. When entering BASIC, answer the "MEMORY SIZE?" with *65529*.

4. Get together with other TRS-80 owners and form a users' group. Learn about your computer as you introduce others to its uses.

5. For you Radio Shack people, I'd like to suggest an upper-lowercase modification and improved graphics.

Though I've had much trouble, I feel my experiences were the exception rather than the rule. I hope my review will encourage others to enter the world of microcomputing. ■

```
1020 PRINT@J*10+634,A(W+J):NEXTJ:RETURN
2000 IFQ=1 THEN FORY=7TO12:A(Z+Y)=0:NEXTY
2020 A(W+7)=A(W+1):FORJ=2TO6:A(W+7)=A(W+7)-A(W+J):NEXTJ
2030 PRINT@705,"NET PAY=";A(W+7):PRINT:PRINT
2050 FORY=1TO7:A(W+Y+7)=A(W+Y+7)+A(W+Y)
2055 NEXTY:FORY=1TO6
A(Z+Y+6)=A(Z+Y+6)+A(W+Y):A(Z+Y+12)=A(Z+Y+12)+A(W+Y):NEXTY
2065 PRINT@384," UPDATED":GOSUB220:PRINT@768," ";
2066 IFA(W+1)=0INPUT"TERMINATE--1=YES,2=NO";D:IFD=1A(Z+5)=3
2070 INPUT"HIT ENTER TO GO ON";B$:NEXTX
2080 IFB<>0GOTO2095
2090 INPUT"# OF NEW EMP.";C:B=N:IFC>0N=C:GOTO70
2095 N=B+C
2100 CLS:GOSUB5000
```

Example 2: My printer would only print partial lines.

With a modem and RS232 your 80 can talk to the world—if you can afford the phone bill.

Computer Bulletin Boards

Jim Cambron
PO Box 10005
Kansas City, MO 64111

One of the fastest growing uses of microcomputers these days is for exchanging information between computers over the phone.

Private companies for a fee will let you access their computers to run programs, games and the like, or find a wealth of information, ranging from the latest sports scores from United Press International, to finding which restaurant in Cleveland offers the best service. There is an alternative to such national enterprises and one may be up and running in your town!

WORD LENGTH	=	7 bits (8 bits for TRS-80 Graphics)
PARITY	=	even, no parity.
STOP BITS	=	1 stop bit

Table 1.

Computer bulletin boards, also known as electronic message systems or electronic mailboxes, supported by local clubs, retailers and individual hobbyists are simple systems permitting the user to enter and retrieve messages.

Several bulletin board software sources are available for Radio Shack's TRS-80. One of the most useful is the Forum-80 package offered by Bill Abney, Kansas City, MO. Abney has organized a network of operators who regularly receive updates of the system software as well as suggest software improvements to the Forum-80 package. One of the latest—not yet available on all systems—allows a user to download a BASIC program from the Forum-80 to your TRS-80 (or other micro) using ST 80 D or another terminal program with similar downloading features.

Although every bulletin board has it's own personality, as you will soon find out, there is almost total standardization of serial word length, parity, stop-start bits and baud rates. This means that you can program your RS-232 board as described in Table 1 and communicate with nearly every bulletin board in the country.

What Do You Need?

To access a bulletin board

Table 2

TYPE	NAME	PHONE NUMBER
ABBS	A.C.E.S. FT. LAUDERDALE FL.	(305)566-0805
ABBS	AKRON DIGITAL GROUP	(216)745-7855
ABBS	ABBS ATLANTA ATLANTA GA.	(404)953-0723
ABBS	ABBS BOSTON BOSTON MA	(617)354-4682
ABBS	CALL APPLE VANCOUVER WA.	(206)524-0203
ABBS	CCNJ POMPTON PLAINS N.J.	(201)835-7228
ABBS	ABBS CHICAGO CHICAGO IL	(312)622-9609
ABBS	COMPUTER COMPONENTS LAWNDALE	(213)371-1660
ABBS	COMPUTER COMPONENTS WESTMIN.	(714)898-1984
ABBS	COMPUTER FORUM BUENA PARK	(714)739-0711
ABBS	COMPUTER LAB MEMPHIS TN	(901)761-4743
ABBS	COMPUTER MERCHANT SAN DIEGO	(714)582-9557
ABBS	COMPUTER STORE SANTA MONICA	(213)394-1505
ABBS	COMPUTER WORLD IRVINE CA.	(714)751-1422
ABBS	DALLAS DALLAS TX	(214)634-2775
ABBS	FT. WALTON BEACH DESTIN FL.	(904)243-1257
ABBS	HAWTHORNE HAWTHORNE CA	(213)675-8803
ABBS	HOUSTON HOUSTON TX	(713)977-7019
ABBS	ILLINI MICROCOMP. NAPERVL. IL	(312)420-7995
ABBS	KORSMEYER ELECT. HUNTINGTON	(714)962-7979
ABBS	MARINA DEL REY MARINA DEL REY	(213)821-7369
ABBS	MIAMI MIAMI FL.	(305)821-7401
ABBS	NEW YORK LONG ISLAND N.Y.	(212)448-6576
ABBS	PCNET SAN FRANCISCO CA.	(415)948-1474
ABBS	PEOPLES MESSAGE SYST. SANTEE	(714)449-5689
ABBS	PERIPHERALS UNLIMITED SIG.	(213)424-3506
ABBS	PERSONAL COMPUTERS OF CHIC.	(312)337-6631
ABBS	ABBS PHOENIX	(602)957-4428
ABBS	RAINBOW COMPUTING LOS ANGELES	(213)349-5728
ABBS	SAN ANTONIO TX	(512)657-0779
ABBS	SAN FERNANDO VALLEY CA.	(213)340-0135
ABBS	WEST PALM BEACH FLA.	(305)689-3234
	BILLBOARD 80 SAN JOSE CA.	(408)263-0248
CBBS	ATLANTA, GA.	(404)394-4220
CBBS	BOSTON MA.	(617)963-8310
CBBS	CHICAGO IL.	(312)528-7141
CBBS	DETROIT MICH.	(313)288-0335
CBBS	NW PORTLAND ORE.	(503)646-5510
CBBS	SAN FERNANDO LOS ANGELES	(213)843-5390
CBBS	WASHINGTON D.C.	(703)281-2125
CMMS	CMMS CHICAGO	(312)925-0259
COMM-80	COMM-80 ORANGE CO. CALIF.	(714)526-3687
COMPUSYST.	COMPUSYSTEMS COLUMBIA S. CAR.	(803)771-0922
FORUM-80	FAMILY HISTORIAN FORUM VA.	(703)978-7561
FORUM-80	DALLAS TEXAS	(214)288-4859
FORUM-80	DUNSTABLE MASS.	(617)649-7097
FORUM-80	FORT WORTH TEXAS	(817)923-0009
FORUM-80	KANSAS CITY, MO	(816)861-7040
FORUM-80	MEMPHIS TENN.	(901)276-8196
FORUM-80	ENGINEER-80 OLATHE	(913)764-1520
FORUM-80	ORANGE COUNTY SANTA ANA CA	(714)730-1206
FORUM-80	PRINCETON NEW JERSEY	(201)874-6833
FORUM-80	SAN FRANCISCO	(415)348-2139
FORUM-80	WICHITA, KANSAS	(316)746-2078
FORUM-80	WICHITA FALLS TEXAS	(817)855-3916
	MUSE CO., PARKVILLE MD.	(301)661-8962/63
	NEW ENGLAND COMP. SOC.	(617)897-0346
	POTOMAC MICRO MAGIC VA.	(703)750-0930
	REMOTE NORTH STAR ATLANTA	(404)939-1520
	S. JERSEY ELECT MAIL SYST.	(201)457-0893

system, you need a Level II TRS-80, an expansion interface with an RS-232 board installed, a modem and a software program. The program must organize the communication between the keyboard, CRT and the serial port.

This program can be written in BASIC, although it is far more efficient to write a machine language program to do the job. An example of such a program in Z-80 code for the TRS-80 is found in Program Listing 1. The concept is simple and you should be able to write a similar program for your system.

The idea is to constantly scan the serial port and the keyboard for an input from the outside world. If a byte is received from the keyboard, it is sent to the serial port. If a byte is received by the serial port, then it is displayed on the CRT or printed.

Once you have written or purchased a terminal program, you are ready to call a bulletin board. With the serial port set up as described in Table 1, pick up the telephone receiver and call the nearest bulletin board system (see Table 2). When the other end answers, you should hear a tone on the line.

Set your modem on Originate as described in the instruction manual. If all goes well, some sort of message will appear on your CRT/printer. If not, press your carriage return key several times until the distant system responds.

The bulletin board system may ask you some mysterious questions like 'DO YOU NEED LINEFEED?' and 'DO YOU NEED NULLS? (0 TO 50)' Answer 'N' to the linefeed question.

If you are using a printer with your TRS-80, slower than 300 baud, answer the NULLS question with an appropriate number between 1 and 50. If the number you enter is too small, the first characters in each line will not print because the head of your printer is in the process of a carriage return while those characters are being sent.

By entering a large enough

Table 3. Forum-80 Operating Instructions.

```
PASSWORD PROTECTED MESSAGES ARE ENTERED USING THE NORMAL
MESSAGE ENTRY ROUTINE (COMMAND 'E').  WHEN PROMPTED TO ENTER
THE 'SUBJECT:', THE FIRST 3 CHARACTERS ENTERED SHOULD BE: (P)
(THE P MUST BE UPPER CASE !). THIS SHOULD BE FOLLOWED BY YOUR
PASSWORD (NO SPACE BETWEEN '(P)' AND THE FIRST CHARACTER OF
YOUR PASSWORD !).  ALL CHARACTERS FOLLOWING (P) UP TO THE FIRST
SPACE WILL BE CONSIDERED AS YOUR PASSWORD.  PROTECTED MESSAGES
ARE RETRIEVED USING THE PROTECTED MESSAGE RETRIEVAL ROUTINE
WHICH IS ENTERED BY COMMAND 'P'.  THE MESSAGE CANNOT BE
BE RETRIEVED WITHOUT THE CORRECT PASSWORD.

CONTROL CHARACTERS RECOGNIZED BY THE SYSTEM:
  'D' -  DELETES THE GREETING PAGE & COMMAND LIST UPON
         ENTERING THE SYSTEM.  RECOGNIZED ONLY IN THE PAUSE
         AFTER ANSWERING THE LINEFEED QUESTION.
  'S' -  STOPS CURRENT FUNCTION AND PROMPTS FOR NEW COMMAND
         ACTIVE IN SUMMARY, MESSAGE RETRIEVAL, USER LOG,
         HELP AND INFORMATION ABOUT SYSTEM.
  'P' -  PAUSES AND ALLOWS 2 MIN TO STUDY SCREEN. TO CONTINUE
         HIT ANY CHARACTER. ACTIVE WHEN 'S' IS.
  'N' -  USED ONLY IN MULTIPLE MESSAGE RETRIEVAL.  SKIPS THE
         MESSAGE CURRENTLY PRINTING & GOES IMMEDIATELY TO THE
         NEXT MESSAGE IN SEQUENCE.
NOTE:  THESE ARE UPPER CASE ALPHABETICAL CHARACTERS,
NOT CONTROL CODES !

TO RECEIVE GRAPHICS YOU MUST MODIFY YOUR R.S. TERM PROGRAM
TO INITIALIZE WITH 8 BIT CHARACTERS AND TO ACCEPT AND TRANSMIT
ALL 256 POSSIBLE CHARACTER CODES.  THIS CAN BE DONE BY LOADING
YOUR TERM PROGRAM INTO MEMORY ALONG WITH TBUG OR RSM. MAKE THE
FOLLOWING CHANGES.  THEN PUNCH A NEW TAPE.  (DISK USERS WILL FIND
IT CONVENIENT TO USE DEBUG TO MAKE THE CHANGES. RETURN TO DOS
AND CREATE A NEW DISK FILE WITH TAPEDISK.)
CHANGE ALL THE FOLLOWING ADDRESSES TO CONTAIN THE BYTE SHOWN:
  5010 = B1                  5018 THRU 5020 = 00
  5001 = D3                  5007 = 3E
  5002 = E8                  5008 = 55
  5003 = 3E                  5009 = D3
  5004 = E5                  500A = E9
  5005 = D3                  500B = C9
  5006 = EA
REMEMBER, WHEN OPERATING IN GRAPHICS MODE, YOU MUST ENTER THE
FORUM-80 BY HITTING THE 'G' KEY, NOT C/R.  WHEN ENTERING
GRAPHICS MESSAGES, BE SURE TO START THE SUBJECT LINE WITH (G)
FOR THE PROTECTION OF THE NON-GRAPHICS USER.

S (SUMMARY OF MESSAGES IN SYSTEM)       L (LOG ON AGAIN)
Q (QUICK SUMMARY OF MESSAGES)           E (ENTER A MESSAGE)
H (HELP WITH SYSTEM OPERATION)          R (RETRIEVE MESSAGES)
I (INFORMATION ABOUT SYSTEM)            K (KILL A MESSAGE)
P (PROTECTED MESSAGE RETRIEVAL)         U (USERS LOG)
O (OTHER SYSTEM PHONE NUMBERS)          N (NULLS CHANGE)
T (TERMINATE CONNECTION)                G (GRAPHICS INFO)
F (FLAGGED MESSAGE RETRIEVAL)           C (CHANGE L/F SWITCH)

MESSAGE RETRIEVAL IS ENTERED USING THE 'R' COMMAND.  YOU WILL
BE ASKED IF YOU WISH SELECTIVE RETRIEVAL.  AS IN THE SUMMARY,
ANSWERING YES WILL ALLOW YOU TO SEE ONLY THE MESSAGES WHICH
MEET YOUR SEARCH CRITERIA.  IN BOTH REGULAR MULTIPLE RETRIEVAL
AND SELECTIVE MULTIPLE RETRIEVAL YOU WILL BE ASKED FOR THE
STARTING AND ENDING MESSAGE NUMBERS.  THIS ALLOWS YOU TO
RETRIEVE OR SEARCH A SPECIFIC RANGE OF MESSAGES. HITTING C/R
FOR STARTING MESSAGE NO. DEFAULTS TO THE BEGINNING OF THE FILE
AND FOR THE ENDING MESSAGE NO. DEFAULTS TO THE END OF THE FILE.
IF YOU ANSWER NO (OR C/R) TO THE MULTIPLE RETRIEVAL QUESTION
YOU WILL BE IN SINGLE RETRIEVAL MODE.  THE SYSTEM WILL RETRIEVE
EACH MESSAGE AS YOU ENTER THE NUMBER.  AT THE END OF A MULTILPLE
RETRIEVAL YOU ARE RETURNED TO SINGLE REPRIEVAL MODE.  TO RETURN
TO COMMAND MODE HIT C/R.

MESSAGE ENTRY IS ACCOMPLISHED WITH THE 'E' COMMAND.  THIS
FUNCTION IS SELF PROMPTING.  YOU ARE ALLOWED 16 LINES OF 63
CHARACTERS EACH.  THE ONLY CONTROL CODES RECOGNIZED IN MESSAGE
TEXT ENTRY ARE:
          CONTROL 'H' - (LEFT ARROW) BACKSPACE AND DELETE
                        ONE CHARACTER.
          CONTROL 'I' - (RIGHT ARROW) TABS RIGHT IN 8 SPACE
                        INCREMENTS.
          CONTROL 'X' - (SHIFT LEFT ARROW) DELETES THE ENTIRE
                        LINE.

SUMMARY IS ENTERED USING THE 'S' COMMAND.  YOU WILL BE ASKED
IF YOU WISH A SELECTIVE SUMMARY.  ANSWERING YES TO THIS
WILL ALLOW YOU TO SEE A SUMMARY OF ONLY THE MESSAGES WHICH MEET
```

```
YOUR SEARCH CRITERIA. FOR EXAMPLE, IF YOU WANT A SUMMARY OF
ALL THE MESSAGES PERTAINING TO THE TRS-80, YOU WOULD SEARCH THE
F(ROM) FIELD FOR THE STRING 'TRS-80'. IF YOU DO NOT WANT TO
SEARCH THE FILE THEN ANSWER 'N' OR JUST C/R. IN BOTH SELECTIVE
AND NORMAL SUMMARY YOU WILL BE ASKED FOR A STARTING MESSAGE
NUMBER. ENTERING THE NUMBER OF THE LAST MESSAGE YOU HAVE SEEN
WILL GIVE YOU A SUMMARY OF ALL MESSAGES SINCE YOUR LAST CALL
OR HITTING C/R WILL GIVE SUMMARY OF THE ENTIRE FILE.

FLAGGED MESSAGE RETRIEVAL IS ACCOMPLISHED AS FOLLOWS:

   1. ENTER THE REGULAR SUMMARY WITH COMMAND 'S'
   2. AS EACH MESSAGE HEADER IS PRINTING, DECIDE IF YOU WISH TO
      SEE THAT MESSAGE.
   3. AFTER THE HEADER HAS COMPLETED PRINTING, IF YOU WISH TO
      SEE THAT MESSAGE, HIT 'F' ANYTIME DURING THE PRINTING OF
      THE NEXT HEADER.
   4. AFTER YOU HAVE FINISHED THE SUMMARY AND RETURNED TO THE
      COMMAND MODE, USE COMMAND 'F' TO RETRIEVE THE FLAGGED
      MESSAGES.
```

number to the NULLS question, you force the bulletin board system to send null bytes (00 hex) while your printer is executing a return.

Using the Bulletin Board

The bulletin board now prints a greeting and asks you your name. Use your given name; it's important. Many of the bulletin boards scan the name inputted and compare it to the TO field of messages in the system. The board indicates to the person signing on that there is a message for him awaiting retrieval. Signing on as "KLEM CADIDDLEHOPPER" defeats the whole purpose of bulletin board systems.

Once you tell the system who you are, you can access the various functions of that particular system. A summary of functions for the Forum-80 bulletin board systems is given in Table 3.

If, after calling the system, you get lost, press your RETURN key several times to get back to the FUNCTION mode. Then enter an H for HELP! This function explains how to use the system. If you are totally lost, enter a T or G or EXIT, while in the FUNCTION mode, and the bulletin board system will disconnect.

Do not hang up on a bulletin board system! Although the software is, in many cases, written to handle the user who just hangs up, hanging up could cause a system crash putting it out of service for many other users!

Conclusion

In closing, if you have any problems communicating with a bulletin board system, consult your local computer club, bulletin board SYSOP or computer store. All these sources are willing to help you.

I would like to thank Bill Abney for information he provided about Forum 80. I am thankful to Hank Damm and the guys at the Kansas City Radio Shack Computer Center for their support. ■

Addendum

In the last couple of weeks, there has been a substantial improvement in the Forum 80 software. By the time this article is published, many Forum 80 systems will be using this new software. Readers can now obtain a *Forum 80 3.0 Users Guide*, which describes in detail how to use the Forum 80 to the best advantage, by sending a SASE with at least 30 cents postage affixed thereto to:

3.0 Users Guide
Forum 80 Headquarters
7600 E. 48th St. Terrace
Kansas City, MO 64129

Program Listing 1. TRS-80 Terminal Program

```
            00100 ;         SIMPLE MACHINE LANGUAGE TERMINAL PROGRAM FOR THE
            00110 ;         RADIO SHACK TRS-80 MICROCOMPUTER.  WRITTEN BY
            00120 ;         JIM CAMBRON P.O. BOX 10005 KANSAS CITY, MO 64111
            00130 ;         REQUIRES EXPANSION INTERFACE AND RS-232 BOARD!!!
            00140 ;
            00150 ;         1. PROGRAM INITIALIZATION
            00160 ;
7F90        00170           ORG     7F90H   ; FIRST INSTUCTION IS AT 7F90H
002B        00180 KBD       EQU     2BH     ; TRS-80 KEYBOARD SCAN ROUTINE
            00190                           ; IN BASIC ROM IS AT LOCATION 2BH
0033        00200 CRT       EQU     33H     ; TRS-80 VIDEO DISPLAY ROUTINE
            00210                           ; IN BASIC ROM IS AT LOCATION 33H
            00220 ;
            00230 ;         NOTE TO NON TRS-80 USERS...TO RUN THIS PROGRAM
            00240 ;         YOUR Z-80      MICRO, PLUG YOUR KEYBOARD AND
            00250 ;         VIDEO ROUTINES INTO THE ABOVE EQUATES...
            00260 ;
            00270 ;         THE ACTUAL PROGRAM STARTS HERE
            00280 ;
7F90 310080 00290 START     LD      SP,8000H        ; SET STACK POINTER
7F93 3E1C   00300           LD      A,28D           ; CLEAR SCREEN
7F95 CD3300 00310           CALL    CRT             ; (HOME AND CLEAR SCREEN)
7F98 3E1F   00320           LD      A,31D           ;
7F9A CD3300 00330           CALL    CRT             ;
7F9D 3E0E   00340           LD      A,14D           ; TURN ON CURSOR
7F9F CD3300 00350           CALL    CRT             ; CHARACTER
7FA2 D3E8   00360           OUT     (0E8H),A        ; RESET RS-232
7FA4 3E55   00370           LD      A,55H           ; SET BAUD RATE (55H = 300 BAUD)
7FA6 D3E9   00380           OUT     (0E9H),A        ;
7FA8 3EA5   00390           LD      A,0A5H          ; INITIALIZE FOR THE
7FAA D3EA   00400           OUT     (0EAH),A        ; CORRECT BIT COUNT
            00410                                   ; AND PARITY ETC.
            00420                                   ; 0A5H = 10100101 BINARY
            00430                                   ;       SEVEN BIT REGULAR
            00440                                   ;       MODE
            00450                                   ; 0E5H = 11100101 BINARY
            00460                                   ;       EIGHT BIT GRAPHIC
            00470                                   ;       MODE
            00480 ;
            00490 ;         2.   THE TERMINAL SCAN LOOP
            00500 ;
7FAC CDB47F 00510 LOOP      CALL    INPUT           ; CALL THE INPUT ROUTINE
7FAF CDC27F 00520           CALL    OUTPUT          ; CALL THE OUTPUT ROUTINE
7FB2 18F8   00530           JR      LOOP            ; THEN DO IT AGAIN......
            00540 ;
            00550 ;         3.   THE INPUT ROUTINE
            00560 ;
7FB4 DBEA   00570 INPUT     IN      A,(0EAH)        ; IS THE UART READY TO
            00580                                   ; RECIEVE A BYTE?
7FB6 CB7F   00590           BIT     7,A             ; CHECK BIT SEVEN TO SEE
7FB8 C8     00600           RET     Z               ; RETURN IF NOTHING THERE
7FB9 DBEB   00610           IN      A,(0EBH)        ; LOAD BYTE INTO A REG.
7FBB 28F7   00620           JR      Z,INPUT         ; IF ZERO, TRY ANOTHER
7FBD CD3300 00630           CALL    CRT             ; OTHERWISE, DISPLAY BYTE!
7FC0 18F2   00640           JR      INPUT           ; THEN GET ANOTHER BYTE..
            00650 ;
            00660 ;         4    THE OUTPUT ROUTINE
            00670 ;
7FC2 CD2B00 00680 OUTPUT    CALL    KBD             ; CHECK THE KEYBOARD...
7FC5 B7     00690           OR      A               ; ANYTHING??
7FC6 C8     00700           RET     Z               ; NO, RETURN.
7FC7 F5     00710           PUSH    AF              ; SAVE THE A AND F REGS.
7FC8 DBEA   00720 SEND      IN      A,(0EAH)        ; IS THE UART READY???
7FCA CB77   00730           BIT     6,A             ; IF NOT, THEN
7FCC 28FA   00740           JR      Z,SEND          ; LOOP BACK & TRY AGAIN.
7FCE F1     00750           POP     AF              ; RESTORE A AND F REGS
7FCF D3EB   00760           OUT     (0EBH),A        ; OUTPUT BYTE TO UART.
7FD1 C9     00770           RET                     ; AND RETURN TO LOOP
            00780 ;
            00790 ;
7F90        00800           END     START           ; END OF PROGRAM...
00000 TOTAL ERRORS
CRT    0033 00200  00310 00330 00350 00630
INPUT  7FB4 00570  00510 00620 00640
KBD    002B 00180  00680
LOOP   7FAC 00510  00530
OUTPUT 7FC2 00680  00520
SEND   7FC8 00720  00740
START  7F90 00290  00800
```

Got a fast 80? Use these modifications to speed up your DOS.

FASTDOS

Maynard B. Neher, Ph.D.
Battelle Columbus Division
505 King Avenue
Columbus, OH 43201

The August 1979 issue of Microcomputing (page 138) carried an article describing a simple modification which speeds up the TRS-80 by 50 percent.* However, the faster clock speed, 2.66 MHz, provided by this modification is incompatible with TRSDOS, the disk operating system software for the TRS-80.

The problem is that the floppy disk controller chip requires a delay of at least 20 microseconds after an instruction is loaded into the command register before the status register contains a valid status word. At 2.66 MHz the system software provides only about 15 microseconds delay.

TRSDOS 2.2 or 2.3 can operate at the regular speed of the TRS-80 (1.77 MHz clock rate) as well as at the faster speed with certain changes. Two system modules require modification, the bootstrap loader (BOOT), and the resident system module (SYSO). TRSDOS contains no provisions for these software modifications. However, a monitor system for the TRS-80, RSM-2D, allows alteration of TRSDOS. RSM-2D is available from Small Systems Software, Newbury Park, CA. Any system that permits disk modifications without regard for the protection status of the modules could be used.

Following the instructions, provided with the RSM-2D monitor load a copy of RSM-2D on your TRSDOS disk and execute it. You can also load RSM-2D and execute it directly from the RSM disk and then the TRSDOS diskette can be inserted into the drive. RSM-2D will operate only at 1.77 MHz.

BOOT Modification (Sector 0)

Load BOOT with RSM's load command at address 5000:

None of the other system modules (SYS1 to SYS6) include any disk I/O and consequently do not require any changes. However, two system utilities, BACKUP and FORMAT will not operate at the faster clock speed. So far, I have not been able to modify them even when

L 0 5000 1 (BOOT is in Sector 0)

Use RSM's EDIT command to alter locations according to Table 1.

Location	Old Contents	New Value
5091	F1	EE
50BB	F5 (PUSH BC)	CD (CD (CALL 42F9)
50BC	F1 (POP BC)	F9
50BD	F5 (PUSH BC)	42
50BE	F1 (POP BC)	00 (NOP)
50ED	54 "T"	0D CR
50EE	45 "E"	17
50EF	40 "M"	E8
50F0	0D "CR"	44 "D"
50F1	17	49 "I"
50F2	E8	53 "S"
50F3	44 "D"	4B "K"
50F4	49 "I"	20 "Space"
50F5	53 "S"	45 "E"
50F6	4B "K"	52 "R"
50F7	20 "Space"	52 "R"
50F8	45 "E"	0D "CR"
50F9	52 "R"	C5 (PUSH BC)
50FA	52 "R"	06 (LD B, 08H)
50FB	4F "O"	08
50FC	52 "R"	10 (DJNZ $-2)
50FD	0D "CR"	FE
50FE	EB Garbage	C1 (POP BC)
50FF	5F	C9 (RET)

Finally, write the modified BOOT back to the disk using RSM's save command:

$ 0 5000 1

Table 1. BOOT Modification (Sector 0).

* The simple switch modification is not entirely satisfactory. Switching from one speed to the other (in either direction) causes the system to hang fairly frequently. At least two commercial speed change kits are available that avoid this problem. One is a switch operated modification, the second is operated by software. Three of our TRS-80s at Battelle use the commercial switch operated speed change kit.

inserting delays where apparently needed. I would appreciate hearing from anyone that is able to find the key "append-to," making these routines operate at the faster clock speed ■

SYSO Modification (Sectors 5-19)

Load SYSO into memory at 5000 using RSM's load command:

```
L 5 5000 15
```

Use RMS's EDIT command to modify locations according to Table 2.

Location	Old Contents	New Value
51EC	1C	F3 (DI) (Delay Subroutine)
51ED	1F	C5 (PUSH BC)
51EE	1E	06 (LD B, 10H)
51EF	1F	10
51F0	1E	10 (DJNZ $-2)
51F1	1F	FE
51F2	1E	C1 (POP BC)
51F3	1F	C9 (RET)
5293	C5 (PUSH BC)	CD (CALL 45F3)
5294	C1 (POP BC)	F3
5295	C5 (PUSH BC)	45
5296	F3 (DI)	C5 (PUSH BC)

Write the modified code back to disk with RSM's save command:

```
S 5 5000 15
```

Table 2. SYSO Modification (Sectors 5-19).

Use this elegant program to help design video layouts and utilize them in your programs.

Etch-A-Screen

James K. Shrum
4318 Pathfinder Avenue
Cheyenne, WY 82001

Though graphic displays are one of the most interesting attributes of personal microcomputing, producing them on a TRS-80 from scratch can be an exhausting exercise in frustration.

First, you must plot the display on the planning worksheet, construct equations for the proper locations, formulate the FOR-NEXT loops, write the program statements. Run them . . . adjust them. Run tne program again . . . readjust it. Run it once more . . . still more readjustments . . . There's gotta be a better way!

There is! Remember the children's toy on which you drew pictures by turning knobs? With a little bit of effort, we can construct a Level II BASIC program that will do that and even more. It will give us real-time keyboard control of the graphics functions, let us mix text with graphics, and save patterns on tape for recall and redisplay at any time.

Floor plans, circuit diagrams, maps—whatever you do that can use computer graphics, can be done much more easily with the program we are going to construct.

The program is divided into four major modules: drawing, text, tape save and tape load. We'll complete one module at a time, merging each one with the previous ones. Finally, we'll condense the program to make it usable on a 4-K machine.

The Drawing Module

Let's write this program to simulate a cursor that the user can control simply with five keys: the space bar and the four arrows. As the cursor moves in an arrow direction, we want to leave it a line behind to show its trail. The space bar must move the cursor without drawing a line, and be able to erase a previous line. Finally, by using two arrows at once, the cursor must move diagonally.

The first trick is to input our program. The INKEY$ function won't cut it, since it only accepts input from a single key at a time. What we have to remember is that the TRS-80 keyboard is memory-mapped to the CPU. In other words, the CPU "sees" the keyboard as a section of memory not as an I/O device. The memory map in the owner's manual, tells us that the CPU sees the keyboard addresses 14336 to 15359 decimal.

BASIC can also access these addresses, using a PEEK.

If you run Program 1 you can see what goes on in those memory addresses as various keys are depressed.

The resulting numbers don't look much like ASCII, but don't worry about that. The CPU gets ASCII by scanning the keyboard in two dimensions and interpreting the results. What we want is the raw data in the memory without the CPU's translation of it. By using the two lines in Program 2, we'll see what we have to work with.

Look at the labeled sample run. A unique number is present in memory address 145990 each time an arrow key, the space bar, or any combination of two or three of these is depressed. With a PEEK followed by appropriate decision statements, we now have the foundations of our drawing module. (Note: Although I used memory address 14590, there's nothing very special about it. A number of others would serve equally well.)

Building the Draw Module

Now we can build the drawing module itself, shown expanded in Program 3. Not too formidable looking, but very effective. Let's take a close look at it and see how it operates.

We start with a clear screen in line 10. Line 20 takes whatever is in 14590 and calls it B. We want to do a little checking of B to prevent the computer from taking any action if we give it a strange command such as "move the cursor up and down simultaneously." In addition, if the space bar alone is depressed, we want line 20 to immediately branch program execution to the step that displays the current cursor location.

If we make it past these tests, lines 25 through 60 check for a depressed arrow, by testing all of the possible values we found with Program 2. If an arrow key is depressed, we adjust our cursor location accordingly, by changing the values of X and Y. Since X (the horizontal coordinate) must fall between 0 and 127 and Y (the vertical coordinate) must fall between 0 and 47, their values are checked. (If the end portions of these statements look a little odd, it is because I used implied THEN statements quite a bit.)

```
10 FOR I=14336 TO 15539 : PRINT I; PEEK(I), : NEXT

 14440  128      14441  128      14442  0       14443  0
 14444  16       14445  16       14446  32      14447  32
 14448  0        14449  0        14450  16      14451  16
 14452  128      14453  128      14454  128     14455  128
 14456  0        14457  1        14458  128     14459  128
 14460  128      14461  64       14462  0       14463  64
 14464  0
```

Program 1. In this sample run, the five-digit numbers are the memory addresses and the smaller numbers next to them are the contents of each address. The numbers were put into those addresses by randomly pressing keys during the run.

The updated cursor location is displayed with line 70. Statement 80 then checks the space bar. If it is depressed, we enter a short delay loop to give the cursor time to show us where it is, then turn the cursor off. Line 90 repeats the entire module.

Let's see how this module operates. Enter the statements and RUN. The screen will go blank. Press and hold the space bar and a spot in the upper left of the screen will begin blinking. That is our cursor in its starting location.

Now, while holding the space bar, press and hold the right ar-

```
1Ø PRINT PEEK (1459Ø);
2Ø GOTO 1Ø

              UP ARROW
8    8    8    8    8    8    8    8    8    8

              DOWN ARROW
16   16   16   16   16   16   16   16   16   16

              RIGHT ARROW
64   64   64   64   64   64   64   64   64   64

              LEFT ARROW
32   32   32   32   32   32   32   32   32   32

              SPACE BAR
128  128  128  128  128  128  128  128  128  128

              UP AND DOWN
24   24   24   24   24   24   24   24   24   24

              RIGHT AND LEFT
96   96   96   96   96   96   96   96   96   96

              UP AND RIGHT
72   72   72   72   72   72   72   72   72   72

              UP AND LEFT
4Ø   4Ø   4Ø   4Ø   4Ø   4Ø   4Ø   4Ø   4Ø   4Ø

              DOWN AND RIGHT
8Ø   8Ø   8Ø   8Ø   8Ø   8Ø   8Ø   8Ø   8Ø   8Ø

              DOWN AND LEFT
48   48   48   48   48   48   48   48   48   48

              UP WITH SPACE
136  136  136  136  136  136  136  136  136  136

              DOWN WITH SPACE
144  144  144  144  144  144  144  144  144  144

              RIGHT WITH SPACE
192  192  192  192  192  192  192  192  192  192

              LEFT WITH SPACE
16Ø  16Ø  16Ø  16Ø  16Ø  16Ø  16Ø  16Ø  16Ø  16Ø

              UP AND RIGHT WITH SPACE
2ØØ  2ØØ  2ØØ  2ØØ  2ØØ  2ØØ  2ØØ  2ØØ  2ØØ  2ØØ

              UP AND LEFT WITH SPACE
168  168  168  168  168  168  168  168  168  168

              DOWN AND RIGHT WITH SPACE
2Ø8  2Ø8  2Ø8  2Ø8  2Ø8  2Ø8  2Ø8  2Ø8  2Ø8  2Ø8

              DOWN AND LEFT WITH SPACE
176  176  176  176  176  176  176  176  176  176
```

Program 2. This program watches memory address 14590 decimal and shows us continually what is in that location. The labels tell what keys were pressed during each portion of the run. For instance, UP AND RIGHT WITH SPACE means that the up arrow, the right arrow, and the space bar were all depressed simultaneously.

row and the cursor will move to the right. While the cursor is moving, release the space bar and a solid line will be drawn to the right.

Experiment by pressing all of the arrows, both alone and in combination with the space bar. Hold the right arrow and the down arrow depressed together, and you'll draw a line diagonally down and to the right.

To erase an unwanted line, just position the cursor and move it in the desired direction while holding the space bar down. Much easier to use than describe, isn't it?

With this module operating, we can draw whatever we want without the trials and tribulations of constructing equations and loops and putting them into program statements.

Experiment with the keyboard. While this module operates, certain other keys also give us valid input. For instance, the W key sends the same signal to our memory address as the space bar, and there are several other duplications as well.

The Text Program

Now that we have our computer drawing pictures for us, let's construct a module for text. The text module allows us to display alphanumeric characters on the screen with our drawing. With it we can add labels or captions. This may be a nanotrifle more involved than the preceeding, so hang on.

Program 4 is the expanded listing of our text module, along with the two changes and one addition we have to make to the drawing module so they can work together.

First, we need a decision statement so the computer knows when to stop drawing and start displaying characters. We can use the INKEY$ function for this. If you played with the keyboard while drawing, you found out we can't use W for write, or T for text or P for print, because all of those letters put valid numbers into our memory address. That's why our new line 15 now checks to see if D for "done drawing" is pressed. If it is, we branch to text. Since this line slides in at the very beginning of drawing, we have to change lines 20 and 90 to reference the new line 15.

We start the text module by assigning values to the variables we will use in line 100. "A" will be the screen memory address used for our cursor location. If you look again at the memory map, you'll see that memory addresses 15360 through 16383 control screen display locations 0 through 1023, with 15360 being the upper left corner. We assign the value of 191 to variable C. This is the code number for a full graphics block and we will be using it as our text cursor.

Now comes the meat of the module. Line 110 takes whatever is currently being displayed at screen location A and saves that character's code number as variable D, then replaces it with

```
   CHANGE LINE 10
10 CLEAR 1250 : DIM S$(15) : CLS

    CHANGE LINE 100
100 B$=INKEY$:IFB$=""THEN100 ' *** CHECK FOR INPUT

    ADD LINES 102 THROUGH 105
102 IFB$="T"THEN108'_ *** GOTO TEXT ROUTINE
103 IFB$="S"THEN200'_ *** GOTO SAVE ROUTINE
104 IFB$="L"THEN500'_ *** GOTO LOAD ROUTINE
105 GOTO 100'_ *** UNACCEPTABLE-TRY AGAIN

    OLD LINE 100 BECOMES 108
108 A=15360:C=191

    OLD LINE 200 BECOMES 190
190 GOTO 110
```

Example 1. These are the changes we need to make to our growing program so that we can add the tape save and load modules.

our text cursor. We want to save D so we can reconstruct our display when we move the cursor to a new location.

Line 120 checks for a pressed key. If there are none, we blink the cursor to show us where it is and then check again for a keypress.

If a key has been pressed, we go to line 125, which gives us the ASCII code for that key and assigns it variable B.

Line 140 is our escape hatch from the text module. If the key we pressed was ENTER, our display is restored to its original form by returning D to location A and we are returned to drawing.

If not, line 140 checks for a pressed alphanumeric by examining the ASCII value. If so, it is displayed at the cursor's current position and the cursor is advanced by adding 1 to A. At the end of the line, we make sure that the value of A remains within the permissible limits so we don't go POKE somewhere we shouldn't.

Lines 145 through 180 check for an arrow key. If a right or left arrow was pressed, we adjust the value of A by 1 to move horizontally on the same line. For the up and down arrows, we use an adjustment of 64, to move us in a vertical line.

After each use of an arrow key, we must ensure that the value of A is still within the limits, and we replace the character we took out of that location to display our cursor. This way, we avoid leaving a trail of destruction across the screen.

```
5 REM      *** TRS-80 "ETCH-A-SCREEN" MAY 1979 BY J.K. SHRUM ***
10 CLS
15 REM      *** UP & DOWN OR LEFT & RIGHT TOGETHER ARE INVALID ***
16 REM      *** SPACE BAR ALONE JUMPS TO CURSOR FLICKER ***
20 B=PEEK(14590):IF B=0 OR B=24 OR B=96 THEN 20 ELSE IF B=128 THEN 70
25 REM      *** CHECK FOR UP ARROW ***
30 IF B=8 OR B=40 OR B=72 OR B=136 OR B=168 OR B=200  Y=Y-1:IF Y<0  Y=0
35 REM      *** CHECK FOR DOWN ARROW ***
40 IF B=16 OR B=48 OR B=80 OR B=144 OR B=176 OR B=208  Y=Y+1:IF Y>47  Y=47
45 REM      *** CHECK FOR LEFT ARROW ***
50 IF B=32 OR B=40 OR B=48 OR B=160 OR B=168 OR B=176  X=X-1:IF X<0  X=0
55 REM      *** CHECK FOR RIGHT ARROW ***
60 IF B=64 OR B=72 OR B=80 OR B=192 OR B=200 OR B=208  X=X+1:IF X>127  X=127
70 SET (X,Y)
75 REM      *** IF SPACE BAR DEPRESSED, ERASE ***
80 IF B>85 FOR I=1 TO 10:NEXT:RESET (X,Y)
90 GOTO 20
```

Program 3. Expanded listing of our drawing module.

```
     ADD LINE 15
15 IF INKEY$ = "D" THEN 100

     CHANGE LINE 20
20 B= PEEK(14590): IF B=0 OR B=24 OR B=96 THEN 15 ELSE IF B=128 THEN 70

      CHANGE LINE 90
90 GOTO15

                    LISTING FOR 'TEXT' MODULE

95 REM              *** THIS PART OF PROG. ADDS TEXT ***
98 REM              *** SET VARIABLE VALUES ***
100 A=15360 : C=191
105 REM             *** SAVE CURRENT GRAPHICS CHARACTER ***
110 D= PEEK(A):POKE A,C
115 REM             *** GET KEY FROM KEYBOARD AND FLICKER CURSOR ***
120 B$= INKEY$:IF B$ ="" POKE A,32: FOR I=1 TO 10: NEXT: POKE A,D: GOTO 110
123 REM             *** GET ITS VALUE ***
125 B=ASC(B$)
128 REM             *** IF C.R. -- RET. TO DRAWING ***
130 IF B= 13 POKE A,D:GOTO 15
135 REM             *** CHECK FOR ALPHA CHARACTER ***
140 IF B >31 AND B <91 POKE A,B:A=A+1:IF A >16383 A=16383
145 REM             *** CHECK FOR LEFT ARROW ***
150 IF B=8 POKE A,D: A=A-1: IF A <15360 A=15360
155 REM             *** CHECK FOR RIGHT ARROW ***
160 IF B=9 POKE A,D: A=A+1: IF A >16383 A=16383
165 REM             *** CHECK FOR DOWN ARROW ***
170 IF B=10 POKE A,D :A=A+64: IF A >16383 A=A-64
175 REM             *** CHECK FOR UP ARROW ***
180 IF B=91 POKE A,D: A=A-64: IF A <15360 A=A+64
200 GOTO 110 ' *** DO IT AGAIN ***
```

Program 4. The top three program lines are the changes necessary to interface the text and drawing modules. The text module itself is listed in expanded form below them.

The Return Trip

Finally, line 200 sends us back to the beginning of the module to repeat the process.

Like the drawing module, this portion of our program is much more easily used than explained. Once the two modules have been combined, draw yourself a picture. After you're done, tap the D to tell the computer. A large block of light will appear in the upper left corner of the screen and blink.

Now, use the arrow keys to move the cursor to the desired location on the screen. Because we are using INKEY$, each separate key press moves the cursor one step. Notice that you can move the cursor right through the pattern on the screen without damage. In addition, since the screen display is controlled by a single block of memory, we have automatic "wrap-around" from one line to the next, moving both right and left.

After the cursor appears where it's desired, simply type in your label or message. The letters or numbers will be placed in the cursor's position, and the cursor will move right one space. The captions on the screen will be unaffected by the cursor's movement, so to replace something, just reposition the cursor and type over it. To erase, position the cursor and hit the space bar.

Notice that characters will erase portions of our pattern at times. That is why we used a full graphics block as our cursor. The block delineates each character location's boundary so we can see what we are going to erase prior to actually doing so.

Make a mistake? Erase something you didn't want erased? No problem.

Hit ENTER and you will be returned to drawing so you can touch up or change the display. In fact, you can alternate at will between drawing and text by using the D and ENTER keys.

```
200 REM          *** SAVE ON TAPE ***
210 REM          *** CONSTRUCT THE STRINGS ***
215 FOR I= 0 TO 15 : S$(I) = "" : NEXT ' *** BLANKS THE STRINGS
220 FOR I= 0 TO 15 '  *** VERT LOOP
230 FOR J= 0 TO 63 '  *** HORIZ LOOP
240 S$(I) = S$(I) + CHR$( PEEK( 15360+I*64+J )) ' *** ADDS NEXT CHR
250 POKE 15360+I*64+J,42 ' *** SHOW WHERE WE ARE
260 NEXT J,I ' *** CLOSE THE LOOPS
270 REM          *** ADD ASTERISKS ***
280 FOR I = 0 TO 15
290 S$(I) = "*" + S$(I) + "*"
300 NEXT
310 REM          *** PRINT THE PROMPTS ***
320 CLS : PRINT CHR$(23)
330 PRINT "LOAD CASSETTE & POSITION TAPE"
340 PRINT : PRINT "DEPRESS 'RECORD' AND 'PLAY' KEYS"
350 PRINT:PRINT:PRINT:PRINT:INPUT"PRESS 'ENTER' WHEN READY";Z
355 PRINT "RECORDING"
360 REM          *** PUT STRINGS ON TAPE ***
370 PRINT #-1,S$(0),S$(1),S$(2)
380 PRINT #-1,S$(3),S$(4),S$(5)
390 PRINT #-1,S$(6),S$(7),S$(8)
400 PRINT #-1,S$(9),S$(10),S$(11)
410 PRINT #-1,S$(12),S$(13),S$(14)
420 PRINT #-1,S$(15)
430 GOTO 670 ' *** REDISPLAY PATTERN & RETURN TO DRAWING
```

Program 5. Expanded listing of our save module.

```
500 REM  *** LOAD FROM TAPE ***
510 REM          *** PRINT THE PROMPTS ***
520 CLS : PRINT CHR$(23)
530 PRINT "LOAD CASSETTE AND POSITION TAPE"
540 PRINT : PRINT "DEPRESS 'PLAY' KEY"
550 PRINT:PRINT:PRINT:INPUT"PRESS 'ENTER WHEN READY";Z
560 PRINT "LOADING"
570 REM          *** BLANK THE STRINGS ***
580 FOR I = 0 TO 15
590 S$(I) = ""
600 NEXT  ' *** NOW READ THE TAPE ***
610 INPUT #-1,S$(0),S$(1),S$(2)
620 INPUT #-1,S$(3),S$(4),S$(5)
630 INPUT #-1,S$(6),S$(7),S$(8)
640 INPUT #-1,S$(9),S$(10),S$(11)
650 INPUT #-1,S$(12),S$(13),S$(14)
660 INPUT #-1,S$(15)
665 REM          *** GET RID OF ASTERISKS ***
670 FOR I = 0 TO 15
680 S$(I) = MID$( S$(I),2,64 )
690 NEXT
700 REM          *** DISPLAY 1ST 15 LINES ***
705 CLS
710 FOR I = 0 TO 14
720 PRINT S$(I);
730 NEXT
740 REM          *** PRINT 1ST 63 CHR OF LAST ***
750 PRINT LEFT$( S$(15),63 );
760 REM          *** SHOW LAST CHR ***
770 POKE 16383, ASC( RIGHT$( S$(15),1 ))
780 GOTO 15 ' *** RETURN TO DRAWING
```

Program 6. Expanded listing of our load module.

Once your pattern is complete and labeled, just toggle your screen printer for a hard copy of your graph of monthly sales or your picture of a clown.

Tape Considerations

Now that we have the drawing and text modules making pretty pictures for us, we have to have some way of saving them. We do have a method of saving data on tape, but problems arise using the PRINT# statement the easy way:

```
10 FOR I = 15360 TO 16383
20 PRINT # - 1, PEEK (I)
30 NEXT I
```

This program does indeed work, but if you decide to try it, I have to suggest using a loooooong cassette, because it will take between 60 and 70 minutes to output all of the screen data.

The tape I/O format causes delays. Each separate PRINT# statement must turn on the cassette recorder, output a carrier tone and add a string of 128 leader bytes to a synch byte prior to actually transmitting the data called. Each print statement takes just over four seconds to execute and there are over a thousand bytes of screen data to record.

All hope is not lost, though. Each PRINT# statement can handle up to 255 bytes of string data. At a transmission rate of 500 baud, that only lengthens the execution time of a PRINT# statement by a little more than half a second. If we can structure our screen memory into five or six strings of characters, we can record all of the data in less than forty-five seconds. Sound better than taking an hour?

Example 1 shows the program changes and additions we have to make in our previously completed modules so that we can add our upcoming save and load routines.

Our new line 10 gets us ready, by reserving space for the strings we will be constructing. S$ is a sixteen element array (remember, S$(0) is a usable element), so each element will handle one line of the screen display. This method makes our programs easier to handle and to troubleshoot.

Line 100 is changed because we will have three options from which to choose after typing D for done drawing. Our new line 100 will wait for us to decide and then accept input from us with INKEY$.

The new lines 102 through 104 then branch us to the appropriate module. We'll continue to use T for text, and add S for save and L for load. If none of these letters are entered, line 105 sends the computer back to line 100 to give us another chance.

Set the variables that text uses in line 108 and move our old line 200 to line 190, so we can start our save module at line 200 (no special reason—I like to use round hundreds for starting lines during program development).

Save Module

The save module is shown in expanded form in Program 5. It looks relatively simple, but it accomplishes quite a bit.

First of all, we have to clear

✓Reader Service—see page 162

our string array in case it has already been used. Line 215 does this by making each element of S$ a null string (" ").

Lines 220 and 230 then set up the nested loops we will use for building the strings from the screen display. The loop indexed by I will take each line on the screen from top to bottom, while the J loop covers each character position on the I line from left to right.

Line 240 looks complex, but isn't as bad as it may appear. It merely takes the S$ array element addressed by the I loop and adds the characters on that screen line. Since there are 64 characters per line, multiplying I by 64 puts us at the beginning of each new line every time I is incremented in the loop. Adding the value of J will move us from left to right and adding 15360 puts us into the block of memory controlling the screen display. PEEK gives us the code at that address, and CHR$ gives us a character we can add to our growing string.

I dislike programs that run on without letting me know that something is happening, so line 250 displays an asterisk each time a character is added to our S$ array element. This way we can watch the progress of our string building.

Finally, line 260 closes both our loops.

Lines 230 through 300 solve a problem that had me baffled for some time. Prior to adding these lines, I would lose characters from my strings after saving and then reloading them. S$(0) had 64 characters as it was supposed to, but each one following had fewer until S$(15) ended up with about 45 or so. Needless to say, this resulted in some interesting displays. I discovered, by trying different display patterns, that strings did not reliably transfer to tape if they began and ended with a blank character (a space). Lines 280 through 300 tack on an asterisk (I like asterisks) to the beginning and end of each string, ensuring they don't drop from sight.

Another problem is caused by commas when used in the display. Commas are string delimiters and will be read as such by

```
 10 REM             *** TRS-80  "ETCH-A-SCREEN"
                                MAY 1979 BY J.K. SHRUM
20 CLEAR1250:DEFINTA-Z:DEFSTR S:DIMS(15):A=15360:L=A:U=16383:C=1
91:CLS
30 IFINKEY$="D"THEN130
40 REM             *** DRAWING ***
50 B=PEEK(14590):IFB=0ORB=240RB=96THEN30ELSEIFB=128THEN100
60 IFB=80RB=400RB=720RB=1360RB=1680RB=200Y=Y-1:IFY<0Y=0
70 IFB=160RB=480RB=800RB=1440RB=1760RB=208Y=Y+1:IFY>47Y=47
80 IFB=320RB=400RB=480RB=1600RB=1680RB=176X=X-1:IFX<0X=0
90 IFB=640RB=720RB=800RB=1920RB=2000RB=208X=X+1:IFX>127X=127
100 SET(X,Y)
110 IFB>85FORI=1TO10:NEXT:RESET(X,Y)
120 GOTO30
130 S=INKEY$:IFS=""THEN130
140 IFS="T"THEN190
150 IFS="S"THEN290
160 IFS="L"THEN330
170 GOTO 130
180 REM            *** TEXT ***
190 D=PEEK(A):POKEA,C
200 S=INKEY$:IFS=""POKEA,32:FORI=1TO10:NEXT:POKEA,D:GOTO190
210 B=ASC(S)
220 IFB=13POKEA,D:GOTO30
230 IFB>31ANDB<91POKEA,B:A=A+1:IFA>UTHENA=U
240 IFB=8POKEA,D:A=A-1:IFA<LTHENA=L
250 IFB=9POKEA,D:A=A+1:IFA>UTHENA=U
260 IFB=10POKEA,D:A=A+64:IFA>UTHENA=A-64
270 IFB=91POKEA,D:A=A-64:IFA<LTHENA=A+64
280 GOTO190
290 REM            *** SAVE ON TAPE ***
300 FORI=0TO15:S(I)="":NEXT:FORI=0TO15:FORJ=0TO63:S(I)=S(I)+CHR$
(PEEK(L+I*64+J)):POKEL+I*64+J,42:NEXTJ,I
310 FORI=0TO15:S(I)="*"+S(I)+"*":NEXT:CLS:PRINTCHR$(23):PRINT"LO
AD CASSETTE AND POSITION TAPE":PRINT:PRINT"DEPRESS 'RECORD' AND
'PLAY' KEYS":PRINT:PRINT:PRINT:PRINT:INPUT"PRESS 'ENTER' WHEN RE
ADY";Z:PRINT"RECORDING"
320 PRINT#-1,S(0),S(1),S(2):PRINT#-1,S(3),S(4),S(5):PRINT#-1,S(6
),S(7),S(8):PRINT#-1,S(9),S(10),S(11):PRINT#-1,S(12),S(13),S(14)
:PRINT#-1,S(15):GOTO370
330 REM            *** LOAD FROM TAPE ***
340 CLS:PRINTCHR$(23):PRINT"LOAD CASSETTE AND POSITION TAPE":PRI
NT:PRINT"DEPRESS 'PLAY' KEY":PRINT:PRINT:PRINT:INPUT"PRESS 'ENTE
R' WHEN READY";Z:PRINT"LOADING"
350 FORI=0TO15:S(I)="":NEXT
360 INPUT#-1,S(0),S(1),S(2):INPUT#-1,S(3),S(4),S(5):INPUT#-1,S(6
),S(7),S(8):INPUT#-1,S(9),S(10),S(11):INPUT#-1,S(12),S(13),S(14)
:INPUT#-1,S(15)
370 FORI=0TO15:S(I)=MID$(S(I),2,64):NEXT
380 CLS:FORI=0TO14:PRINTS(I);:NEXT:PRINTLEFT$(S(15),63);:POKEU,A
SC(RIGHT$(S(15),1)):GOTO30
```

Program 7. The final product. The variable types have been defined and their values assigned at the beginning of the program. Variable labels have been reused within the program when possible. All spaces in program lines have been deleted and multiple-statement lines are used.

INPUT# statements when reading data tapes. The solution to this one is simple—don't use them when putting text on the screen.

Since we want our program to be as user-interactive as possible, we have to tell our user what to do and the proper time to do it. Lines 320 through 355 accomplish this task by giving instructions in double-width characters for cassette recorder set-up and delaying the program until the user indicates he is ready.

Lines 370 and 420 finally complete the purpose of this module. Since each PRINT# statement can output 255 bytes and each of our strings is 66 characters long (64 screen characters and 2 asterisks), we can comfortably put three strings on tape with each PRINT# statement. That leaves us one string left over requiring its own PRINT# statement. On my machine, lines 370 through 420 take about 40 to 45 seconds to complete their execution.

Line 430 completes our save module by branching into the load module, redisplaying our screen pattern, and returning us to drawing.

Load Module

Program 6, the load module, will read our tapes back into the machine.

Lines 520 through 560 tell the user how to prepare the cassette recorder and give him time to do it.

Lines 580 through 600 then blank our string array just as it was done for the save module.

After these preparations, lines 610 through 660 read the previously recorded strings from the tape, using exactly the format for the INPUT# statements as was used for the PRINT# statements. The formats must be identical to avoid blowing the data read.

Now that the data is back in

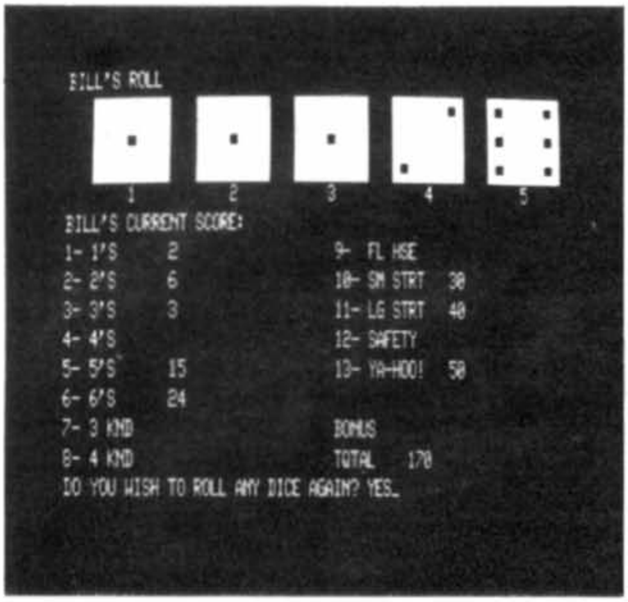


Photo 1. Etch-A-Screen was used to format this output display for a graphics game program. Much easier than trial and error.

the S$ array, we have to get rid of our friends the asterisks. This is done in lines 670 through 690, using the MID$ function to pull our original strings from the middle. If you remember, line 670 was also accessed from the save module, to strip the asterisks from the strings for redisplay after saving.

Now, all we have to do is display the data.

Since each element in the S$ array is 64 characters long, it generates a carriage return. Each PRINT statement, as well, triggers a carriage return unless overridden. Semi-colons will let us avoid double-spacing the first 15 lines, but a semi-colon won't override a carriage return when a character is printed in the very last screen position in the bottom right corner.

Accordingly, we clear the screen and print the first 15 display lines with program lines 705 through 730. Now we print all but the last character in line 750. Once here, we get sneaky by taking the code number of our very last character and putting it directly into the proper screen memory address with line 770. This avoids a carriage return and leaves us with our display intact.

At the end, we return to drawing with line 780.

Operating the Complete Program

The finished program is no harder to use than the individual modules. The program runs the drawing, module first. Typing D for done drawing lets you choose the module you want next. D followed by T puts you into text, while D followed by S or L lets you save or load, respectively.

Both save and load return you to drawing automatically, while to return to drawing from text you must tape the ENTER key. One caution: You cannot go from text directly into save or load—you must hit ENTER, then D, then S or L.

Compacting the Program

As I said at the beginning, we can make this program usable on a 4-K machine by condensing it with a few memory saving tricks. Reducing the memory any program uses pays a double bonus, even if your machine has a great amount of memory. First, memory-efficient programs usually run faster and second, they store on less tape and CLOAD faster.

The first and most obvious step is to delete remarks (REM). *Be certain you have the documentation maintained elsewhere.*

I usually keep two separate versions of my programs, one expanded and the other compacted that I can run. When I got my printer, I also started keeping a hard copy of the expanded version.

Another memory saving step is to remove all unnecessary spaces from program statements and use multiple statements per line when possible. This might make the condensed version harder for you to read, but your computer will be very grateful and will show its gratitude by increasing the program's execution.

More memory savings can be had by using the most precise variables and defining them at the beginning of the program. For instance, by defining variable S as a string, we save a byte every time we avoid typing S$ and use S. This can be a real saver, especially if you use integer variables defined the same way.

Finally, use variables for constant values, use the same variables in different parts of the program and you'll save yet a few more precious bytes.

Program 7 is our final product: our Etch-A-Screen program condensed like crazy. You'll see I used the memory savers we just discussed. This final version leaves about 300 bytes (not a lot, but enough) unused when running in a 4-K memory.

The program has been renumbered by 10's to accommodate those who like to use automatic line numbering for program entry, and has been listed in 64 characters per line format. This way, what shows on the screen should be identical with the printed listing.

Conclusion

I've included a couple of samples to help you in Photos 1 and 2. The first photo is the display format for a dice game I wrote based on Yahtzee. Believe me, it made the entire programming process go much more easily. Photo 2 is just a quick sketch I made of my house plan.

Can this program be further improved? Most assuredly. How about a routine to display instructions to the user? Or, change the program to use command drawing. In other words, type in the appropriate commands to tell the computer you want a line drawn 25 units to the right and then sit back and watch the computer work.

The list of modifications and improvements is endless. Should anyone come up with any they would like to share, I'd like to hear about them. ■

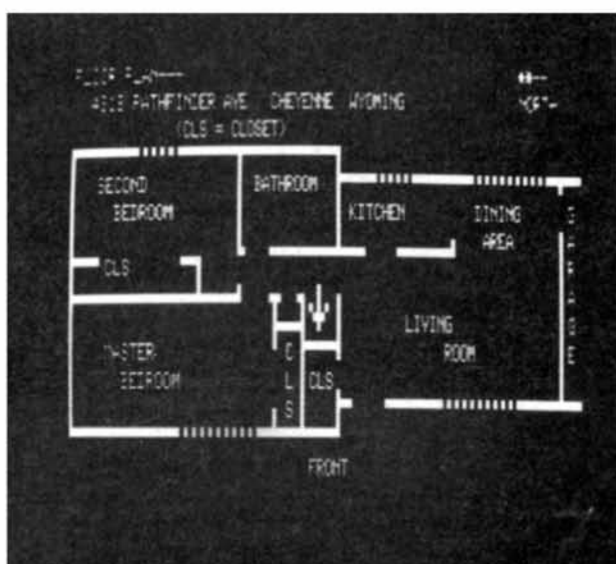


Photo 2. A quick sketch (it took about ten minutes) of our house. Notice in the examples that I did not use commas in the address heading. This avoids problems with INPUT# statements.

✓Reader Service—see page 162

Edit hybrid programs with ease, let the 80 do it!

Progdata

Steve Kelley
9506 Peach St.
Oakland, CA 94603

Rule number one: Nothing should ever be entered more than once. Ever!! A good rule to display over your keyboard somewhere, prominently.

Once programs or data are entered, the process need never be repeated. Text changes are made with an editor. Relocation of an assembly language program is done by reassembling it at a different address. Adding a line to a BASIC program is simplest of all. Just enter the line. Room is made for the new line automatically at the right location.

No retyping entire programs; no housekeeping to worry about; no hassle.

Now the "GOTCHA." How do you edit the hybrid program—part machine language and part BASIC. How are they combined?

Converting to BASIC

Since BASIC deals primarily with decimal numbers, it is customary to create and edit the machine language part of the program with an editor/assembler. The machine language program must then be converted to decimal numbers and put into BASIC DATA statements. A FOR-NEXT loop is also necessary to convert these DATA statements back to the machine language program.

When the BASIC program RUNs, the FOR-NEXT loop READs the DATA statements and POKEs the numbers into the proper locations. This is done only once because the resulting machine language program stays in memory, even if the BASIC program that POKEd it there is cleared and another is run. A USR statement calls the machine language routine from a BASIC program.

Until now, converting the machine language program to BASIC statements meant changing a list of HEX numbers to their decimal equivalents, manually typing them into DATA statements and providing the FOR-NEXT loop. The program shown in Listing 1 eliminates most of this busy-work because it follows rule number one.

After your program is assembled and the object code (machine language program) is loaded, the computer has all the information necessary to write its own BASIC program complete with FOR-NEXT loop and DATA statements. The BASIC program in Listing 1 does just that, and even stores the resulting program on disk. This computer-made program runs on its own or can be RENUMbered and MERGEd with any BASIC program.

Lines 60 and 70 input the starting and ending addresses respectively, which lines 90 and 100 utilize when generating the FOR-NEXT loop statements. Lines 130-190 PEEK at the machine language program and add the resulting decimal numbers to the current DATA statement until it exceeds 55 characters. Line 140 starts each DATA line with a line number and the word DATA. The completed DATA line is PRINTed on the display and on the disk by line 180. This process repeats itself until the entire machine language program is converted to BASIC. Lines 240-360 make up the routine for converting the hex addresses to decimal. The routine doesn't accept illegal characters or numbers that are too large for addresses. Also, it ignores leading zero's and trailing H's and converts numbers greater than 32767 to the negative numbers necessary for BASIC.

```
10 '              ***     PROGDATA     ***
20 CLEAR 2000
30 E$="TEMPDATA/BAS"
40 HX$="H0123456789ABCDEF"
50 CLS
60 D$="STARTING ADDRESS (HEX) ":GOSUB 240:ST=IN
70 D$="ENDING ADDRESS (HEX) ":GOSUB 240:EN=IN
80 OPEN "O",1,E$
90 A$=" 10 FOR Z8="+STR$(ST)+" TO"+STR$(EN)
100 A$=A$+":READ Z9:POKE Z8,Z9:NEXT Z8"
110 PRINT A$:PRINT#1,A$:A$=""
120 LN%=20
130 FOR Z=ST TO EN
140 IF A$="" THEN A$=STR$(LN%)+" DATA":LN%=LN%+10
150 B$=STR$(PEEK(Z))
160 IF RIGHT$(A$,4)<>"DATA" THEN MID$(B$,1,1)=","
170 A$=A$+B$
180 IF LEN(A$)>55 PRINT A$:PRINT#1,A$:A$=""
190 NEXT Z
200 IF A$<>"" PRINT A$:PRINT#1,A$:A$=""
210 CLOSE
220 PRINT"PROGRAM IS IN DISK FILE NAMED "+CHR$(34)+E$+CHR$(34)
230 END
240 PRINT D$;:C$="":INPUT C$
250 IF LEN(C$)<1 THEN 240
260 IN=0
270 FOR Z=1 TO LEN(C$)
280   Z1=INSTR(HX$,MID$(C$,Z,1))
290   IF Z1=0 THEN 240
300   IF Z1=1 THEN 320
310   IN=IN*16+Z1-2
320 NEXT Z
330 IF IN>65535 THEN 240
340 IF IN>32767 THEN IN=IN-65536
350 PRINT"DECIMAL =";IN
360 RETURN
```

Listing 1.

One Suggestion

If you don't have a disk, try changing line 180 to LPRINT the line instead of PRINTing it on disk, and take out the OPEN and CLOSE statements. You still have to type in the program but it's handier than a completely manual conversion. The program also converts graphic displays or text to DATA statements although there are more efficient methods. ■

Use your 80 to work out model measurements.

Model Conversion

Tom W. Blackburn, III
15593 Paseo Jenghiz
San Diego, CA 92129

Colossus came into our home solely to provide diversion and intellectual stimulation. This he accomplished beyond our greatest expectations. Though utility was not a requirement, any time he could be of assistance was certainly appreciated. Colossus is a most welcome Radio Shack TRS-80, Level II, 16K microcomputer.

To date Colossus has taken over such household tasks as maintaining our address book, serving as a calendar of events and most importantly, keeping me off the streets at night. His latest accomplishment grew out of Christy's most recent project.

Not long ago, Colossus discovered Christy sitting at the desk surrounded by little pieces of wood, paints and a vast assortment of small tools. She was going to build and furnish a new house, an adult doll house—a showcase built with great precision and detail, adhering to the traditional scale of one inch to the foot.

Conversions

Furniture and fixtures soon began to appear. However, a fly landed in Christy's ointment. She has always ranked math right up there with trips to the dentist and ingrown toenails. Now, suddenly, she was encountering an endless series of conversions, feet and inches reduced to the 1/12th scale.

Colossus began to study her project and did not rest until he could make the desired conversions in the blink of his cycloptic eye. The following is a description of Colossus' new-found knowledge.

A quick inventory of the measuring instruments in the house revealed that like most of the rest of this country—we had not made the switch to the metric system. This meant Colossus had to train himself to handle fractions. He made the arbitrary decision that 1/64 of an inch was the highest degree of accuracy with which he cared to contend. Scaling measurements down to 1/12th of the original might not be sufficient, he guessed. He would have to scale upward as well—as a check of the miniature's faithfulness.

The Program

Lines 20-60 set up the program. The variable M$ in line 40 is used as the output format for displaying the scaled measurements utilizing the PRINT USING statement. The two variables M$(1) and M$(2) indicate whether Colossus has scaled the given measurement up or down and supply these two words as appropriate at display time. The variable F in line 20 is set equal to the decimal rendering of 1/64. This variable is used later for the conversion from decimal to fraction.

Colossus asks in line 50 for your guidance as to whether he should scale upward or downward. This question and your response is wiped out by the two PRINT CHR$ (27)'s which begin line 70. The rest of the line requests the measurement which is to be scaled—feet and inches separated by a comma.

Fractions of an inch are entered in standard notation; e.g. 5/8. Separate the whole inches from the fraction, if any, with a space.

A measurement can be entered in inches only (preceded by a comma) without converting it to feet and inches; e.g. 58 inches instead of 4 feet 10 inches.

Lines 80-130 break down the measurement. As applicable, Colossus assigns values to these variables:

F1 = number of feet
I1 = number of whole inches
N = numerator of inches fraction
D = denomonator of inches fraction

This disassembly process involves two loops each searching, character by character, the string that contains the inches portion of the given measurement; i.e. I$. The first loop, line 90, looks for a slash (/). This signifies that a fraction was included in the inches measurement.

If no slash is found, Colossus assumes whole inches only and bypasses the rest of this section. The second loop, line 100 starts at the position of the slash found in *line 90* and searches backward for a space. If no space is found, he as-

```
20 CLS:F=.015625:M$(1)="UP":M$(2)="DOWN":S=12
30 PRINT"MINIATURES CONVERSION"
40 M$="SCALED %  % DIMENSION:     ### FEET    %          %
      INCHES"
50 PRINT:INPUT"SCALE UP OR DOWN";B$
60 F$="":I$=""
70 PRINTCHR$(27);CHR$(27):INPUT"ENTER MEASUREMENT (FEET
      , INCHES):";F$,I$
80 F1=VAL(F$):LI=LEN(I$):N=0:D=1:I1=0
90 FORX=1TOLI:IFMID$(I$,X,1)<>CHR$(47)THENNEXT:I1=VAL(I
      $):GOTO160
100 FORY=XTO1STEP-1:IFMID$(I$,Y,1)<>" "NEXT:GOTO130
110 I1=VAL(LEFT$(I$,Y-1))
120 IFI1>=12THENF1=F1+INT(I1/12):I1=I1-(INT(I1/12)*12)
130 N=VAL(MID$(I$,Y+1,X-Y-1)):D=VAL(RIGHT$(I$,LI-X))
140 X=LI:IFY<1NEXT:GOTO170
150 Y=1:NEXTY,X
160 IFI1>=12THENF1=F1+INT(I1/12):I1=I1-(INT(I1/12)*12)
170 IFLEFT$(B$,1)="U"THEN270
180 Z=2:X=0:Y=64:I2=(I1+N/D)/S:F2=INT(F1/S):I1=I2+(F1/S
      -F2)*12:I2=I1-INT(I1):IFI1=0THENI1$="   "ELSEI1$=S
      TR$(INT(I1))
190 F1=F2:I2=I2-INT(I2):IFI2=0THEN260
200 X=INT(I2/F+.5):IFX=0THEN260
210 IFX/2=INT(X/2)THENX=X/2:Y=Y/2:GOTO210
220 X$=STR$(X)
230 Y$=STR$(Y):Y$=RIGHT$(Y$,LEN(Y$)-1)
240 IFX=1ANDY=1THENI1=I1+1:I1$=STR$(I1):GOTO260
250 PRINTUSINGM$;M$(Z),F1,I1$+X$+"/"+Y$:GOTO50
260 PRINTUSINGM$;M$(Z),F1,I1$:GOTO50
270 Z=1:F1=F1*S+INT(I1*S/12):I1=I1*S-(INT(I1*S/12)*12)
280 I1=I1+INT(N*S/D):I2=(N*S/D)-INT(N*S/D):IFI1=0THENI1
      $="   "ELSEI1$=STR$(I1)
290 IFI2=0THEN260
300 X=INT(I2*64+.5):Y=64:GOTO210
```

Program listing.

sumes that only a fraction was given for the inches portion of the measurement.

Lines 140 and 150 reset the search loops appropriately. This is necessary since finding the search key causes Colossus to exit the loop. Without these two lines, the search loops would have been left active. In a repeating program such as this, memory will slowly be eaten up by these abandoned active loops.

Lines 120 and 160 check the inches in excess of 12 or one foot and convert them to feet and inches appropriately, adding any additional feet generated to the number of feet input.

Colossus is many things, but psychic he is not. Therefore, in line 170 he checks your earlier instructions as to whether he is to scale upward or downward. Up to this point the routines are compatible for either conversion.

Scaling upward causes a branch to line 270.

Scaling downward is performed in lines 180-240. The actual scaling takes place in the first two lines (180 and 190) and the rest are used to convert any decimal fraction to a conventional fraction. This is accomplished by dividing the decimal fraction by F (.015625). The rounded integer yields the number of 64ths in the decimal.

Colossus then repeatedly divides by 2 to reduce the fraction as far as possible (e.g. 32/64 = 16/32 = 8/16 = 4/8 = 1/2). This reduction is performed in lines 210–240 and is used in both upward and downward scaling.

Lines 270-300, which control an upward scale, multiply the measurement by 12 and create a new fraction with a denominator of 64, if required. He then goes back to lines 210–240 to reduce the fraction.

Conclusion

At this point Colossus need only tell us the answer. If no fraction is involved this is done with line 260. If a fraction is displayed, line 250 if used. The output variables are:

F1 = feet
I1$ = inches
X$ = numerator of inches fraction
Y$ = denomonator of inches fraction

Colossus opted for string variables to display the inches so that he could concatenate them, thereby suppressing the leading and trailing spaces found in integer variables.

Also by changing the value of variable S in line 20, the scaling factor is changed. If S were changed to 6 for example, the scaled measurements would be converted with the ratio of 2 inches to the foot (12/2 = 6).■

Produce program sounds through your cassette recorder.

Super Sound

David G. Morr
6599 Red Fox Rd.
Reynoldsburg OH 43068

Adding the ability to create sound effects or even a limited music ability to the Radio Shack TRS-80 Level II computer is easy and free! It is easy because there are no hardware modifications required, and your BASIC programs require only three additional lines of code to enable the sound output. It is free because the software is given in this article, and the audio output can come from your cassette player.

The sounds that your TRS-80 can make with this program are limited only by your imagination . . . from the beep and boop of ping-pong video games to the motor putt-putt of car motors or space-game phasor sounds. Single-voiced music sounds are also possible.

What It Does

The machine-language routine shown in Fig. 1 is used to operate the audio output interface normally used to write programs and data to the cassette recorder. The software provides a means to control how often the dc voltage available at the output is toggled from .85 volts to .05 volts and back again. In addition, the software is used to determine the length of the transitions. This means the frequency and duration of tones are readily defined by your BASIC program.

Using an oscilloscope, I measured the frequency and duration of the Super Sound output. Figs. 2a and 2b are tabulations of what was measured. The shortest and longest durations are 7 milliseconds and 1.753 seconds, respectively. The highest and lowest frequencies are 14.3 kHz and 75 Hz, respectively. Producing these different sounds from your Level II BASIC programs is quite easy, as we shall see.

Software Description

Fig. 3 is the flowchart for Super Sound. The routine is called from BASIC by the USR(X) command. The value X is passed to the routine when the USR(X) statement is executed and is converted to a 16-bit signed binary integer by the CALL 0A7FH instruction. This leaves the H and L registers of the Z-80 CPU with the binary value. The value in the H register is used by the routine to fix the length or duration of the sound output, and the value in the L register is used to determine how often the output will be toggled during the output.

The fundamental timing of the routine is determined by the C register. This register is continually decremented, while it uses the B register to determine when to toggle the audio, until it reaches 0, then the H register is decremented and tested for 0. If the H register is not 0, then the C register will be decremented as before until it reaches 0 again. The routine is finally exited when the H register equals 0. Since the H register is decremented before being tested, the smallest H value that will result in the shortest tone duration is 1.

The value in the L register is unchanged by the routine. Its content value is, however, continually loaded into the B register and used as a down counter during the C register decrementing mentioned above. Every time the B register reaches 0 the audio output is toggled and the B register is reloaded from the L register. The smaller the number in the L register, the higher the sound frequency will be (L = 1 results in the highest frequency).

The shortest-duration/highest-frequency sound occurs with the H register = 01 and the L register = 01 or the combined HL register = 0101 hexadecimal. To convert 0101H to a decimal value, multiply the H value by 256 and add the L value. This results in 257 decimal. The longest-duration/lowest-frequency sound occurs with the combined HL register = 0FFFFH (actually, 0 is a bit longer and lower).

The Super Sound routine is written in all relative code. With the Z-80 CPU, relative code translates to a special set of jump instructions that do not specify the address value of where they are to go. Instead, they use a single byte displacement value (either + or −) from the current memory location. The advantage of relative code is that it can be located anywhere in memory without changing any of the program statements or values. The ability to use relative addressing is one of the important advantages of the Z-80, and its use with this routine reduces the size of the subroutine and makes it easy to use.

Implementation

In order to produce the sound from your computer, you must connect an audio amplifier and speaker to the cassette output plug of your TRS-80. Since the voltage out is less than 1 volt peak to peak, *you can use almost any sound amplifier* (do not use any amplifier that is not

```
0000  CD 7F 0A     ENTER   CALL  0A7FH
0003  3E 01                LD    A,1
0005  0E 00                LD    C,0
0007  ED 5B 3D 40          LD    DE,(403DH)
000B  45           LOAD$B  LD    B,L
000C  2F                   CPL
000D  E6 03        LOOP    AND   3
000F  B3                   OR    E
0010  D3 FF                OUT   (0FFH),A
0012  0D                   DEC   C
0013  28 04                JR    Z,DN$TST
0015  10 F6                DJNZ  LOOP
0017  18 F2                JR    LOAD$B
0019  25           DN$TST  DEC   H
001A  20 F1                JR    NZ,LOOP
001C  C9                   RET
```

Fig. 1. Assembly-language listing of Super Sound subroutine.

Frequency Value FV	Tone Frequency in Hertz	Frequency Value FV	Tone Frequency in Hertz
1	14300	19	1000
2	8200	20	943
3	5750	25	758
4	4445	30	633
5	3571	35	544
6	3030	40	476
7	2632	50	381
8	2299	60	317
9	2062	80	240
10	1869	100	192
11	1695	125	153
13	1450	150	146
15	1250	200	96
17	1111	255	75

Fig. 2a. Frequency values.

Duration Value DU	Time Duration in seconds
1	.007
5	.035
10	.069
25	.173
50	.345
100	.690
127	.876
0	1.753

Fig. 2b. Duration values.

To generate a tone, pick a frequency value, FV, from Fig. 2a and a duration value, DU, from Fig. 2b. Generate the tone by executing the BASIC statement, SS = USR (256•DU + FV).

transformer isolated from the 120 volt ac line).

One possible amplifier is your cassette recorder. To use your cassette unit: (1) disconnect the motor control cable from the recorder; (2) remove any cassette from the machine; (3) open the cassette door and press the small lever at the back left of the cassette well towards the back of the well; (4) press the RECORD and PLAY buttons simultaneously; and (5) plug a speaker into the earphone jack. Any audio output from your computer can now be heard from the speaker.

Using your cassette unit and a speaker is a reasonable means for audio output from your computer. An even better way is to use a gadget that your local Radio Shack sells: the speaker/amplifier, part number 277-1008. This small unit has an input jack that is plug compatible to the cassette cable from the TRS-80. So to use it, all you need do is unplug the cable from the MIC input of your cassette unit and plug it into the speaker/amplifier, turn it on, and you're in business.

The BASIC program is one example of how to use the Super Sound routine. Before the program is loaded, the MEMORY SIZE? prompt must be set equal to one less than the decimal address of the first byte of the routine. In the program the value is set to 32737, allowing the routine to be placed at the top of a 16K RAM TRS-80. If you have 4K memory, then answer the prompt with 20447 and change line 110 or line 1010 to read AD = 20448.

Each time line 230 or line 1040 is executed in the program, a random duration value is added to a random frequency value. The sum is then passed to the USR( ) routine, and the tone is generated. Subsequent loops of the programs will generate differing tones of differing lengths.

Example Game Program

The Race program shows an example game program using the Super Sound routine. The program is a heavily modified adaptation of the original, which was conceived by Bill Louden, a local Radio Shack store manager. In the game, a car appears to be traveling down an ever-curving road that wanders back and forth across the screen. The road becomes narrower as the game progresses. The car continually makes a putt-putt sound, and the player must steer the car to avoid running off the road or falling into potholes. Appropriate sound effects are added if misadventure strikes. The program takes less than 2.5K of memory.

Whenever the statement SS = USR(xx) appears in the program, a tone will be generated. The length and duration of the tone are set by the value xx as described earlier in this article. All the xx values in the program were derived from experimenting and playing with the game. The first few lines of the program are used to load the USR( ) subroutine as in the BASIC program. Spend some time studying Race to get some ideas for your own programs.

A Challenge

I wrote this article not only to

```
10 '  PROGRAM TO DEMONSTRATE THE USE OF "SUPER SOUND"
20 '
30 '  AUTHOR: D.G. MORR
40 '
50 '
60 '  THE MEMORY SIZE PROMPT MUST BE ANSWERED WITH A
70 '  NUMBER EQUAL TO 1 LESS THAN SUBROUTINE ADDRESS, AD
80 '
100 DEFINT A-Z: ' INTEGER VARIABLES RUN FASTER WITH BASIC
110 AD = 32738: ' RESERVED MEMORY ADDRESS FOR ROUTINE
                  (CHANGE TO 20448 FOR 4K RAM SYSTEM)
120 HI = INT(AD/256): POKE 16527,HI: ' SET UPPER BYTE OF USR() ADDRESS
130 LO = AD-HI*256: POKE 16526,LO: ' SET LOWER BYTE OF USR() ADDRESS
140 FOR I = AD TO AD+28: ' LOAD RESERVED MEMORY WITH ROUTINE
150 READ DT: POKE I,DT
160 NEXT I
170 DATA 205, 127, 10, 62, 1, 14, 0, 237, 91, 61, 64, 69, 47,
        230, 3, 179, 211, 255, 13, 40, 4, 16, 246, 24, 242, 37,
        32, 241, 201
180 '
200 '  DEMONSTRATION EXAMPLE OF RANDOM TONE GENERATION
210 DU = 2048*RND(10): ' DURATION MUST BE INTEGER MULTIPLE OF 256
220 FV = 5+RND(100): ' FREQUENCY VALUE MUST BE NON-ZERO INTEGER
230 SS = USR(DU+FV): ' GENERATE TONE
240 GOTO 210: ' GO AND DO IT AGAIN
250 END
960 '
970 '
980 ' ****** ANOTHER WAY OF EXPRESSION ******
990 '
1000 DEFINTA-Z
1010 AD=32738: HI=INT(AD/256): POKE16527,HI: POKE16526,AD-HI*256
1020 FOR I=AD TO AD+28: READ DT: POKE I,DT: NEXT
1030 DATA 205,127,10,62,1,14,0,237,91,61,64,69,47,230,3,179,211,
     255,13,40,4,16,246,24,242,37,32,241,201
1040 SS=USR(2048*RND(10)+5+RND(100)):GOTO1040
```

BASIC program to generate random tones.

make a couple of bucks, but also to try to stimulate those *Microcomputing* readers lucky enough to own TRS-80 computers to write some super programs that can make use of the ability to create sound effects. I hope that it is apparent from reading the article how easy and simple it can be.

If you would like to avoid typing the Race program into your computer, I can provide a prerecorded cassette for $6. Send a check or money order to D. G. Morr, 6599 Red Fox Rd., Reynoldsburg OH 43068, with a note requesting the Race program and include your name and address.■

```
1 REM
2 REM   THE GAME OF RACE AS MODIFIED BY DAVE MORR,
3 REM
4 REM            MEMORY SIZE MUST BE SET TO 32737 FOR 16K RAM
                                            20447 FOR  4K RAM
5 REM
6 DEFINTA-Z:CLEAR100
8  AD=32739:HI=INT(AD/256):POKE16527,HI:POKE16526,AD-HI*256:
    REM CHANGE AD TO EQUAL 20448 FOR 4K RAM SYSTEM
10 FOR I=AD TO AD+28: READ DT: POKE I,DT: NEXT
12 DATA 205,127,10,62,1,14,0,237,91,61,64,69,47,230,3,179,211,255,13,40,
   4,16,246,24,242,37,32,241,201
15 CLS:PRINTCHR$(23):PRINT@266,"TOUCH RIGHT ARROW(":CHR$(94);")":PRINT@342,
   "TO STEER RIGHT":PRINT@464,"OR LEFT ARROW(":CHR$(93);")":PRINT@534,
   "TO STEER LEFT"
16 PRINT@776,"PRESS ENTER WHEN READY"
18 A$=INKEY$:IFA$=""THENPRINT@84,"*** RACE ***":SS=USR(5119):PRINT@84,
   "             ":SS=USR(5115):GOTO18
20 SC=0:SD=0:DS=290:A=16024:B=20:C=8:D=1:F=0:G=25:H=46:KY=14400:RI=64:
   LE=32:ST=15999:P1=255:LC=153:RC=166:CLS
25 FORI=0TO15:PRINTSTRING$(B,"*");TAB(B+C)STRING$(64-B-C,"*");:NEXT
30 X=PEEK(KY):IFXANDRITHENG=G+1:GOTO50
40 IFLEANDXTHENG=G-1
50 IFRND(8)=3THEND=-D
60 B=B+D:IFB<3THENB=3:D=-D
70 J=B+C:IFJ>60THENB=60-C:J=B+C:D=-D
100 POKEA,H:POKEA+1,H:A=ST+G:SS=USR(DS):PRINTSTRING$(B,42);
105 IFRND(8)=3THENPRINTSTRING$(RND(C),32);"0";TAB(J)STRING$(64-J,42);
    ELSEPRINTTAB(J)STRING$(64-J,42);
110 P1=PEEK(A):P2=PEEK(A+1):IF(P1=32)AND(P2=32)THEN120ELSEIF(P1=79)
    OR(P2=79)THEN160ELSE140
120 POKEA,LC:POKEA+1,RC:F=F-1:IFF<1THENF=40:C=C-1:DS=255+2*C:IFC<6THEN200
125 GOTO30
140 SC=SC+1:G=INT(C/2+B):PRINT"YOU RAN OFF THE ROAD!"
150 FORT=0TO50STEP10:FORI=277TO267STEP-1:SS=USR(I+T):NEXTI:NEXTT:FORI=1019
    TO1023:PRINTSTRING$(B,42);TAB(B+C)STRING$(64-J,42);:SS=USR(I):NEXTI:
    GOTO30
160 SD=SD+1:PH=572+B+C/2:FORT=0TO30:PRINT@PH,"POTHOLE";:SS=USR(900):PRINT@PH,
    "       ";:SS=USR(820):NEXT:PRINT@960,;:GOTO30
200 CLS:PRINTCHR$(23):PRINT:PRINT:PRINT"GAME OVER":PRINT:PRINT:
    PRINT"YOU RAN OF THE ROAD";SC;"TIMES"
205 PRINT:PRINT:PRINT"AND FELL INTO";SD;"POTHOLES!"
210 PRINT:PRINT:PRINT"WANT TO PLAY AGAIN (Y OR N)?"
220 A$=INKEY$:SS=USR(2590*RND(5)+RND(5)):IF(A$="Y")OR(A$="N")THEN225
    ELSE220
225 IFA$="Y"THENRUN
250 CLS:X=0:FORI=0TO50:R=RND(10):X=X+3*R:PRINT@X,"BYE";:SS=USR(768+R):NEXT:
    FORI=511TO257STEP-1:SS=USR(I):NEXT:CLS
```

Race program using the Super Sound subroutine.

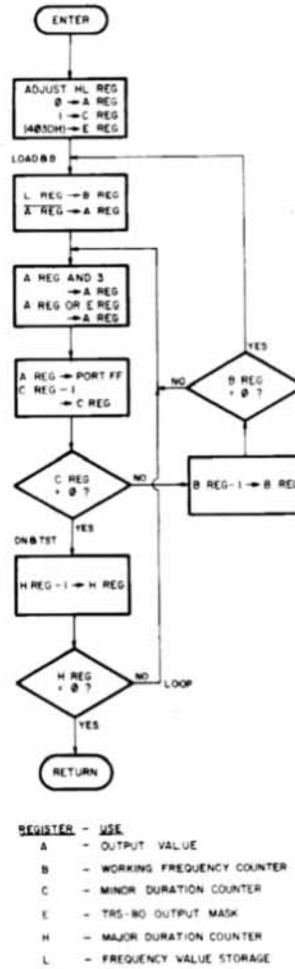


Fig. 3. Super Sound flowchart.

Use block movement to re-order your programs.

Cutting and Splicing BASIC

R. B. Nottingham
c/o The Book Rack
1619 SE 3rd Ct.
The Cove
Deerfield Beach, FL 33441

The technique of loading a program to follow an existing program ("Whip File Wipeouts," by Roger L. Pape, *Microcomputing*, July 1979, p. 39) offers several advantages to the TRS-80 programmer. Reducing keyboard entries to save time and minimize errors is one and storing a library of routines to insert at any point in a program is another.

Suppose we have a program divided into three blocks. Block A is numbered from 10 to 990, block B 1000–1990 and block C 2000–2990. We wish to re-order the program so that instead of ABC the sequence is ACB. Normal renumbering techniques will not permit this.

Load a renumbering routine and then the program. Delete 10–990 and 2000–2990. The remaining block B is renumbered and moved from 1000–1990 to 2000–2990. Save the renumbered block B on tape.

Now, reload the program, delete block B (Delete 1000–1990) and renumber the program from the beginning so that it is now numbered 10–1995. Then using Mr. Pape's method, load block B. Presto! ACB. This may seem like a lot of work, but it saves both the time of re-entering 100 lines of code and the errors.

Linear regression and graphing data with either a horizontal or vertical axis are two routines which may be useful in a variety of programs. We can now load these, either in the process of entering a program or after the program is completed and de-bugged. (Being able to de-bug a program in small sections is often advantageous.)

One word of caution in adding material as part of the main program or as subroutines: Variables must be consistent. Try adding a line at the beginning and end of the section to translate: A = Z, B = Q, R = Y, etc., followed by Z = A, Q = B, Y = R, etc.

To refresh your memory, the splicing procedure is as follows:

```
PRINT PEEK (16633), PEEK (16634)
```

This determines the end address of the program in memory. The computer will display two numbers:

```
          206          74
POKE 16548, 204 : POKE 16549, 74
```

If two cannot be subtracted from the first number, add 256 before subtracting and decrease the second number by one.

```
CLOAD (program to be added)
READY
POKE 16548,233 : POKE 16549, 66
```

Now LIST the program and RENUM if required. ■

3 FEET BY 2 FEET

GIANT! - VIDEO SCREEN DISPLAY WORKSHEET

PRINTED ON ENAMEL STOCK AND THEN VARNISHED SO IT IS WASHABLE
---No more looking in Level II Manual for Video Layout.

$3.95 plus 1.50 postage

A MUST FOR SERIOUS PROGRAMERS!

TRS-80* Registered Trademark of Tandy Corp.

Business Proven — **ALSO** —

CHECK WRITER!

NEW

- TRS-80* 32K - 2 Disk Drives (No Printer Required)
- 8 Accounts Per Check
- Balances Check Book - Keeps Current Balance
- LIST OUT Checks, Accounts or Deposits (To Screen or Printer)
- 100 Accounts
- Complete Edit Routines

39.95 plus 1.50 Postage

CASH REGISTER

48K - 4 Disk Drive
Handles Cash Sale, Charge Sale, Deducts from Inventory, Post to Charge Accounts - Gives Receipt, Prints 2 Line Message at Bottom of Receipt.

GOLF HANDICAPPER!

Golf Club Proven

- TRS-80* 32K - 1 Disk Drive - Printer
- 100 Members
- Figures Golf Handicaps by USGA (USGA Approved Method)
- Prints Handicap Card for Each Member
- Sorts all Members by Name, lists Handicap, Number of Games Played, Latest Date Played ...A MUST FOR TOURNAMENT PLAY!

$24.95 plus 1.50 Postage

COMPLETE DOCUMENTATION!!!

CRISS-CROSS

48K - 4 Disk Drive
Total 5400 Names, Address Phone Numbers - Puts in Phone Number Order.

COMING SOON!!
MAILING LABEL PULLER
(Makes Addressing a Snap)
HOLE PUNCH FOR BINDING COMPUTER MAGAZINES

VISA · master charge

MICON MICRO SYSTEMS ✓212
P.O. Box 360
Azle, Texas 76020
Ph.817 444-1719 ext. 2

All Texas Residents add 5% Sales Tax

"HISPĒD"

TAPE OPERATION

A NEW SOFTWARE PROGRAM for the TRS-80™

- Seven speed ranges
- Saves, Verifys and loads array data or programs (BASIC of SYSTEM)
- Can transfer array data to tape up to 30 times faster than normal operation using PRINT#
- Printer control features—Auto carriage return, line spacing, page spacing, screen image

Write for full specifications or send $24.95 (ck or mo) + $1.00 P/H (Calif. residents add 6% sales tax)

VISA · master charge

PALOMAR SOFTWARE

170 S. Palomar Dr.
Redwood City, Ca. 94062
✓228

"TRS-80 is a registered trademark of TANDY CORP."

QUICKSET

GET YOUR WARP SPEED UP

FOR REALLY FAST TRS-80*

GRAPHICS

- A cassette SOFTWARE addition to your TRS-80* level II BASIC.
- Lets you draw graphics with your computer the same way you draw on paper.
- All BASIC commands work normally.
- Draw shapes, large alphabetical characters, etc. *ten* times faster, more easily and using less memory then level II.
- SETs one graphic spot at a time.
- Does not use strings or POKE.
- Paint your TV screen white in 4.5 seconds (TRS-80 takes 45 seconds).
- Draw the starship ENTERPRISE, above, beginning at any point on the screen, in 0.3 seconds (TRS-80 takes 4 seconds and uses twice as much memory).
- No effect on your warranty.

WINKING CURSOR

- Play this cassette program into your TRS-80* and the cursor starts winking.
- It keeps on winking until you turn the machine off (not affected by NEW).

QS-1 QUICKSET on cassette	$7.95
WC-1 WINKING CURSOR on cassette	$4.95
QW-1 Both on one cassette	$9.95

Add $1.00 per order handling charge.
(PA residents—add 6% tax) ✓249

JMS Corp
P.O. Box 18083 Pittsburgh, PA. 15236
(412) 655-7017

* TRS-80 is a trademark of the Radio Shack Division of Tandy Corporation.

GIN!

YOUR TRS-80 CAN BE A TOUGH OPPONENT

GIN RUMMY 2.0 plays a strong game, good enough to challenge an expert player. Plays a full regulation game, keeps score to game level, allows rearrangement of player's hand, and changes strategy to counter its opponent's play. Hours of good card playing, a fascinating program, one you'll enjoy playing against and trying to beat. MGR-1 $14.95

CHECKBOOK PLUS solves the problem of monthly bank statement reconciliation. No cumbersome tape record keeping. Just do your checkbook once a month and let Checkbook Plus handle all the details and find the errors. MCB-1 $9.95

CALCULATOR PLUS is an on-screen or printing calculator, with chain and mixed calculations, memories for answer storage or calculations with constants, item count and on-screen or printed review of long add-and-subtract operations. Optional dollar format. MPC-1 $9.95

CHECKBOOK PLUS and **CALCULATOR PLUS** both on one cassette. MC-2 $14.95

THE LISTMAKER Powerful, versatile program allows entry of 400 names with codes in 16K. Lists by code on-screen or to printer. Sorts, provides editing of name or code, dumping, loading lists. Ideal for clubs, organizations, small businesses, individuals. MLM-1 $9.95

✓90

MANHATTAN SOFTWARE, Inc.
P.O. Box 5200 Grand Central Station
New York City, New York 10017

✓Reader Service—see page 162

Why POKE your screen onto the line printer when you can use this flexible routine?

LPVIDEO

David E. Powers
10 Wilben Ct.
New Hyde Park, NY 11040

Often I want hard copy of everything on my TRS-80 video screen. A favorite method to print out the TRS-80 display is to POKE the printer driver address into the video control block in dedicated RAM (POKE 16414, 141: POKE 16415, 5). Thereafter every PRINT command will act as an LPRINT.

The trick is clever but very limiting. While it is in place, the CRT is inactive, the keyboard is tied to the line printer (you won't see what you are typing until you press ENTER) and output required on the screen will be sent only to the printer. The POKE routine and the one to cancel it (POKE 16414,88: POKE 16415, 4) are second-best techniques.

What's LPVIDEO

LPVIDEO (line print video), presented here in two versions, is a generalized subroutine to produce hard copy of everything on the TRS-80 screen, without interrupting the video display or compromising readability of keyboard input. Both versions produce identical machine language software which may be

Listing 1

```
              00220 ; PROGRAM TO PRINT THE CONTENTS OF THE VIDEO SCREEN TO THE LINEPRINTER
              00230 ;
              00240 ;
              00250 ; PROGRAM IS WRITTEN FOR TRS-80 48K, USING TRSDOS 2.2
              00260 ;
              00270 ; TO REWRITE FOR 32K SYSTEMS, ALL ADDRESSES BEGINNING "FF" SHOULD BE REWRITTEN
              00280 ;        TO BEGIN "9F."
              00290 ; TO REWRITE FOR 16K SYSTEMS, ALL ADDRESSES BEGINNING "FF" SHOULD BE REWRITTEN
              00300 ;        TO BEGIN "7F."
              00310 ;
              00320 ;
              00330 ;
              00340 ;
FF0C          00350        ORG   0FF0CH        ;TRANSFER POINT IS FF0CH, DECIMAL 65292,
              00360 ;                          FOR THE 48K VERSION.
3C00          00370 VIDEO  EQU   3C00H         ;STARTING ADDRESS FOR VIDEO MEMORY.
37E8          00380 LPADDR EQU   37E8H         ;LINE PRINTER ADDRESS.
              00390 ;
              00400 ;
              00410 ;
              00420 ;            INITIALIZATION SEQUENCE
              00430 ;
FF0C F5       00440 START  PUSH  AF            ;PROGRAM USES AF,BC AND DE REGISTER PAIRS.
FF0D C5       00450        PUSH  BC            ;THEREFORE, IT IS NECESSARY TO PRESERVE THE ENVIRONMENT
FF0E E5       00460        PUSH  HL
FF0F 21003C   00470        LD    HL,VIDEO      ;REGISTER PAIR HL WILL BE USED AS A POINTER TO
              00480 ;                          VIDEO SCREEN LOCATIONS.
FF12 0600     00490        LD    B,0           ;INITIALIZE REGISTER B TO COUNT NUMBER OF LINES THE PROGRAM
              00500 ;                          HAS COPIED FROM VIDEO TO THE LINE PRINTER.
              00510 ;
              00520 ;
              00530 ;
              00540 ;
              00550 ;            THIS SECTION OF CODE LOADS A LINE OF VIDEO INTO LINE PRINTER.
              00560 ;
FF14 0E00     00570 LDLINE LD    C,0           ;INITIALIZE REGISTER C TO COUNT NUMBER OF CHARACTERS
              00580 ;                          THE PROGRAM HAS COPIED FROM A LINE TO THE LP BUFFER.
              00590 ;                          LINE 570 IS THE START OF THE LINE LOADING ROUTINE.
FF16 CD36FF   00600 NXTBYT CALL  CHEKLP        ;CALL ROUTINE TO CHECK IF LP IS READY TO RECEIVE DATA.
FF19 7E       00610        LD    A,(HL)        ;ON RETURN FROM CHEKLP, LOAD BYTE FROM VIDEO TO A REGISTER.
FF1A 32E837   00620        LD    (LPADDR),A    ;THEN LOAD A REGISTER INTO THE LP BUFFER.
FF1D 23       00630        INC   HL            ;SET HL POINTER TO NEXT VIDEO BYTE.
FF1E 0C       00640        INC   C             ;BUMP THE COUNTER.
FF1F 3E40     00650        LD    A,40H         ;CHECK TO SEE IF 40H (DECIMAL 64) CHARACTERS HAVE
              00660 ;                          BEEN LOADED INTO THE LP BUFFER (I.E., FULL VIDEO LINE)
FF21 B9       00670        CP    C             ;BY COMPARING THE COUNT TO 40H.
FF22 20F2     00680        JR    NZ,NXTBYT     ;IF NOT, THEN GET THE NEXT BYTE, OTHERWISE FALL THROUGH.
              00690 ;
              00700 ;
              00710 ;
              00720 ;
              00730 ;            THIS SECTION OF CODE PRINTS A FULL LINE.
              00740 ;
FF24 CD36FF   00750        CALL  CHEKLP        ;CALL ROUTINE TO DETERMINE LINE PRINTER READINESS.
FF27 3E0D     00760        LD    A,0DH         ;ON RETURN FROM CHEKLP, LOAD A REGISTER WITH
              00770 ;                          ASCII CARRIAGE RETURN.
FF29 32E837   00780        LD    (LPADDR),A    ;SEND CARRIAGE RETURN TO LINE PRINTER.
FF2C 04       00790        INC   B             ;BUMP THE LINE COUNTER.
FF2D 3E10     00800        LD    A,10H         ;CHECK TO SEE IF 10H (DECIMAL 16) LINES HAVE
              00810 ;                          BEEN PRINTED (I.E., A FULL SCREEN).
FF2F B8       00820        CP    B
FF30 20E2     00830        JR    NZ,LDLINE     ;IF FULL SCREEN NOT YET PRINTED, GET NEXT LINE.
FF32 E1       00840 EXIT   POP   HL            ;OTHERWISE, RESTORE THE ENVIRONMENT.
FF33 C1       00850        POP   BC
FF34 F1       00860        POP   AF
FF35 C9       00870        RET                 ;RETURN TO CALLING PROGRAM
```

```
          00880 ;
          00890 ;
          00900 ;
          00910 ;
          00920 ;       THIS SECTION OF CODE CHECKS FOR PRINTER READINESS.
          00930 ;
FF36 3AE837 00940 CHEKLP LD    A,(LPADDR)     ;PLACE PRINTER STATUS IN THE A REGISTER.
FF39 CB7F   00950        BIT   7,A            ;IS PRINTER READY?
FF3B C8     00960        RET   Z              ;IF SO, RETURN
FF3C CB6F   00970        BIT   5,A            ;IS PRINTER IN LINE BUT DISABLED?
FF3E 2009   00980        JR    Z,DSABLD       ;IF SO JUMP TO MESSAGE ROUTINE.
FF40 CB77   00990        BIT   6,A            ;IS PRINTER OFF LINE?
FF42 28F2   01000        JR    Z,CHEKLP       ;IF NOT, CHECK PRINTER STATUS UNTIL READY.
            01010 ;
            01020 ;
            01030 ;
            01040 ;
            01050 ;       THIS SECTION OF CODE LOADS AND PRINTS ADVISORY OF PRINTER STATUS.
            01060 ;
FF44 2190FF 01070 OFFLIN LD    HL,MSSG2       ;IF PRINTER OFF LINE, HL POINTS TO START ADDRESS
            01080 ;                           OF ADVISORY MESSAGE
FF47 1803   01090        JR    DSPLAY         ;THEN JUMP TO DISPLAY SECTION.
FF49 2161FF 01100 DSABLD LD    HL,MSSG1       ;IF PRINTER DISABLED, HL POINTS TO START ADDRESS
            01110 ;                           OF ADVISORY MESSAGE.
FF4C D5     01120 DSPLAY PUSH  DE             ;SET UP TO CALL THE LEVEL2 CRT DISPLAY ROUTINE.
FF4D FDE5   01130        PUSH  IY
FF4F 7E     01140 LOOP   LD    A,(HL)         ;PUT BYTE FROM MESSAGE INTO A REGISTER.
FF50 CD3300 01150        CALL  33H            ;CALL LEVEL2 DISPLAY ROUTINE.
FF53 23     01160        INC   HL             ;BUMP MESSAGE POINTER.
FF54 3E00   01170        LD    A,0            ;SET UP TO CHECK FOR END OF MESSAGE.
FF56 BE     01180        CP    (HL)           ;IS IT?
FF57 20F6   01190        JR    NZ,LOOP        ;IF NOT, GET NEXT BYTE.
FF59 FDE1   01200        POP   IY             ;AT END OF MESSAGE, RESTORE IY AND DE
FF5B D1     01210        POP   DE
FF5C 2132FF 01220        LD    HL,EXIT        ;SETUP FOR RETURN TO CALLING PROGRAM.
FF5F E3     01230        EX    (SP),HL
FF60 C9     01240        RET                  ;BACK TO EXIT FOR FINAL RETURN.
            01250 ;
            01260 ;
            01270 ;
            01280 ;
            01290 ;       MESSAGE AREA
            01300 ;
FF61 50     01310 MSSG1  DEFM  'PRINTER NOT READY.  CHECK PAPER AND SWITCH.   '
FF62 52     FF63 49     FF64 4E     FF65 54     FF66 45     FF67 52     FF68 20     FF69 4E     FF6A 4F     FF6B 5
4     FF6C 20     FF6D 52     FF6E 45     FF6F 41     FF70 44     FF71 59     FF72 2E     FF73 20     FF74 20
 FF75 43     FF76 48     FF77 45     FF78 43     FF79 4B     FF7A 20     FF7B 50     FF7C 41     FF7D 50     FF7E
45     FF7F 52     FF80 20     FF81 41     FF82 4E     FF83 44     FF84 20     FF85 53     FF86 57     FF87 49
  FF88 54     FF89 43     FF8A 48     FF8B 2E     FF8C 20     FF8D 20     FF8E 20                 01320 ;
            01330 ;
            01340 ;
FF8F 00     01350        DEFB  0              ;TERMINATOR, INDICATES END OF MESSAGE.
            01360 ;
            01370 ;
            01380 ;
FF90 50     01390 MSSG2  DEFM  'PRINTER NOT IN LINE.  CHECK POWER OR CABLE.   '
FF91 52     FF92 49     FF93 4E     FF94 54     FF95 45     FF96 52     FF97 20     FF98 4E     FF99 4F     FF9A 5
4     FF9B 20     FF9C 49     FF9D 4E     FF9E 20     FF9F 4C     FFA0 49     FFA1 4E     FFA2 45     FFA3 2E
 FFA4 20     FFA5 20     FFA6 43     FFA7 48     FFA8 45     FFA9 43     FFAA 4B     FFAB 20     FFAC 50     FFAD
4F     FFAE 57     FFAF 45     FFB0 52     FFB1 20     FFB2 4F     FFB3 52     FFB4 20     FFB5 43     FFB6 41
  FFB7 42     FFB8 4C     FFB9 45     FFBA 2E     FFBB 20     FFBC 20     FFBD 20                 01400 ;
            01410 ;
            01420 ;
FFBE 00     01430        DEFB  0              ;TERMINATOR
            01440 ;
            01450 ;
            01460 ;
0000        01470        END
00000 TOTAL ERRORS
LOOP    FF4F
MSSG1   FF61
DSPLAY  FF4C
MSSG2   FF90
OFFLIN  FF44
DSABLD  FF49
EXIT    FF32
CHEKLP  FF36
NXTBYT  FF16
LDLINE  FF14
START   FF0C
LPADDR  37E8
VIDEO   3C00
```

HEX ADDRESS	MACHINE CODE	LINE NUMBER	LABEL	OPERATION	OPERAND(S)	COMMENTS
FF16	CD36FF	00600	NXTBYT	CALL	CHEKLP	;CALL ROUTINE . . . .
FF19	7E	00610		LD	A,(HL)	;ON RETURN . . . .
.						
.						
FF32	E1	00640	EXIT	POP	HL	;OTHERWISE . . . .

Example 1. Assembly Language Format.

left in protected memory as a user service routine (USR) available to any BASIC program. Printing the entire CRT screen becomes as easy as selectively calling the USR from a BASIC program.

Listing 1 is an assembly language program in Zilog Z-80 mnemonics. Don't be threatened if assembly language is new to you; this article steps through the listing slowly. In addition to learning about assembly language programming, you'll acquire a very useful subroutine. Listing 2 produces the same result as Listing 1, but uses BASIC to POKE instructions into memory. It's an easy way to get LPVIDEO running so you can appreciate how well machine language works.

Program Theory and Operation

The assembly language program (Listing 1) was produced using the TRS-80 Editor/Assembler. Even if you don't have an Editor/Assembler, you can still load the program through BASIC, or with T-BUG (a Radio Shack programming and debugging package) or, if you are using a disk-oriented system, with the TRSDOS DEBUG commands.

Listing 1 is in seven-column format (see Example 1). Column three, the only unbroken column, is a series of line numbers. Column four contains optional labels, as in line 600, labeled NXTBYT. Rather than specify an address as hexadecimal FF16, for example, I can refer to that address by the arbitrary label, NXTBYT.

This is particularly useful, as the address FF16H was determined after I wrote the program, when the assembler worked on the source code (in columns five and six) to develop column one (the RAM hexadecimal addresses) and column two (the machine language object code). Column two contains the actual program. Everything else is supporting material.

LPVIDEO begins with the definition of its first memory location (FF0CH, decimal 65292, at line 350) and the assignment of values to the labels VIDEO and LPADDR. VIDEO, set at 3C00H, is the address of the first byte of the CRT memory.

The TRS-80 is memory-mapped. Frequently input/output data are channeled through RAM addresses rather than through the 256 ports of the Z-80 microprocessor. The display occupies 1024 bytes from 3C00H to 3FFFH (decimal 15360 to 16383). LPADDR stands for 37E8H (decimal 14312), the location in RAM at which software addresses the Centronics parallel port, used by most Radio Shack printers.

The remainder of LPVIDEO may be regarded as six modules (see Table 1). The initialization sequence preserves the data in register pairs AF, BC and HL. Register pair HL, which will be used as a pointer to successive screen locations, is loaded with the starting address of the video memory.

Listing 2

```
1 REM *** BASIC PROGRAM TO POKE LPVIDEO INTO HIGH MEMORY
2 REM *** 48K DISK VERSION
3 REM *** SEE ACCOMPANYING ARTICLE FOR MODIFICATION DETAILS FOR
        SMALLER SYSTEMS AND NON-DISK SYSTEMS
4 '
5 '
6 POKE 16553,255 'CORRECTION FOR ERROR IN SOME LEVEL2 CHIPS
7 '
8 '
9 '
10 REM *** INITIALIZATION SEQUENCE
20 DATA 245, 197, 229, 33, 0, 60, 6, 0
30 '
40 '
50 '
60 REM *** LOAD LINE OF VIDEO INTO LP BUFFER
70 DATA 14, 0, 205
80 DATA 54, 255  'MODIFY THIS LINE, IF NECESSARY, ACCORDING TO
    DIRECTIONS
90 DATA 126, 50, 232, 55, 35, 12, 62, 64, 185, 32, 242
100 '
110 '
120 '
130 REM *** PRINT LINE
140 DATA 205
150 DATA 54, 255  'MODIFY THIS LINE, IF NECESSARY, ACCORDING TO
    DIRECTIONS
160 DATA 62, 13, 50, 232, 55, 4, 62, 16, 184, 32, 226, 225,
    193, 241, 201
170 '
180 '
190 '
200 REM *** CHECK FOR PRINTER READINESS
210 DATA 58, 232, 55, 203, 127, 200, 203, 111, 40, 9, 203,
        119, 40, 242
220 '
230 '
240 '
250 REM *** LOAD AND PRINT ADVISORY OF LP STATUS
260 DATA 33
270 DATA 144, 255  'MODIFY THIS LINE, IF NECESSARY, ACCORIDNG TO
    DIRECTIONS
280 DATA 24, 3, 33
290 DATA 97, 255  'MODIFY THIS LINE, IF NECESSARY, ACCORDING TO
    DIRECTIONS
300 DATA 213, 253, 229, 126, 205, 51, 0, 35, 62, 0, 190, 32,
        246, 253, 225, 209, 33
310 DATA 50, 255  'MODIFY THIS LINE, IF NECESSARY, ACCORDING TO
    DIRECTIONS
320 DATA 227, 201
330 '
340 '
350 '
360 REM *** MESSAGES
370 DATA 80, 82, 73, 78, 84, 69, 82, 32, 78, 79, 84, 32, 82,
        69, 65, 68, 89, 46, 32, 32, 67, 72, 69, 67, 75, 32
380 DATA 80, 65, 80, 69, 82, 32, 65, 78, 68, 32, 83, 87, 73,
        84, 67, 72, 46, 32, 32, 32, 0
390 DATA 80, 82, 73, 78, 84, 69, 82, 32, 78, 79, 84, 32, 73,
        78, 32, 76, 73, 78, 69, 46, 32, 32, 67, 72, 69, 67, 75
400 DATA 32, 80, 79, 87, 69, 82, 32, 79, 82, 32, 67, 65, 66,
        76, 69, 46, 32, 32, 32, 0
410 '
420 '
430 '
440 FOR I%= -244 TO -66  'MODIFY THIS LINE, IF NECEESARY,
    ACCORDING TO ACCOMPANYING DIRECTIONS
450    READ B%
460    POKE I%, B%
470 NEXT
```

CHEKLP, the subroutine beginning at line 940, examines the printer's status. If the printer is ready the subroutine executes a return to the calling location. LPADDR contains 5DH (decimal 93) when the printer is ready. The only other possible data at LPADDR are FFH (decimal 255) if the printer is not on line (e.g., power off), DFH (decimal 223) if it is on line but disabled (e.g., out of paper), or BFH (decimal 191) if it is unable to receive data because it is printing.

Is the Printer Ready

Bit test operations determine which printer condition is true. Bit tests were chosen rather than compares because they save program space by minimizing loads and register usage and by diminishing the number of pushes and pops to and from the stack. The bits of each byte are numbered 0 through 7, from right to left, corresponding to the power of two represented by that bit. Accordingly, printer condition reports may be expressed as in Example 2.

The instruction at line 940 loads LPADDR data into the A register. Bit 7 is 0 (reset) if and only if the printer is ready. At line 950 we test for readiness. Line 960 is a return on zero; if bit 7 is reset, control returns to the calling address. If bit 7 is 1 (set) the program continues with the next instruction, testing bit 5 of the datum at LPADDR. Bit 5 is 0 if and only if the printer is on line but disabled.

If the printer is disabled the program executes a jump to DSABLD (line 1100), which displays a status advisory. If the printer is enabled but otherwise unready, the subroutine tests bit 6, which is set if the printer is disabled (already eliminated) or off line.

If bit 6 is set, the program jumps to OFFLIN (line 1070) to print another message. If no jump is executed, the program assumes that the printer was unready because it was printing. The program returns to CHEKLP and repeats the loop until it is exited by printer availability, disability or off line status.

When the printer is ready, the loading module (LDLINE, begin-

READY	decimal 63	3FH	binary 00111111
PRINTING	decimal 191	BFH	binary 10111111
DISABLED	decimal 223	DFH	binary 11011111
OFF LINE	decimal 255	FFH	binary 11111111

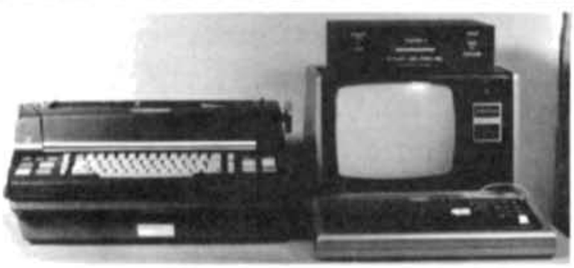
Example 2. Printer Condition Report.

```
FF00 => 43 E5 21 ED FE 22 16 40  E1 CD 2D 40 F5 C5 E5 21
FF10 => 00 3C 06 00 0E 00 CD 36  FF 7E 32 E8 37 23 0C 3E
FF20 => 40 B9 20 F2 CD 36 FF 3E  0D 32 E8 37 04 3E 10 B8
FF30 => 20 E2 E1 C1 F1 C9 3A E8  37 CB 7F C8 CB 6F 28 09
FF40 => CB 77 28 F2 21 90 FF 18  03 21 61 FF D5 FD E5 7E
FF50 => CD 33 00 23 3E 00 BE 20  F6 FD E1 D1 21 32 FF E3
FF60 => C9 50 52 49 4E 54 45 52  20 4E 4F 54 20 52 45 41
FF70 => 44 59 2E 20 20 43 48 45  43 4B 20 50 41 50 45 52
FF80 => 20 41 4E 44 20 53 57 49  54 43 48 2E 20 20 20 00
FF90 => 50 52 49 4E 54 45 52 20  4E 4F 54 20 49 4E 20 4C
FFA0 => 49 4E 45 2E 20 20 43 48  45 43 4B 20 50 4F 57 45
FFB0 => 52 20 4F 52 20 43 41 42  4C 45 2E 20 20 20 00 3D
FFC0 => FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF
FFD0 => FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF
FFE0 => FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF
FFF0 => FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF
```

Example 3. Memory Dump of LPVIDEO.

STARTING ADDRESS	STARTING LINE	MODULE FUNCTION
FF0C	00440	INITIALIZATION SEQUENCE
FF14	00570	LOAD VIDEO LINE TO PRINTER BUFFER
FF24	00750	PRINT CONTENTS OF PRINTER BUFFER
FF36	00940	SUBROUTINE CHECK FOR PRINTER READINESS
FF44	01070	PRINT ADVISORY ON PRINTER STATUS
FF61	01310	MESSAGE AREA CONTAINS ASCII STRINGS

Table 1. Program Modules.



VERSION	ORG & HEX TRANS ADDRESS	DEC TRANS ADDRESS	MEMORY SIZE?
48K DISK	0FF0CH	65292	65291
48K NON DISK	0FF4CH	65356	65355
32K DISK	0BF0CH	48908	48907
32K NON DISK	0BF4CH	48972	48971
16K DISK	7F0CH	32524	32523
16K NON DISK	7F4CH	32588	32587

Table 2. Entry Points and Memory Allocation.

LSB* OF ADDRESS	48K DISK	48K NON	32K DISK	32K NON	16K DISK	16K NON
17	36	77	36	77	36	77
18	FF	FF	BF	BF	7F	7F
25	36	77	36	77	36	77
26	FF	FF	BF	BF	7F	7F
45	90	D1	90	D1	90	D1
46	FF	FF	BF	BF	7F	7F
4A	61	A2	61	A2	61	A2
4B	FF	FF	BF	BF	7F	7F
5D	32	73	32	73	32	73
5E	FF	FF	BF	BF	7F	7F

*Denotes least significant byte

Table 3. Machine Language Modifications.

Listing 3

```
460 CLS
470 PRINT @ 128, "THE FOLLOWING" N% "SLIDE NUMBERS INCLUDE
    SCENES OF " S$ ". "
480 FOR I%=1 TO N%
490    PRINT SN%(N%);
500 NEXT
510 PRINT
520 PRINT "<R>EQUEST PHYSICAL DESCRIPTION. "
530 PRINT "<S>ELECT NEW CATEGORY. "
540 PRINT "<D>ELETE SLIDES FROM LIST. "
550 PRINT "<A>DD SLIDES TO LIST. "
560 A$=INKEY$: IF A$="" THEN 560
570 IF A$="P" THEN X=USR0(X) :  REM *** SHIFT P
580 IF A$="R" THEN 1570
590 IF A$="S" THEN 350
600 IF A$="D" THEN 7000
610 IF A$="A" THEN 9000
620 GOTO 560
```

ning at 570) copies one screen line from the CRT memory to the printer buffer.

Line 610 loads a byte from the video display, indicated by register HL, into the A register which is written to LPADDR and sent to the printer buffer by line 620. HL is incremented to point to the next video byte. Register C is incremented and checked for 40H (decimal 64, the number of characters on each line). Until C contains 40H the program loops back to NXTBYT (line 600). When C contains 40H, control drops through to line 750.

At line 750 the program enters the print module, which calls CHEKLP. If the printer is ready, the A register is loaded with DH (decimal 13), an ASCII carriage return, which is sent to the printer at line 780 and forces a print of the buffer contents.

Register B, which counts lines, is incremented and checked for 10H (decimal 16). If B contains less than 10H the program has not examined every video line. Control then jumps to LDLINE, zeroing the C register, beginning a load of the next video line. If B contains 10H control passes to EXIT, which restores the environment and executes a return to the calling program.

The section of code that outputs printer advisories begins at line 1070. OFFLIN loads HL with the starting address of MSSG2 (line 1390). DSABLD does the same for MSSG1 (line 1310). The program continues with DSPLAY (line 1120) which calls the Level II display routine at ROM address 33H. Each of the messages is established in memory with the DEFM pseudo-operation. DEFM is not a Z-80 operation but an assembler instruction to reserve a block of memory for ASCII values of a message string. After each message is a 0, entered with the DEFB pseudo-op that defines a byte according to the programmer's wishes. The loop in DSPLAY repeatedly calls the Level II display routine until it encounters the 0 terminator, an invalid ASCII code.

A TRSDOS DEBUG dump of the memory area containing the complete machine language program is reproduced as Example 3. The enclosed section beginning at FF0CH and ending at FFBEH is LPVIDEO. The dump should aid the reader in producing LPVIDEO using T-BUG or the DEBUG commands in TRSDOS.

Calling the Subroutine

LPVIDEO is useful when running a BASIC program. The memory size inquiry of the BASIC initialization dialogue should be answered with a number smaller than the transfer address (entry point) of LPVIDEO. Responses are suggested in Table 2. The standard transfer to LPVIDEO is by a USR call.

Programs in disk BASIC (or G2's Level III by Microsoft) should include instructions such as DEFUSR0 = &HFF0C that defines user service routine 0 as having a transfer address of FF0CH (decimal 65292). In Level II BASIC it is necessary to POKE the hexadecimal address into memory one byte at a time the least significant byte first—12 (0CH) into 408EH (decimal 16526) and 255 (FFH) into 408FH (decimal 16527).

Disk BASIC and Level III call the subroutine with an instruction like X = USR0(X), while Level II users should try X = USR(X). Processing is branched to LPVIDEO, the screen is printed (or the user is advised why it is not) and BASIC regains control of the program.

Most of the BASIC software I write accepts keyboard input via the INKEY$ statement, a technique particularly suitable for entering LPVIDEO. A typical calling sequence, illustrated in Listing 3, is part of a hypothetical BASIC program.

Line 560 begins the input routine. If the user types a shift P the BASIC program calls LPVIDEO. Since the subroutine returns control to line 580, the INKEY$ routine is continued and awaits further user input.

Modifications for Different TRS-80 Configurations

LPVIDEO is written for a Model I disk-oriented TRS-80 system with 48K of user RAM. It resides near the top of the 48K memory. The very top is not employed as TRSDOS 2.2 uses the highest 64 bytes of memory when BASIC or BASICR is loaded.

The program is easily modified for other configurations and is adaptable to other computer systems. Use Table 2 to determine origin addresses for pro-

VERSION	DATA ON LINE #					LINE 440
	80	150	270	290	310	
48K DISK	54, 255	54, 255	144, 255	97, 255	50, 255	FOR I% = −244 TO −66
48K NON DISK	119, 255	119, 255	209, 255	162, 255	15, 255	FOR I% = −179 TO −1
32K DISK	54, 191	54, 191	144, 191	97, 191	50, 191	FOR I% = −16628 TO −16450
32K NON DISK	119, 191	119, 191	209, 191	162, 191	15, 191	FOR I% = −16563 TO −16385
16K DISK	54, 127	54, 127	144, 127	97, 127	50, 127	FOR I% = 32524 TO 32702
16K NON DISK	119, 127	119, 127	209, 127	162, 127	15, 127	FOR I% = 32589 TO 32767

Table 4. Modifications to Listing 2.

ducing LPVIDEO with an assembler program. Table 2 also provides answers to the memory size inquiry and addresses to be specified for a USR transfer.

Table 3 lists the least significant bytes of RAM addresses and the data those addresses should contain for various configurations. The most significant byte for all 32K addresses is BF, for 16K, 7F.

Table 4 suggests modifications of Listing 2, the BASIC program to POKE LPVIDEO into memory. Tables 2, 3 and 4 consider a TRSDOS 2.1 system to be a non-disk system.

Conclusion

LPVIDEO is a useful routine. Like every program, it can be changed or enlarged. A particularly enticing challenge is to write a patch to the TRS-80 keyboard scan so that it will look continuously for a shift P and execute LPVIDEO, regardless of what other task is being undertaken, much the same way that shift @ stops background tasks.

The user may also want to modify the program so that it prints only up to the current cursor position. Working with LPVIDEO can give the novice assembly language programmer valuable experience using important instructions and techniques.■

References

William Barden, Jr. *TRS-80 Assembly Language Programming.* Fort Worth: Radio Shack. 1979.

____. *How to Program Microcomputers.* Indianapolis: Howard W. Sams & Co. 1977.

____. *The Z-80 Microcomputer Handbook.* Indianapolis: Howard W. Sams & Co. 1978.

Adam Osborne et. al. *Z80 Programming for Logic Design.* Berkeley: Osborne & Associates, Inc. 1978.

Kathe Spracklen. *Z-80 and 8080 Assembly Language Programming.* Rochelle Park, N.J.: Hayden Book Company, Inc. 1979.

Why enter figures in decimal, when you can use these routines!

Fractional Input

David R. Cecil
Dept of Mathematics
Texas A & M University
Kingsville TX 78363

Would you like to be able to enter fractions such as −8/5.6 with the INPUT statement in BASIC? How about variable length numeric data with one INPUT?

If you've tried simultaneous equations, matrix-inversion problems or even sorting, then you know how handy these two features would be.

The programs given here allow fractions and/or variable length data streams. They are written for the TRS-80 Level II, but should work with any extended BASIC having the string functions LEN, MID$, RIGHT$ and VAL.

If you are only going to input one number at a time and wish to be able to enter a fraction for your input, then see Program 1. (More extensive REMARKS about Program 1 appear in Program 3 starting at line 1140.)

To call Program 1 your main program will need the statement INPUT C$: GOSUB 2010. For example, Program 2 displays x and 5x for user-chosen x's.

To use Program 2 type RUN. When ? appears type in your numeric value, such as −17/681 or 12.3, and press the ENTER key. Use the BREAK key to terminate this program.

If you want to enter more than one number per INPUT and also be able to enter fractions, then take a look at Program 3.

To use Program 3 type RUN 1010. When the prompt ? of the INPUT statement appears, type the first entry followed by exactly one space, then the second entry, then one space, etc. (for example, −1ƀ3/4ƀ−5. 086ƀ7/−9.1). Do not use commas to separate the data values or you will get the message ?EXTRA IGNORED. Don't put a space after the last number entered or between a negative sign and its value and do not put more than one space between numbers, or you will have some unwanted zeros in your output. You are permitted entries such as −3/4, 3/−4 or even −3/−4 if you prefer that to 3/4.

The dimension of the output matrix D given in line 1050 is the maximum. You may wish to use a smaller DIM size.

Output matrix D is available for future use, other than for just displaying its contents as is done in lines 1090-1110. If line 1085 is changed to 1085 RETURN, then this program, like Program 1, would be nice to use with the INPUT statement. To utilize Program 3 as a subrou-

```
2000 REM*THE INPUT IS STRING C *
2010 FOR J = 1 TO LEN(C$)
2020 IF MID$(C$,J,1) = "/" THEN 2050
2030 NEXT J
2040 V = VAL(C$): GOTO 2060
2045 REM*THE OUTPUT IS THE NUMBER V *
2050 V = VAL(C$)/VAL(RIGHT$(C$,LEN(C$) - J))
2060 RETURN
```

Program 1.

```
10 INPUT C$: GOSUB 2010
20  PRINT V, 5*V
30 GOTO 10
```

Program 2.

```
1001 REM *A PROGRAM TO INPUT FRACTIONS*
1002 REM *OR VARIABLE LENGTH NUMERIC DATA*
1003 REM *WITH THE INPUT STATEMENT*
1004 REM *BY DAVID R. CECIL*
1010 CLEAR 1200: INPUT B$
1020 LET B = 1: LET K = 1
1025 REM *LINE 1030 ADDS A SPACE AT THE END*
1026 REM *OF THE INPUT STREAM*
1030 A$ = B$ + "ƀ"
1035 REM *THE OUTPUT IS THE MATRIX D. IT IS*
1036 REM *DIMENSIONED TO BE THE LENGTH OF STRING A*
1040 Z = LEN(A$)
1050 DIM D(Z)
1055 REM *CHECK TO SEE IF STRING A HAS A SPACE*
1056 REM *(A BLANK) AT POSITION I*
1060 FOR I = 1 TO Z
1070 IF MID$(A$,I,1) = "ƀ" THEN GOSUB 1130
1080 NEXT I
1085 REM *OUTPUT MATRIX D HAS SIZE K-1 WHERE*
1086 REM *K-1 IS THE NUMBER OF ITEMS ENTERED.*
1087 REM *DISPLAY THE OUTPUT, ONE ITEM PER LINE*
1090 FOR S = 1 TO K-1
1100 PRINT D(S)
1110 NEXT S
1120 END
1125 REM *CHECK TO SEE IF THE ENTRY BETWEEN TWO*
1126 REM *BLANKS HAS A DIVISION SYMBOL APPEARING*
1130 C$ = MID$(A$,B,I-B)
1140 FOR J = 1 TO LEN(C$)
1150 IF MID$(C$,J,1) = "/" THEN 1180
1160 NEXT J
```

tine, use INPUT B$: GOSUB 1020 in place of an INPUT statement.

As an illustration, Program 4 displays the numbers entered as decimals, followed by the original entry (whether fractional or not) and does a sort of numbers at the same time so that the results appear in ascending order.

To use Program 4 be sure Program 3 is still loaded. Then a) change line 1050 to read DIM D (Z), E$(Z); b) change line 1085 to read RETURN; and c) change line 1210 to read E$(K) = C$: B = I + 1: K = K + 1. After these changes type RUN and input fractions or whatever numeric data desired when ? appears.

The division sign "/" in lines 1150 and 1180 could be replaced by "+" if you will be entering numbers in a form such as 2.7081 + 6.952.

You might notice that Program 3 can be shortened from line 1150 to the end by interchanging some lines and eliminating a GOTO. The formulation presented here was used so that omitting lines 1140 through 1190 (if your data has no fractions occurring but you want to input a stream of numbers with one INPUT) will have no effect on the rest of the program.■

```
1170 GOTO 1200
1175 REM "ASSIGN THE RESULT OF THE DIVISION TO"
1176 REM "THE OUTPUT D(K)"
1180 D(K) = VAL(C$)/VAL(RIGHT$(C$,LEN(C$)-J))
1190 GOTO 1210
1195 REM "IF / DOES NOT APPEAR THEN ASSIGN D(K)"
1196 REM "TO THE NUMBER APPEARING BETWEEN TWO BLANKS"
1200 D(K) = VAL(C$)
1210 B = I + 1: K = K + 1
1220 RETURN
```

Program 3.

```
10 CLEAR 1200: INPUTB$: GOSUB 1020
20 FOR S = 1 TO K-1: FOR J = 1 TO K-2
30 IF D(J) > D(J + 1) THEN 80
40 NEXT J,S
50 FOR S = 1 TO K-1
60 PRINT D(S), "ORIGINAL ENTRY = b"; E$(S): NEXT S
70 END
80 A = P(J + 1): D(J + 1) = D(J): D(J) = A
90 B$ = E$(J + 1): E$(J + 1) = E$(J): E$(J) = B$
100 GOTO 40
```

Program 4.

If you want to PEEK inside your 80, try this neat utility.

MEMORY DUMP

Allan S. Joffe W3KBM
1005 Twining Road
Dresher PA 19025

Every program starts with an idea or a premise. But only with some tenacity, does it develop into a finished product. The premise here is that by using the following program fragment you can print the contents of memory.

```
5 CLS
10 FOR X=0 TO 100
20 PRINT PEEK(X);
30 NEXT X
```

Running this program, you will now see the contents of memory locations 0 to 100 or the first 101 memory locations of your BASIC ROM. Remember, 0 is a very real number. (This information is in the memory map of your Level II book.)

When printed the notation of the program is decimal. However, to be truly useful, the printout should be in hexadecimal form

If you modified line 10 to read:

```
10 FOR X=0 TO 3000
```

the BASIC ROM contents will print out and you will find that the numbers (in decimal) run from 0 to 255 with no number in excess of 255. In hex notation, 255 (decimal) = FF.

To represent the hexadecimal value of any memory cell we can make up a decimal to hexadecimal "converter."

Hex Conversion

Hex values for numerals 0 to 15 can be displayed by a single character—0 through 9 and A through F—with the alphabetical characters A through F used to represent the values 10 through 15 in decimal. For the sake of display uniformity we will use two characters to represent each value. To display this initial set of hex characters we will append a "leading 0" to each one: 00 07 09 0C 0F etc. Any hex equivalent of a decimal value in excess of 15 is automatically two characters wide (in the range of 0 to 255 decimal.)

The base or radix of hex is 16. If we take any value in our considered range of 0 to 255 decimal and divide it by 16, we will get a whole number plus a remainder of so many sixteenths. Even if a number is divided by sixteen evenly, the result is still a whole number plus a remainder of zero sixteenths.

Now let us look at the MSB and LSB. In our two digit hex representation the leftmost character is the MSB (most significant bit). It will have the greatest value in determining the complete value of the number. The righthand character is the LSB (least significant bit) of the pair and has the least value to add to the numerical value.

Here is an example. Let us break the decimal 93 down into its component parts. The 9 is worth nine tens or 90 and is the MSB, for obviously its numerical worth is greater than that of the 3 which is three ones and is the LSB. The sum of the MSB + the

```
5 CLS
10 FOR X=0 TO 255
20 G=X/16
30 B=G-INT(G)
40 IF G-B>9 PRINT CHR$((G-B)+55); ELSE PRINT G-B;
50 IF 16*B>9 PRINT CHR$((16*B)+55); ELSE PRINT CHR$((16*B)+48);
60 NEXT X
```

Initial Program Listing

```
5 CLS
6 INPUT "START ADDRESS IN DECIMAL";S
8 INPUT "END ADDRESS IN DECIMAL";E
10 FOR X=S TO E
12 IF J=16 THEN J=0
13 IF J=0 AND X>S THEN PRINT X-1;:PRINT CHR$(11);
15 A=PEEK(X)
20 G=A/16
30 B=G-INT(G)
40 IF G-B>9 PRINT CHR$((G-B)+55);ELSE PRINT CHR$((G-B)+48);
50 IF 16*B>9 PRINT CHR$((16*B)+55);ELSE PRINT CHR$((16*B)+48);
55 PRINT CHR$(32);
60 J=J+1;NEXT X
```

Finished Program: Decimal to hex memory dump

LSB represents the full numerical value of this number.

If we take the same numerical pair, 93, in hex, we can do the same thing, remembering that the MSB is worth the value of the digit multiplied by 16.

The LSB is determined as in decimal with the exception that in decimal the greatest value of the LSB is nine, while in hex its greatest value is 15, or to be more precise, F.

Thus, 93 in hex is the sum of nine sixteens + three ones, which translates to 144 + 3 or 147 in decimal notation.

Determining Hex

Now we are ready for the next step of the game plan. Divide the decimal 93 by 16. The answer is five + thirteen sixteenths. In this way we have determined that the MSB of this number is 5. The LSB is now multiplied by 16 to turn it from a fraction into an integer, making it 13. Since we are now speaking in hex, 13 is annotated as D. Thus 93 in decimal is 5D in hex.

Here now is what our program has to do:

1. It has to take any decimal number from 0 to 255 and divide it by 16.
2. The program has to recognize the whole number and print it properly. That is it must print the number as letters A through F if the number is in excess of nine.
3. The program must recognize the fractional remainder and multiply it by 16. Then it has to print it properly to the right of the first digit (again if this number is larger than nine, it is represented by characters A through F.
4. Finally, the program must recognize the presence of a "single digit number" (numbers 0 to 15 decimal) and it must append a 0 to the left of the digit (03,0D, etc.)

First we must come up with a routine that will divide a number in the desired range (0 to 255) by sixteen and give a separate identity to the integer and the fractional portions of the answer.

Take any mixed number our possible division could produce, for example 5-7/16. Separate and give an identity to the fractional remainder by using the INT function. If X = 5-7/16 then X − INT(X) would be the fractional remainder.

```
 5  CLS
10  INPUT X
20  G = X/16
30  B = G - INT(G):PRINT B
40  GOTO 10
```

The above program fragment has separated this fractional remainder and given it an identity B.

It follows that if G represents the whole number plus the fractional remainder and B equals this fractional remainder, that G − B equals the whole number. Thus we have given the whole number a programmable identity just as we did for the fraction B.

We now have enough information from these logically developed facts along with the use of some string functions to construct an initial program.

In line 10 we are inputting our range of interest. The next two lines are the same with the exception of line 30 where we had printed B. This is no longer of use as it was for demonstration only.

Line 40 says in English: If the whole number portion of X is greater than nine print the ASCII character which is represented by the whole number + 55. If G − B is NOT greater than nine just print G − B. This line takes care of the MSB.

Line 50 serves the same function for the LSB.

The value of G − B will range from 0 to 15, as X ranges from 0 to 255. The ASCII code values for numerals 0 to 9 ranges from 48 to 57. The ASCII code values for the letters A through F run from 65 to 70. In addition, the value of 16•B ranges from 0 to 15 as the fractional remainder ranges from 0/16 to 15/16. If you carefully examine the conditions set forth in lines 40 and 50 you can see how the computer sorts out the given facts and prints the proper two characters in hex format for each decimal value of X that we input.

```
15  A = PEEK(X)   REM this is a new line
20  G = A/16      REM this is old line 20 modified
```

Example 1

```
12  IF J = 16 THEN J = 0          REM this is a new line
13  IF J = 0 THEN PRINT CHR$(11);  REM this is a new line
60  J = J + 1 :NEXT X             REM this is old line 60 modified
```

Example 2: The Counter

```
 6  INPUT "START ADDRESS IN DECIMAL";S   REM new line
 8  INPUT "END ADDRESS IN DECIMAL";E     REM new line
10  FOR X = S TO E                       REM changed line 10
```

Example 3

The Bug

At this stage of the game we are very lucky not to have run into a gotchagotcha or bug as it is commonly called. Don't cheer just yet, for if you run the program it will fill your screen with the proper values, BUT the printout looks funny.

The numbers from 0 to 159 (decimal) will run together so that one pair is difficult to distinguish from another. The condition worsens for the numbers 160 to 255.

We need to add one more line to the program.

```
55  PRINT CHR$(32);
```

This line tells the computer that each time it prints an LSB, it is to print a CHR$(32), or a blank space, before it processes the calculations for the next number.

Now that our program is in some sort of shape, though with a few formatting bugs (display format problems), let's add the necessary instructions to allow the program to dive into the memory locations and print the values in hex. To the existing program add the lines in Example 1.

Line 15 uses the PEEK function to examine the contents of the memory cell locations in line 10. In our present instance this would be the first 256 locations of the BASIC ROM. If you had wanted to examine the contents of the video memory locations line 10 would read:

```
10  FOR X = 15360 TO 16383
```

The display format problem and its cause are clear enough. Because of the 64 characters per line limit the final two character hex number of a line is split between the end of that line and the beginning of the next.

The solution is to print no more than 16 sets of hex characters per line.

We can insert a counter into the program to add a carriage return after the sixteenth pair of characters on a line (Example 2).

However, now when we run the program we find another problem—split lines.

If you examine the print statements in the program you will see that all but the second half of line 40 use some form of CHR$ function. Let's change this to agree in format with the rest of the print statements:

```
:ELSE PRINT CHR$((G - B) + 48);
```

Now when you run the program there should be no problems with the display format. A good rule, as demonstrated by the built in "gotcha" is to be consistent with your formatting of print statements. Like everything in life this may not be universal, but it is a good way to stay out of trouble—most of the time.

Printing Location

Finally, we will modify line 13 so that the decimal location will print out after the last hex digit on each line. Line 13 will now read:

```
13 IF J=0 AND X□S THEN PRINT X-1;
   :PRINT CHR$(11);
```

It may help a bit to examine why the line is written this way. If you run the program and specify a starting address (S) of 0 and an end address (E) of 16, the program will print cell 0 to cell 16 for a total of 17 cells, as it prints out from S to E inclusive. Cell 0, remember, is in reality cell one physically.

When the program prints out the first sixteen cell contents and then prints 15 at the end of the line, it means that while this cell is the sixteenth cell printed, that it is (in decimal) cell number 15 because of the fact that zero is the first location. This remains true even if the S value is NOT the figure 0.

An S (start) value of 53 and an E (end) value of 69 would give an X value of 68 at the end of the line. Looking at it another way, you are printing the value of cell 53 and then printing 15 more values to make up the total of 16 values in the line (53 + 15 = 68). This is the end of line value.

Changing the Area

If we modify the input statement we won't have to modify line 10 every time we want to change the area of memory to be examined. Add the new lines in Example 3.

One last bell and whistle. After each full screen has been printed, you might like the program to halt, while you examine it. Then when you hit the ENTER key, the program should continue to the next full screen. I suggest a counter, similar to the one that limits the line length to sixteen hex characters, and some way of making the ENTER key a trigger for program resumption.

There is a lot of meat here to digest. Follow the program fragment one step at a time and your own homebrew programs will develop more easily.

The computer will tell you if you make a mistake—sometimes—but it will only do what you direct it to do, which is the main reason why debugging is such a popular indoor sport. ■

PROGRAM STORE

(202) 337-4691

4200 Wisconsin Ave NW PO Box 9609 Washington D.C. 20016

Save time & space!

PACKER

by Cottage Software

This is the ultimate editing tool for BASIC program lines. There are five commands which allow easier reading of BASIC programs and more efficient execution by the computer.

The 'unpack' command breaks multiple statement lines into single statement lines with extra spaces for easier reading and editing. The 'short' command deletes any unnecessary words like LET and all REMarks.

The 'pack' commmand compresses lines into multiple statement lines up to the maximum length you specify while maintaining complete program logic. This can easily reduce the memory requirement by more than 33%! As you can imagine this also speeds up execution of a program, saves time in loading a program from either tape or disk and saves disk space.

And the 'move' command allows you to move any section of your program to a new location. With the 'renumb' command you can renumber your BASIC lines.

So if your programs need more memory, or you need more time, order your 'packer'! 16k, 32k and 48k versions supplied on two cassettes for $29.95.

KEYEDIT

by Phil Pilgrim from Discovery Bay

This machine language utility is for all users of either Level II or Disk BASIC. The functions give you auto-repeat on every key, screen-oriented editing, and debounce. In the editing mode, you just move the cursor to any position on the screen to instantly insert or delete. Plus, whole BASIC statements can be copied to other parts of the program.

You also get single keystroke entry. For example, just press <SHIFT> R to enter the word READ. And you can redefine the keys to any BASIC keyword.

In addition, the keyboard macro facility allows you to define any frequently typed pattern and then execute it with a single keystroke. This is especially great for DATA entry and other repetitive tasks.

Order the greatest keyboard utility yet! $18.95

DEBUG

by Bob Pierce from Quality

Single step your programs through RAM and ROM, with this relocatable machine language utility. Includes multiple breakpoints, option to execute CALLs in a single step and several display options. $14.95

DISASSEMBLER 1.2

by Roy Soltoff from Misosys & Acorn

This two pass Z-80 disassembler produces symbolic labels with output to either the video monitor, printer or tape. Radio Shack's Editor Assembler can load the tapes. If you own the Editor Assembler, complete the package with this program. Program on tape for two different memory locations. TRS-80 Level II ~~$19.95~~ NOW only $14.95

Morloc's Tower

from Automated Simulations

As Brian Hammerhand, you race to save the comely wench Imelda from the fireballs of Morloc the Mad. An exciting fantansy role playing game in the Dunjonquest tradition that started with "The Temple of Apshai" and continued with "Datestones of Ryn." This is the deadliest Dunjonquest yet! $14.95

"Datestones of Ryn" $14.95

"Temple of Apshai" $24.95

Supplied on disk for an additional $5 per program.

Dungeon

from Chameleon

A full fantasy & adventure where you create your character, choose from among 26 types of armor and 80 weapons, and search for fame and fortune. In each dungeon there are random events which occur, but skill most often determines the outcome. The program has many prompts and the twenty page, illustrated manual explains the game. Requires 32k and two disk drives. $34.95

Air Flight

from Instant Software

Transform your TRS-80 into a cockpit full of elaborate controls. Take off, land and do acrobatics once you learn how to fly. $7.95

Air Raid

from Small Systems

Real-time, "shooting gallery" game which is better than the arcades. In machine language for excellent graphics and endless hours of fun. ~~$14.95~~ NOW only $9.95

DDT

Disk Drive Timer

by Disco-Tech

Analyze and adjust your disc drive motor speed with real-time graphic display. Manual details use for Radio Shack, Shugart, MPI, Pertec, and Vista drives, but will work with any drive. All you need is DDT, two screwdrivers and five minutes. On disk for $19.95

Galactic Trader

by Douglas Carlston

Trade your way to a fortune in a three dimensional universe. You must barter, deal with the energy cartel and watch for spies. This is truly an outstanding computer game by the author of "Galactic Empire." $14.95

Also available "Galactic Empire" for $14.95

Space War

by Device Oriented Games from Acorn

Two-player, real-time space battle lets each player control a spaceship with rotate, thrust, fire and hyperspace. Five game options (including gravity) and three speeds included. In machine language. $9.95

Calculator Plus

from Manhattan Software, Inc.

Turn your TRS-80 into a calculator. This program has optional printer features, performs all calculations in double precision and will chain and mix calculations. Optional dollar format. $9.95.

STRUCTURED BASIC TRANSLATOR

by Gene Bellinger

Try structured programming. You can write programs using PROCEDURES, CALLS, CASE-CALLS, IF-THEN-ELSE, WHILE and UNTIL. Once written, SBT will quickly translate the structured code into an effecient BASIC program. Speeds up program development and documentation. The program is both fast (a 20k BASIC program in less than 4 minutes) and compact. Requires 32k and one disk drive. Supplied on disk for $29.95.

Biocurve

by Microflair from Hayden

Graph your biorhythms and calculate your critical days. Compare yours with a friend. $9.95

CREDIT CARD CALLERS MAY PHONE US 24-HOURS A DAY, OR CLIP THIS COUPON AND MAIL YOUR ORDER TODAY.

THE PROGRAM STORE
4200 Wisconsin Ave NW
PO Box 9609 Dept K 8
Washington, D.C. 20016

please send me these TRS-80 programs:

title	price
	postage: $ 1.00
	total:

name:

address:

city, state & code

Check payable to The Program Store

MASTERCHARGE mc bank code:

VISA exp date:

card number:

signature:

(Continued on next page)

LPRINT the screen by pressing I, J, K simultaneously.

Screenprint

Louise Frankenberg
1289 Magothy Road
Pasadena MD 21122

Here's a short (66 bytes) assembly language program for level II DOS or DISK BASIC that will send the contents of your screen to your printer at the push of a button! Well, three buttons: I,J,K pushed simultaneously.

Use it to print directories, sample BASIC runs, or whatever is visible on the screen in the middle of a BASIC program. The program itself pauses while the screen is printed and then continues where it left off.

The program is compatible with any printer-driver. Its only printing limitations are machine-language programs such as DEBUG, since these inhibit the usual keyboard-scan routine, and, of course, it can't reproduce graphics on a non-graphics printer!

Program Notes

Change the ORG in line 145 according to the amount of RAM in your system. You'll probably want to keep it at the top of memory so it's out of the way; mine fits in 48K just below a printer-driver program. If you do move the ORG, you will also have to change the address in line 192 to an address just below it.

Lines 192 and 194 are optional except for TRSDOS 2.2 users. What they do is to fool the computer into thinking that your RAM ends just below SCRNPRT's starting address. This means that you don't have to bother answering MEMORY SIZE? when entering BASIC if SCRNPRNT is the only (or lowest) machine-language program you plan to use; I've started making a habit of adding these two instructions to every program so that I don't have to remember different MEMORY SIZE's.

The lines are an absolute necessity if you use TRSDOS 2.2 and want your program at the very top of memory, because for some unfathomable reason Radio Shack decided to use the top 64 bytes of any size RAM for housekeeping, and whatever resides there is wiped out when DISK BASIC is loaded!

Lines 192-224, START to ENTRY, initialize the program. First SCRNPRNT finds the printer-driver address at 4026H and adds it to the program, so it can be called later. The program then changes the keyboard driver pointer in 4016H to point to ENTRY so that whenever a key is pushed the computer comes to this program instead of to the regular keyboard driver.

```
                00100 ;'SCRNPRNT' BY LOUISE H. FRANKENBERG, 20 OCT 79
                00110 ;
                00120 ;SENDS ENTIRE CONTENTS OF SCREEN TO PRINTER WHEN KEYS
                00130 ;I, J, AND K ARE PUSHED SIMULTANEOUSLY.
                00140 ;
FDF5            00145         ORG     0FDF5H          ;MEMORY SIZE 65013
03E3            00150 KEYSCN  EQU     03E3H           ;KEYBOARD SCAN DRIVER
3802            00160 KEYS    EQU     3802H           ;KEYBOARD MEM. FOR I,J,K
4016            00170 KYPNTR  EQU     4016H           ;KEYBRD DRIVER ADDR. PNTR
4026            00180 PRPNTR  EQU     4026H           ;PRNTER DRIVER ADDR. PNTR
                00190 ;
FDF5 21F0FD     00192 START   LD      HL,0FDF0H       ;ADR. JUST BELOW ORG
FDF8 22D10F     00194         LD      (4049),HL       ;CHANGE 'TOP OF RAM'
FDFB 2A2640     00200         LD      HL,(PRPNTR)     ;PRINTER DRIVER ADDR.
FDFE 2222FE     00210         LD      (PR1+1),HL
FE01 222AFE     00220         LD      (PR2+1),HL
FE04 210DFE     00221         LD      HL,ENTRY        ;CHANGE KEYBRD DRIVR PNTR
FE07 221640     00222         LD      (KYPNTR),HL     ;   TO ENTRY OF THIS PROG
FE0A C32D40     00224         JP      402DH           ;JMP TO DOS (1A19 LEV.II)
FE0D 3A0238     00230 ENTRY   LD      A,(KEYS)        ;GET KEY PUSH
FE10 FE0E       00240         CP      0EH             ;SEE IF IJK
FE12 C2E303     00250         JP      NZ,KEYSCN       ;GO TO ROM DRIVER IF NOT
                00260 ;PRINT WHOLE SCREEN
FE15 F5         00262         PUSH    AF              ;SAVE REGISTERS
FE16 C5         00264         PUSH    BC
FE17 D5         00266         PUSH    DE
FE18 E5         00268         PUSH    HL
FE19 21003C     00270         LD      HL,3C00H        ;START OF SCREEN MEMORY
FE1C 1610       00280         LD      D,16            ;NUMBER OF LINES
FE1E 0640       00290 NXTLIN  LD      B,64            ;LENGTH OF LINE
FE20 4E         00300 NXTCHR  LD      C,(HL)          ;GET CHAR. TO BE PRINTED
FE21 CD0000     00310 PR1     CALL    $-$             ;PRINT IT
FE24 23         00320         INC     HL              ;NEXT SCREEN ADDR.
FE25 10F9       00330         DJNZ    NXTCHR          ;DO UNTIL WHOLE LINE DONE
FE27 0E0D       00340         LD      C,0DH           ;CARRIAGE RETURN
FE29 CD0000     00350 PR2     CALL    $-$             ;DO CARRIAGE RETURN
FE2C 15         00360         DEC     D
FE2D C21EFE     00370         JP      NZ,NXTLIN       ;GO START NEXT LINE
FE30 E1         00372         POP     HL              ;RESTORE REGISTERS
FE31 D1         00374         POP     DE
FE32 C1         00376         POP     BC
FE33 F1         00378         POP     AF
FE34 C3E303     00380         JP      KEYSCN          ;ALL DONE- RETURN
FDF5            00390         END     START
00000 TOTAL ERRORS
ENTRY   FE0D 00230    00221
KEYS    3802 00160    00230
KEYSCN  03E3 00150    00250 00380
KYPNTR  4016 00170    00222
NXTCHR  FE20 00300    00330
NXTLIN  FE1E 00290    00370
PR1     FE21 00310    00210
PR2     FE29 00350    00220
PRPNTR  4026 00180    00200
START   FDF5 00192    00390
```

Program Listing: SCRNPRINT

The last statement in the section is a jump back to DOS READY or BASIC READY. Leave it as it is for DOS, but if you don't have a disk, change it to 1A19H.

Whenever a key is pushed, SCRNPRNT checks to see if I, J, and K have been pushed simultaneously. If not, control is immediately returned to the regular ROM keyboard-scan routine. If so, the whole screen is printed out with a simple nested loop: HL is set to the upper-left-corner of the screen, and one character at a time is put in C and sent to the printer. A carriage-return is sent after each screen line.

If your printer also requires a line-feed, make the following changes: Add a 'LD (PR3+1), HL' statement to the initialization section and add the following two statements after PR2: 'LD C,0AH' and 'PR3 CALL $-$'.

When the entire screen has been printed, line 380 causes a jump back to the ROM keyboard-scan routine.

Using SCRNPRNT

Printing from the screen is easy! If you're using a printer-driver program, be sure to initialize it first, so that SCRNPRNT will find the correct address. If you don't have a disk, load the program as a SYSTEM tape and press '/'. You will be returned to BASIC READY. With a disk, run the program under DOS right after power-up or after your printer-driver has been initialized. Then, whenever you want a printout of what's on the screen, just press IJK simultaneously and you've got it! ■

HOBBYWORLD® ELECTRONICS

America's Largest Mail-Order Computer Store

CALL TOLL FREE:
(800) 423-5387 USA
IN CALIF:
(800) 382-3651
LOCAL & OUTSIDE USA
(213) 886-9200

BBB

Electronics Books from HOBBYWORLD

INTRODUCTION TO TRS-80 GRAPHICS

Provides complete intro to basics of graphics programming using real examples which run on TRS-80. Book begins with basic concepts of line drawing and leads to geometric shapes, figure animation, and more advanced topics. No math required, but familiarity with BASIC assumed. A TRS-80 for running examples helpful, but most concepts apply to all low cost computers with graphics capabilities. (175 pages) **$8.95**
Cat No. d-918398-18-5

MICROSOFT BASIC

Complete introduction on programming in Basic. Illustrates concepts with examples that run using MITS family of BASIC interpreters. Commands are valid at each succeeding level of BASIC. Can save valuable programming time and effort. Book assumes no prior programming experience, but basic understanding of computer fundamentals would be useful. (225 pages)
Cat No. d-918398-23-1 **$9.95**

HOW TO GET STARTED WITH CP/M **$7.95**

Having trouble understanding CP/M? Book will get you into essentials in a few easy steps. Most popular operating system explained in simple terms, glossary, list of hardware manufacturers supporting CP/M and list of CP/M Software.
Cat No. d-918398-32-0 (200 pages)

REAL TIME BASIC FOR TRS-80

Will teach you how to read and understand programs in BASIC. Book introduces control concepts using keyboard and screen. Learn how to interface TRS-80 to other devices and wire some specific circuits. (150 pages)
Cat No. d-918398-05-3 **$6.95**

PASCAL WITH STYLE

A guide written for PASCAL users to help write more accurate, error-free programs, guidelines that stress program organization and "logical thinking", detailed set of standards. Samples of PASCAL programs are among the most readable seen anywhere. Special chapter shows how to use top-down approach with PASCAL.
Cat No. H5124 **$6.95**

TELEPHONE ACCESSORIES YOU CAN BUILD

May prove to be one of the most popular books in technical library. Remote ringing . . . group listening, speaking, speech scrambler, dialing # of police and fire departments . . . burglar alarm . . . phone silencer . . . these are some useful phone accessories you can build for a fraction of commercial cost. Fully illustrated, step-by-step instructions. Info on phone regulations and basic theory.
Cat No. H5748 **$5.95**

ANNOUNCING

The nation's most popular computerized toys and games are now available direct from Hobby-World® Electronics. Name Brand Products include: Atari Home Video Game Center, T.I. Speak and Spell, Hand Held Baseball, Football, Fidelity's "Chess Challenger" and many others.

SEND FOR FREE FLYER FEATURING:

Our new line of Computerized Toys and Games, Computers and Accessories, Disk Drives, Printers, Integrated Circuits, LED's Semi Conductors, Connectors plus more and more. Select from Hundreds of Items at Terrific Prices! Circle our Reader Service Number or Phone/Write for your Free Illustrated Flyer Today!

TRS-80/APPLE/EXIDY 16K Memory Add-on Kit

Everything needed to up-grade your TRS-80, Etc. An additional 16K includes illustrated instructions, RAMS and pre-programmed jumpers. No special tools required. Wt. 4 Oz. **$88.00**

CAT. NO.	DESCRIPTION
1156	TRS-80 Keyboard Unit
1156-A	TRS-80 Exp. Interface Prior to 4/1/79
1156-B	TRS-80 Exp. Interface After 4/1/79
1156-C	For Apple II
1156-D	For Exidy Sorcerer

FULLER ELECTRONICS RF-2
TRS-80 DUAL CASSETTE CONTROL CENTER **$74.95**

- *Aids and simplifies cassette loading!*
- *Allows you to listen to cassettes as they are loading!*

No more plugging and unplugging cassette cables! Saves wear and tear on your computer system, adds the flexibility that you need for level II data storage! Internal speaker allows you to listen to CSAVE and CLOAD, also allows you to listen to fast forward and rewind (CTR-41). Independent volume control lets you adjust monitor volume without affecting loading volume. Built-in LED's let you know status of one or both recorders. Now you can simultaneously CSAVE with both recorders; CLOAD with either recorder; cross tape with both recorders. Tested and reviewed in March '79 Creative Computing. Compact, only 4" x 3¼" x 2¼" high. Completely assembled and tested Wt 16 oz.

Cat No. 1973 For CTR-41
Cat No. 1974 For CTR-80 or CTR-21

5¼" DUAL AND SINGLE DISK DRIVE ENCLOSURES as low as $25

- *Fits most standard disk drives: Shugart, MPI, Pertec and more!*

Dual drive enclosure constructed of hi-impact plastic, one piece molded construction, charcoal-grey finish. Single drive enclosure constructed of rugged beige-finish steel, two piece construction.

Cat No.	Description	Wt	Price
2055	Dual Enclosure	2 lbs	$59
2245	Single Enclosure	1 lbs	$25

LEEDEX Video 100-80

The model 80 features an industrial grade metal cabinet with built-in disk mounting capability and space for an 11"x14" PC board for custom designed electronics.

The solid state circuitry assures a sharp, stable, and trouble-free picture. The front panel controls include power, contrast, horizontal hold, vertical hold, and brightness. Adjustments for size, video level, and width are located on the rear panel.

Rugged metal cabinet with disk space.
Cat No. 2374 **$199**

SHUGART SA-400
5¼" MINIFLOPPY DISK DRIVE

Industry standard in disk drives! Hard and soft sectoring, 35 track, single density, 35 msec accessing, compact; only 5.75" x 3.25" x 8" deep, features direct drive stepping motor actuator and design which assures read/write accuracy. Power requirements: +12VDC at 1.5 amps; +5VDC at 0.5 amps. Less cabinet. Wt. 4 lbs. With manual.

Cat. No. 1154 SA400w/manual $285
Cat. No. 1154D Manual only $2.50

FULLER ELECTRONICS RF-3
TRS-80 I/O & AUXILIARY CONTROL CENTER

- *Controls lights, video recorders, appliances thru your TRS-80!*
- *With complete application software and documentation*
- *Built-in relay protects your TRS-80 relay!*
- *Controls from fractions of a second to months!*

The most versatile TRS-80 accessory available today! Offers features found only in units costing much more. Now you can control lights when you're not home, at random times! Turn video recording equipment on and off at preset times, even months in advance. Internal beeper can be used to signal the end of a long sort. It also lets you know with one beep, two beeps, etc. exactly what part of the program you're in! Also allows you to check your RS-232 output without turning on the printer. Built-in internal relay protects and bypasses control current from your sensitive TRS-80 relay. Now you can even operate 6 or more cassette recorders safely for CSAVE. NOTE: 110VAC CONTROLLER IS NOT INCLUDED. A simple schematic and parts list is supplied. Size: 2" x 4" x 1½".

$39.95 Wt 12 oz. Cat No. 1972

MICRO SQUARED M²-250
DOUBLE SIDED DISK DRIVES

Two "Double Sided" drives yield 358K Bytes on TRS-80, 875K Bytes on S-100 systems. Fast transfer rate of 250K Bytes/Sec. No modification of hardware or software necessary. Complete with Power Supply and Cabinet, Factory Tested. Life Expectancy 20,000 hours, average media life 3 million passes on a single track. Pertec Drives.

CAT. NO.	DESCRIPTION	WEIGHT	PRICE
2054	M2-250 Disk Drives	15 lbs.	$1195.00
2329	#550-16 Hard Sector Diskettes	10 oz.	$59.95bx(10)

VERBATIM 5¼" DISKETTES
as low as $33.00 box of 10
5 boxes for as low as $150

525 SERIES

- *Double density!*
- *Single Sided!*
- *Perfect for commercial and general applications!*

Cat No.	Type	Description	Use for	Price
1147	525-01	Soft Sector	TRS-80	$33.00
1148	525-10	10 hole, hard	North Star, Apple	$33.00
1149	525-16	16 hole, hard	Micropolis	$33.00

550 SERIES

- *Quad density! (double sided, double density)*
- *For commercial and general applications!*

Cat No.	Type	Description	Use for	Price
1492	550-01	Soft Sector	SA450,MP152,	$59.95
2328	550-10	10 hole hard	BASF, Wangco,	$59.95
2329	550-16	16 hole hard	Micro-2	$59.95

577 SERIES

- *Certified twice!*
- *Single sided, double density!*
- *For critical data and applications!*
- *Built-in hub protector ring!*
- *Certified 77 tracks!*

Cat No.	Type	Description	Use for	Price
2330	577-01	Soft sector	TRS-80, etc	$49.95
2331	577-10	10 hole hard	North Star, Apple	$49.95
2332	577-16	16 hole hard	Micropolis, etc	$49.95

TRS-80 EDGE CONNECTOR & CABLE

- *40 pin (2x20)*
- *36" of ribbon cable attached*

Standard connector and cable set plugs into rear of main unit or expansion interface. Factory preassembled.
Cat. No. 2037 **$8.88ea**
Wt. 8 oz.
2 for $16.00

TV MODULATOR (kit)

Assembles in minutes, installs in minutes!

Accepts composite video output from computers, TV games, etc. Requires 6VDC, operates on any standard B&W TV, channel 2 or 3. Completely shielded. Not to be used where prohibited by law. Wt 4 oz.
Cat No. 1975
$4 3 for $10

TRS-80 EDGE CONNECTOR

- *40 Contacts (2x20)*
- *1" Spacing*

For expansion or main unit

Cat. No.	Description	Weight	Price
1980	TRS-80 Edge Connector	2 oz.	$4.75
2147	40 Cond. Cable 3 ft. length	4 oz.	$6.00
2147	40 Cond. Cable 10 ft. length	12 oz.	$16.00
2147	40 Cond. Cable 25 ft. length	30 oz.	$30.00

INTRO TO 8080 & Z-80 ASSEMBLY LANGUAGE PROGRAMMING

An introductory look at assembly language programming for 8080/Z-80 processors. Intended to provide about everything the programmer needs to get the most out of his or her machine. Programming techniques are presented along with instructions. Diagrams and examples provided. Exercises, with answers, included with each chapter.
Cat No. H5167 **$7.95**

TRS-80 LOWER CASE MODIFICATION KIT **$19**

Modifies your machine to display both upper and lower case. Installs in minutes, requires drill, soldering iron and screwdriver. With complete instructions. Cat No. 1550

HOW TO ORDER

Pay by check, Mastercharge, Visa, or COD. Charge orders please include expiration date. Foreign pay in U.S. funds. Order by phone or mail, or at our retail. MINIMUM ORDER $10. please include phone number and magazine/issue you are ordering from. Prices valid thru last day of cover date. SHIPPING: USA: add $2.00 for the first 2 lbs. For ground add 35¢ for addt'l lb. For air add 75¢ for addt'l lb. FOREIGN: surface: add $3.00 for first 2 lbs, 60¢ per addt'l lb. Air: add $11.00 for first 2 lbs, $5 for each addt'l lb. CODs $1 addt'l. Guaranteed satisfaction for 120 days or your money back! Not responsible for typographical errors. Some items subject to prior sale. We reserve the right to limit quantities.

19511 Business Center Dr. Dept. V5 Northridge, Ca. 91324

Reader Service—see page 162

. . . We have them ALL

All the Disks and Printers that interface to the TRS-80®

5¼" Drives — 8" Drives — Single and Double Density — and Hard Disk Systems with up to 40 megabytes!

TRS-80 Disk Drive

SAVE OVER $100

Vista, Percom, Lobo, and others

Fully compatible with expansion interface and TRSDOS Software

Some as Low as

$359

CORVUS 10 megabyte hard disk for Mod. I or II

ONLY $4795

CONVERT YOUR EXISTING SELECTRIC TYPEWRITER TO A PRINTER . . .

Power Supply & electronics, assembled & tested. You make only a simple solenoid installation (or have the factory do it). Manufactured by Escon.

Parallel version, List $575 . . . . . ONLY $514

TRS-80 cable (specify MOD I or MOD II) $25

CP/M® OPERATING SYSTEM

for Mod. I List $145 . . . . $129
for Mod. II List $170. . . . . $149

Complete line of CP/M Software available at discount prices.

CENTRONICS PRINTERS

NEW 730
w/friction and tractor

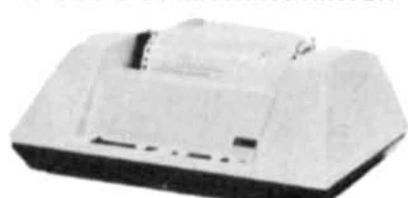
$779†

779-1 (TRS-80 Line Prtr. 1), List $1245	949*
779-2 with tractor, List $1350	1049*
702-2 120 cps, bi-directional, tractor, VFU	1995
703-2 185 cps, bi-directional, tractor, VFU	2395

(shipped freight collect)

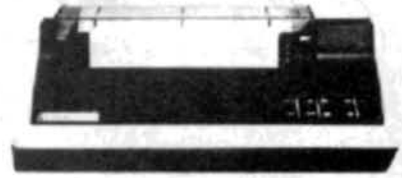
ANADEX

80-COL. DOT MATRIX PRINTER



Complete upper and lower case ASCII char. set, bi-directional at 84 lines/min. Features RS232 20/60 mil current loop and Centronix parallel interface. Ideal for use with TRS-80, Sorcerer, Cromemco, and North Star systems.

OUR PRICE ONLY $749
(shipping $10)

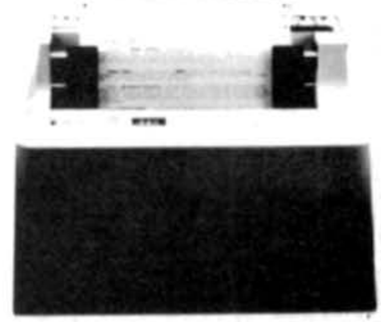


Bi-directional; 150 cps; logic seeking; adjustable tractor. Available with lower case compressed print, forms length control or vertical forms control option. Centronics style parallel interface also available.

T.I. 810 with serial/parallel interface
List 1940 . . . . OUR PRICE $1735

T.I. 810 printer outperforms Line Printer III.

PAPER TIGER®



Paper Tiger, List $995 ONLY $895

w/graphics options, incl. buffer, $1194 . . .	$989
IP-225 w/1210, 1250 options, List $984	834
IP-225 w/tractor, 1210*, 2K buffer, & graphics options, List $1098	899
TRS-80 cable	45

*1210 option is expanded/compressed print.

NEC SPINWRITER™

The Fantastic Letter-Quality Printer at 55 cps

5530 RO/Centronics Interf., $3085 **$2689**
(shipped freight collect)

Printers for TRS-80 require Level II machines. Printer cables extra. Call for price and order number.

* Same as Line Printer I
† Same as Line Printer II

SHIPPING AND INSURANCE: Add $5 for Selectric Converter, $6 for disk drives, $10 for Megabox. Centronics printers shipped freight collect. Contact us for shipping information on other printers.

All prices subject to change and all offers subject to withdrawal without notice. Prices in this ad are for prepaid orders. Slightly higher prices prevail for other-than-prepaid orders, i.e., C.O.D., credit card, etc.

— WRITE FOR FREE CATALOG —

MiniMicroMart, Inc. ✓24

1618 James Street, Syracuse, NY 13203 (315) 422-4467 TWX 710-541-0431

master charge

VISA

If you enjoy driving, you're going to get a COMPUCRUISE. Once you see what it can do, you just won't be able to live without it.

This gadget fits into most dashboards . . . no strain even in a tiny sports car like the Mazda RX-7 . . . and once you have it, every trip is like flying a 747. The darned thing tells you the time, how fast you're going, how far you've been on this trip or since the last regassing, how many miles per gallon you're getting, either at the instant or the average on the trip . . . or gallons per hour at the moment or for the trip . . . temperature outside . . . inside (or coolant temperature, if you prefer) . . . oh, it has an elapsed time for the trip, a stop watch, lap time, an alarm . . . how much further for your trip, how many gallons more the trip will take, how much longer for the trip at your present average speed . . . yes, it gives you your average speed for the trip. You prefer it in metric, no strain . . . liters remaining, etc. Did we mention that it also has cruise control either at a speed set on the control board or at whatever speed you are traveling? The Compucruise will keep you busy and entertained during any trip . . . telling you more than you will ever want to know.

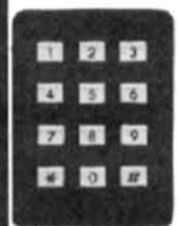
The Compucruise is not difficult to install . . . though it does connect to everything except the cigarette lighter. Until you've tried computerized travel, you haven't found out how much fun driving can be. It will work on any car not having fuel injection . . . and there is a front-wheel drive accessory gadget available for only $4.40-#P001 (regularly $5.50).

The price for the Compucruise is regularly $199.95 . . . and a bargain at that price. We'll sell you one of these fantastic gadgets for $159.95 with cruise control (Model 44-#P002), and $127.95 without (Model 41-#P003). Send money . . . and start having fun!

MOM'S

MAIL ORDER MICROS ✓158

Dept. 580 • PO Box 427 • Marlboro NH 03455
Phone: (603) 924-3041*



Steal Stopper

Ever had your car stolen?
The first reaction is one of disbelief . . .
. . . you *know* it was *right there*!

What you want is a modern combination lock on your ignition . . . The Steal Stopper. It's easy to install and almost impossible to defeat. You can by-pass it, if you want, for parking attendants or a car wash. Other than that, you get a secret four digit code and only will then be able to start the car . . . even if you have the keys in the ignition.

This protection retails for $50 . . . but we have a special for you at $39.95. Don't procrastinate. Order #P004.

Note: This product works best on Detroit cars. Mazda RX7 owners must order additional module, #P008, which costs $8. The Steal Stopper can be modified for Mercedes, Porsche, Ferrari, or other high performance European cars by returning unit to manufacturer with $3. They promise quick modification and return.

SPECIAL 10% OFF REGULAR PRICES

#1 ADVENTURELAND
YOU'LL WANDER THROUGH AN ENCHANTED LAND ENCOUNTERING WILD ANIMALS AND MAGICAL BEINGS WHILE YOU TRY TO RECOVER LOST TREASURES.
Order #SA001T for TRS Level II 16K, #SA001S for Sorcerer 16K, #SA001A for Apple 24K—$13.45 each, on cassette

#2 PIRATE ADVENTURE
SAIL TO TREASURE ISLAND AND TRY TO RECOVER LONG JOHN SILVER'S LOST TREASURES.
Order #SA002T for TRS Level II 16K, #SA002S for Sorcerer 16K, #SA002A for Apple 24K—$13.45 each, on cassette

#3 MISSION IMPOSSIBLE ADVENTURE
SAVE THE WORLD'S FIRST AUTOMATED NUCLEAR REACTOR WHEN YOU COMPLETE YOUR MISSION.
Order #SA003T for TRS Level II 16K, #SA003S for Sorcerer 16K, #SA003A for Apple 24K—$13.45 each, on cassette

#4 VOODOO CASTLE
SAVE COUNT CRISTO FROM THE FIENDISH CURSE OR FOREVER BE DOOMED—BEWARE THE VOODOO MAN.
Order #SA004T for TRS Level II 16K, #SA004S for Sorcerer 16K, #SA004A for Apple 24K—$13.45 each, on cassette

#5 THE COUNT
LOVE AT FIRST BYTE FROM YOUR BIG BRASS BED IN TRANSYLVANIA.
Order #SA005T for TRS Level II 16K, #SA005S for Sorcerer 16K, #SA005A for Apple 24K—$13.45 each, on cassette

#6 STRANGE ODYSSEY
YOU'RE MAROONED AT THE GALAXY'S EDGE AND DISCOVER RUINS OF AN ANCIENT ALIEN CIVILIZATION AND TRY TO COPE WITH UNEARTHLY TECHNOLOGIES WHILE YOU AMASS FABULOUS TREASURES.
Order #SA006T for TRS Level II 16K, #SA006S for Sorcerer 16K, #SA006A for Apple 24K—$13.45 each, on cassette

#7 MYSTERY FUN HOUSE
FIND YOUR WAY THROUGH THE STRANGEST FUN HOUSE BEFORE THE WEIRD PARK CLOSES.
Order #SA007T for TRS Level II 16K, #SA007S for Sorcerer 16K, SA007A for Apple 24K—$13.45 each, on cassette

#8 PYRAMID OF DOOM
AN EGYPTIAN TREASURE HUNT THROUGH A NEWLY UNCOVERED PYRAMID, COMPLETE WITH ANCIENT CURSE.
Order #SA008T for TRS Level II 16K, #SA008S for Sorcerer 16K, #SA008A for Apple 24K—$13.45 each, on cassette

BRAND NEW* TRS-80 and accessories at FANTASTIC SAVINGS

LEVEL II 16K COMPLETE	$720	#TRS-001S
16K EXPANSION UNIT	$400	#TRS-002S
32K EXPANSION UNIT	$525	#TRS-003
DISK DRIVES	$425 each (Specify which drive you want)	#TRS-004
FRICTION MODEL PRINTER	$870	#TRS-005S
TRACTOR FED MODEL PRINTER	$1350	#TRS-006
LINE PRINTER III (AND CABLE)	$1550	#TRS-007
MOD II 64K	$3400	#TRS-008

TERMS: Shipment normally within one week of receipt of your order (with cashier's check, money order, or credit card) for microcomputer and three weeks for accessories (checks take two weeks extra to clear bank). ADD $2.50 PER ITEM for HANDLING. Everything will be sent to you with UPS freight charges COLLECT.

**NOT UPGRADED USED OR RECONDITIONED LEVEL I's WITH OLD KEYBOARDS BUT BRAND SPANKING NEW TRS-80's IN FACTORY CARTONS WITH FULL FACTORY WARRANTY! COMPARE PRICES AND QUALITY AND ORDER FROM MOM's.*

ASTROLOGICAL COMPUTER AND 4-FUNCTION CALCULATOR

ASTRO®

Gives you a fascinating look into your personality traits. Compares 16 different combinations. Matches your astrological influences to any day—past, present, or future. Gives in-depth analysis of your compatibility with your boss, your spouse, your lover, your child. Regularly $49.95. #P0020—$39.95 **SPECIAL PRICE.**

ASK MOM

For details of October 1980 Asian and European Electronics Tours. See January Wayne Green editorials in Kilobaud Microcomputing and 73 Magazine. Info #002 . . . No charge.

HONEYWELL

15 **Honeywell ASR-33 Communications Consoles** with TTY, paper tape reader and punch. Used, working when removed from service. Shipped freight collect or you pick up. Weight 300 lbs. $395. Order #P006.

INSTANT SOFTWARE
HALF PRICE SPECIAL
CLOSEOUT—ONLY $4

TRS-80, Level I, Games
Knights Quest/Robot Chase 4K-#ISI0003.
Cave Exploring 16K-#ISI0010.
Doodles & Display 16K-#ISI0030.
Fun Package I 16K-#ISI0041.

TRS-80, Level I, Finance
Status of Homes 4K-#ISI0012.

TRS-80, Level II, Hobby
Model Rocketry Analyzer-#ISI0024.

PREVIEW

WHAT TO LOOK FOR IN THE JUNE 80

RANDOMNESS

The ins and outs of randomness are explored in an ordered manner. Check the distribution of your randomness with a bar graph demonstration program.

INSIDE LEVEL I

An indepth look at the power and potential of the Level I ROMs. Often scorned by converts to Level II, the basic BASIC is still a language to be reckoned with.

TESTING 1, 2, 3

Hook up your 80 to the outside world, via its parallel ports, and input measurements directly to your program. Get rid of pen and graph paper!

LIFE

Is there life within your computer? There will be next month, with the game of Life. Watch thousands of generations flash before your eyes in minutes. If you don't like the way evolution is progressing, then stop the action and play "GOD"!

ALL THIS PLUS MUCH MORE!

ADVERTISERS

*** Reader Service inquiries not honored. Please contact advertiser directly.**

Ask for Instant Software at a computer store near you.

Alabama
Anderson Computers
3156 University Dr., Huntsville
Computerland of Huntsville
3020 University Dr., Huntsville
Olensky Bros.
3763 Airport Blvd., Mobile

Arizona
Ham Shack
450 6 A N. 16th St., Phoenix
Millets TV & Radio
621 East Broadway, Mesa

California
Byte Shop
8038 Clairmont Mesa Blvd., San Diego
Byte Shop of Mt. View
1415 West El Camino Real, Mt. View
Byte Shop of Sacramento
6041 Greenback Ln., Citrus Heights
Capital Computer Systems
3396 El Camino Ave., Sacramento
Computers Made Easy
819 East Ave. Q-9, Palmdale
Computer Store of San Leandro
701 MacArthur Blvd., San Leandro
Computer World
6791 Westminster Ave., Westminster
Computerland
16720 S. Hawthorne, Lawndale
Computerland of W. LA
6840 La Cienega Blvd., Inglewood
Coast Electronics
3118 No. Main St., Morro Bay
Computerland
24001 via Fabricante No 904, Mission Viejo
Hobbi-tronics
1378 So. Bascom Ave., San Jose
Hobby World
19511 Business Ctr. Dr., Unit 6, Northridge
I.C.E. House Inc.
398 North E. St., San Bernardino
Jade Computer Products
4901 W. Rosecrans, Hawthorne
Opamp/Technical Books
1033 N. Sycamore Ave., Los Angeles
Q.I. Computers, Inc.
15818 Hawthorne Blvd., Lawndale
Radio Shack Dealer
8250 Mira Mesa Blvd., San Diego
Radio Shack Dealer
50 N. Cabrillo Hwy., Half Moon Bay
Santa Rosa Computer Center
604 7th St., Santa Rosa
Silver Spur Elect. Comm.
13552 Central Ave., Chino
The Computer Store
820 Broadway, Santa Monica

Colorado
Byte Shop
3464 S. Acoma St., Englewood
Colorado Computer Systems
311 W. 74th Ave., Westminster
Computerland of North Denver
8749 Wadsworth Blvd., Arvada
Computer Shack
1635 South Prairie, Pueblo
The Computer Store
2300 Welton St., Denver

Connecticut
American Business Computers
454 Thames St., Groton
Computerlab
130 Jefferson, New London
Computerland
1700 Post Rd., Fairfield
Computer Works
1439 Post Rd. E., Liberty Plaza, Westport

D.C.
The Program Store
4200 Wisconsin Ave., N.W., Washington, D.C.

Florida
Adventure International
200 Bald Cypress Ct., Longwood
AMF Electronics
11146 N. 30th St., Tampa
Boyd-Ebert Corporation
1328 West 15th St., Panama City
Computer Center
6578 Central Ave., St. Petersburg
Computerland of Ft. Lauderdale
3963 N. Federal Hwy., Ft. Lauderdale
Computerland of Jacksonville
2777-6 University Blvd. W., Jacksonville
Computer Shack
3336 Beach Blvd., Jacksonville
Curtis Waters Enterprises
236 Talbot Ave., Melbourne
Heath Kit Electronic
405 W. 16th Ave. Center, Hialeah
HIS Computermation
1295 Cypress AvTalbot Ave., Melbourne
Heath Kit Electronic
4705 W. 16th Ave. Center, Hialeah
HIS Computermation
1295 Cypress Ave., Melbourne
Sound Ideas
2201 C N.W. 13th, Gainesville
Ukatan Computer Store
Airport Rd., Destin
Williams Radio & TV Inc.
2062 Liberty St., Jacksonville

Georgia
Atlanta Computer Mart
Atlanta
Computerland of Atlanta
2423 Cobb Parkway, Smyrna

Hawaii
Computerland of Hawaii
567 N. Federal Hwy., Honolulu
Radio Shack Assoc. Store
1712 S. King St., Honolulu

Idaho
Electronic Specialists
6411 Fairview Ave., Boise

Illinois
Computerland
4507 North Sterling, Peoria
Computerland
9511 N. Milwaukee Ave., Niles
Computer Station
3659 Nameoki Rd., Granite City
Midwest Micro Computers, Inc.
708 S. Main St., Lombard

Indiana
Computer Center of South Bend
51591 US 31 North, South Bend
Data Domain
221 W. Dodds, Bloomington
Fall Creek Electronics Store
732 Center St., Pendleton

Iowa
Memory Bank
4128 Brady St., Davenport

Kansas
Central Kansas Computers
6 S. Broadway, Herington

Louisiana
Computer Shoppe Inc.
3225 Danny Park, Suite 222, Metairie

Maine
Radio Shack
315 Main Mall Rd., So. Portland

Maryland
Jack Fives Electronics
4608 Deblien Circle, Pikesville
The Comm Center
9624 Ft. Meade Rd., Laurel

Massachusetts
ComputerCity
175 Main St., Charlestown
ComputerCity
50 Worcester Rd., Framingham
Computerland of Boston
214 Worcester Rd., Wellesley
Computer Packages Unlimited
244 W. Boylston St., West Boylston
Lighthouse Computer Software
14 Fall River Ave., Rehobath
New England Electronics Co.
679 Highland Ave., Needham
The Computer Store
120 Cambridge St., Burlington
Tufts Radio & Electronics
206 Mystic Ave., Medford

Michigan
Computer Center
28251 Ford Rd., Garden City
Computer Connections
38437 Grand River, Farmington Hills
Computerland of Grand Rapids
2927 28th St. S.E., Kentwood
Computerland of Rochester
301 S. Livernois, Rochester
Computerland of Southfield
29673 Northwestern Hwy., Southfield
Computer Mart
560 W. 14 Mile Rd., Clawson
Hobby House
1035 W. Territorial Rd., Battle Creek
The Alternate Source
1806 Ada, Lansing
Ye Olde Teacher Shoppe
1823 Witmyre St., Ypsilanti

Minnesota
Computerland of Hopkins
11319 Hwy F., Hopkins
Digital Den
Burnsville Center
Minnesota Software Inc.
5422 Fisher St., White Bear Lake
Zim Computers
5717 Xerxes Ave., N. Brooklin Center

Mississippi
Dyer's, Inc.
200 E. Main St., West Point
W. Vernon Foster Inc.
816 Foley St., Jackson

Missouri
Computervan, Inc.
51 Florissant Oaks Shopping Center, Florissant
Consolidated Software
16501 Greenwald Court, Belton

Minnesota
Computerland of Hopkins
11319 Hwy F., Hopkins
Digital Den
Burnsville Center
Minnesota Software Inc.
5422 Fisher St., White Bear Lake
Zim Computers
5717 Xerxes Ave., N. Brooklin Center

Nebraska
Computerland of Omaha
11031 Elm St., Omaha
Midwest Computer Co. Inc.
8625 I St., Omaha
Midwest Computer Co. Inc.
4442 S. 84th St., Omaha
Midwest Computer Co. Inc.
4403 S. 87th St., Omaha

Nevada
Century 23
4566 Spring Mountain Rd., Las Vegas

New Hampshire
Bitsnbytes Computer Center
568 Pleasant St., Concord
ComputerCity
1525 S. Willow, Manchester
Paul's TV
Main St., Fremont
Portsmouth Computer Center
31 Raynes Ave., Portsmouth
Radio Shack Assoc. Store
Fairbanks Plaza, Keene

New Jersey
Computer Encounter
2 Nassau St., Princeton
Computerland
35 Plaza Rte. #4, W. Paramus
Computer Mart of NJ
501 Rte. 27, Iselin
Dave's Electronics
Pennsville Shopping Ctr., Pennsville
Personal Computing Inc.
51 Central Sq., Linwood
Radio Shack/J&J Electronic
Mansfield Shopping Ctr.
Rt. 57 Allen Rd., Hackettstown
The Bargain Brothers
Glen Roc Shopping Center
216 Scotch Road, Trenton
The Computer Emporium
Bldg. 103, Avenues of Commerce
2428 Rte. 38, Cherry Hill

New Mexico
Antel Electronics Co.
232 Wisconsin N.E., Albuquerque
Legey and Associates
2908 Tahiti Ct. N.E., Albuquerque
Mitchell's Music (Radio Shack)
407 W. Church, Carlsbad
South West Computer Center
121 Wyatt Drive, Suite 7, Las Cruces

New York
Aristo Craft
314 Fifth Ave., NYC
Bits & Bytes
2800 Straight Rd., Fredonia
Computer Corner
200 Hamilton Ave., White Plains
Computer Era Corp.
1570 3rd Ave., New York
Computer Factory
485 Lexington Ave., NYC
Computer House, Inc.
721 Atlantic Ave., Rochester
Computerland of Nassau
79 Westbury Ave., Carle Place
Computer World
519 Boston Post Rd., Port Chester
Comtek Electronics, Inc.
2666 Coney Island Ave., Brooklyn
Comtek Electronics, Inc.
Staten Island Mall
Store 220A, Staten Island
Home Computer Center
671 Monroe Ave., Rochester
Key Electronics
Schenectady
Lashen Electronics Inc.
21 Broadway, Denville
Mr. Computer
Imp. Plaza, Rte. 9, Wappingers Falls
Soltron Systems
308 Columbia Turnpike, Rensselaer
The Computer Tree Inc.
409 Hooper Rd., Endwell
Upstate Computer Shop
629 French Rd., Campus Plaza, New Hartford

North Carolina
Byte Shop of Raleigh
1213 Hillsborough St., Raleigh

Ohio
Altair Business Systems, Inc.
5252 North Dixie Dr., Dayton
Astro Video Electronics
504 E. Main St., Lancaster
Cincinnati Computer Store
4816 Interstate Dr., Cincinnati
Computerland
4579 Great Northern Blvd., N Olmstead
Computerland
6429 Busch Blvd., Columbus
Computerland
1288 Som Rd., Mayfield Heights
Computer Store of Toledo
18 Hillwyck Dr., Toledo
Forbees Microsystems Inc.
35 N. Broad, Fairborn
Microcomputer Center
7900 Paragon Rd., Dayton
Micro Mini Computer World
74 Robinwood, Columbus
Universal Amateur Radio, Inc.
1280 Aida Dr., Columbus

Oklahoma
Sounds, Etc.
Hyw. 33, Watonga
Vern Street Products
Radio Shack Dealer
114 W. Taft St., Sapulpa

Oregon
Computerland of Portland
12020 S.W. Main St., Tigard
Computer Pathways Unlimited, Inc.
2151 Davcor St. S.E., Salem

Pennsylvania
Audio Mart
518 Fifth Ave., New Brighton
Artco Elect.
302 Wyoming Ave., Kingston
Artco Elect.
Back Mountain Shop. Ctr., Shavertown
Computer Workshoppe
3848 William Penn Hwy., Monroeville
Computerland of Harrisburg
4644 Carlisle Pike, Mechanicsburg
Erie Computer Co.
2127 West 8th St., Erie
Mighty Byte Computer Center
537 Easton Rd., Hersham
Personal Computer Corp.
24-26 West Lancaster Ave., Paoli
Personal Computer Corp.
Frazer Mall, Lancaster Ave., Frazer

Rhode Island
Computer City
165 Angell St., Providence

South Dakota
CB Radio Shack
21st and Broadway, Yankton

Tennessee
Computerlab
671 S. Menden Hall Rd., Memphis
H & H Electronics Inc.
509 N. Jackson St., Tullahoma

Texas
Computer Port
926 N. Collig, Arlington
Houston Microcomputer Tech.
5313 Bissonet, Bellaire
K.A. Elect.
9090 Stemmons Frwy., Dallas
Pan American Elect. Inc.
1117 Conway, Mission
Radio Shack Dealer
21969 Katy Freeway, Katy
Ram Micro Systems
6353 Camp Bowie Blvd., Ft. Worth

Utah
Quality Technology
470 E. 2nd So., Salt Lake City

Virginia
Home Computer Center
2927 Virginia Beach Blvd., Virginia Beach
Southside Radio Comm.
135 Pickwick Ave., Colonial Heights

Washington
American Mercantile Co. Inc.
2418 1st Ave. S., Seattle
Byte Shop of Bellevue
14701 N.E. 20th St., Bellevue
Computerland of South King Co.
1500 S. 336 St., Suite 12, Federal Way
Magnolia Micro Systems
2812 Thorndyke Ave., Seattle
Personal Computers
S 104 Freva, Spokane
Ye Old Computer Shop
1301 G. Washington, Richland

West Virginia
The Computer Corner Inc.
22 Beechurst Ave., Morgantown
The Computer Store
Municipal Parking Bldg., Charleston

Wisconsin
Byte Shop Of Milwaukee
6019 West Layton Ave., Greenfield
Computerland
690 S. Whitney Way, Madison
Petted Microsystems
4265 W. Loomis Rd., Milwaukee

Puerto Rico
The Microcomputer Store
1568 Ave. Jesus T. Pinero, Caparra Terrace

Canada
CANADIAN DISTRIBUTORS
Micron Distributing
409 Queen St., W. Toronto, Ont. M5V 2A5
Computerland of Winnipeg
715 Portage Ave., Winnipeg, Man.
Compumart
411 Roosevelt Ave., Ottawa, Ontario
Computer Mart Ltd.
1055 Yonge St., Suite 208, Toronto, Ontario
Galactia Computers
103rd Ave., Edmonton, Alberta
Micromatic Systems Inc.
101 8136 Park Rd., Richmond, B.C.
Micro Shack of W. Canada
333 Park Street, Regina, Sask.
Orthon Holdings Ltd.
12411 Stony Plain Road, Edmonton, Alberta
Total Computer Systems
Ajax, Ontario

England
Tamays & Farr Ltd.
4 Morgan St., London

France
Sideg
45 Rue de la Chapelle, Paris
Sivea s.a.
20, Rue de Leningrad, Paris

Italy
HOMIC s.r.l.
Piazza De Angeli 1, Milano

West Germany
Electronic Hobby Shop
Kaiserstr. 20, Bonn
MicroShop Bodensee
Markstr. 3, 7778 Markdorf

Australia
Computerware
62 Paisley St., Footscray VIC
Deforest Software
36 Glen Tower Drive, Glen Waverly, VIC
Softronics Micro Systems
Lindfield
Sure-Load Software
P.O. Box 26, Weston, A.C.T.

South Africa
Eddie Talberg
P.O. Box 745, Johannesburg

Instant Software Inc. Peterborough, New Hampshire 03458 603-924-7296

STARTER KIT

EXATRON STRINGY FLOPPY FOR THE TRS-80

Recommended initial purchase:

Exatron Stringy Floppy	$249.50
3 Wafers each: 5', 10', 20', 50'	40.00
Bus Extender, 2-for-1	15.00
ESF Machine Language Monitor	9.95
(Plus shipping and tax if applicable)	$314.45

SPECIAL PRICE FOR THIS STARTER KIT	$299.50
Sales Tax (California only)	
Shipping and Handling	5.50
TOTAL	

For more information see the current Exatron Stringy Floppy Owners Association Newsletter in *Microcomputing*.

Stringy Floppy is a trademark of Exatron Corporation

If you have any questions about the product, about Exatron, or ESFOA, please call the Hot Line. Address letters to ESFOA, 3555 Ryder St., Santa Clara, CA 95051

HOT LINE (For Calls Outside CA) 800-538-8559

excellence in electronics

exatron

3555 Ryder Street • Santa Clara, California 95051
(408) 737-7111