80 Microcomputing
9/80
9

September 1980
USA $2.50/DM 9.

80 microcomputing™

the magazine for TRS-80 users

Tandy Shoots Back with Three New Computers

Inside 80 gives you the breakdown on Tandy's newest market entries:

- THE COLOR COMPUTER
- THE POCKET COMPUTER
- THE MODEL III

Page 10

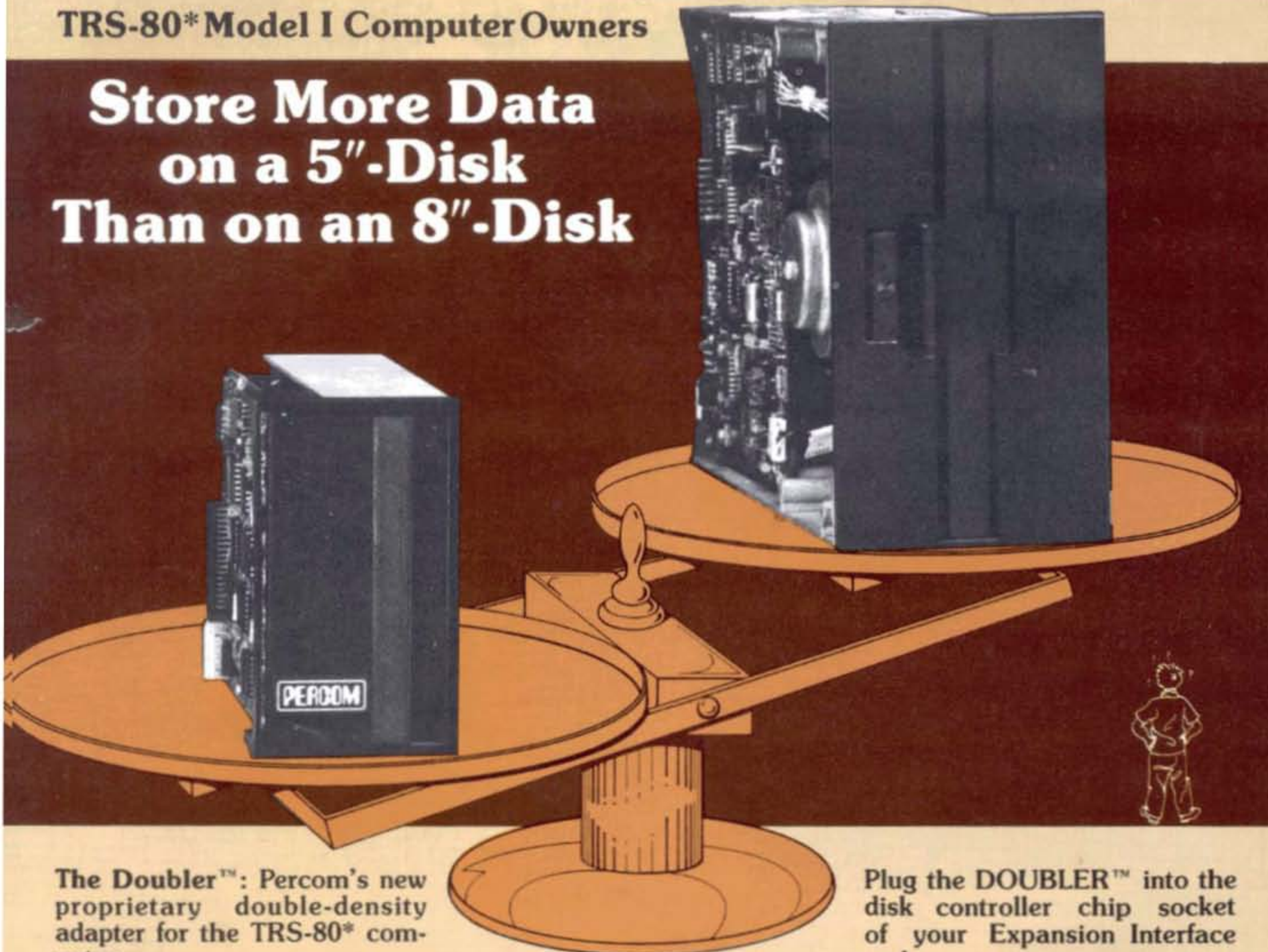
Plus:
More than 30 articles, reviews and columns
Complete contents on page 5.

Color Comes to Tandy

0 74820 08278 09

TRS-80* Model I Computer Owners

Store More Data on a 5″-Disk Than on an 8″-Disk



The Doubler™: Percom's new proprietary double-density adapter for the TRS-80* computer.

Plug the DOUBLER™ into the disk controller chip socket of your Expansion Interface and . . .

Store up to 354 Kbytes of formatted data on five-inch disks.†

- Increase **formatted** storage capacity of your minidiskettes from 1½ to almost 4 times.
- Use with standard 5-inch drives rated for double-density operation.
- The DOUBLER™ reads, writes and formats **either** single- or double-density disks.
- Proprietary design allows you to continue to run TRSDOS*, NEWDOS‡, Percom OS-80™ or other single-density software **without making any changes** to software or hardware.

Mini-Disk Systems
More storage capacity, higher reliability — from Percom, the industry leader. One-, two- and three-drive configurations in either 40- or 77-track format, starting at only $399.

- Includes DBLDOS,™ a **TRSDOS* compatible** double-density disk operating system.
- CONVERT utility, on DBLDOS™ minidiskette, converts files and programs from single- to double-density or double- to single-density.
- **Plug-in installation:** No strapping. No trace cutting. Restore your Expansion Interface disk controller to original configuration by simply removing the DOUBLER™ and reinstalling the original disk controller chip.

PERCOM DISCOUNT COUPON
worth $20
toward
The Purchase of a
DOUBLER™
Coupon No. 80M103
Expires December 30, 1980
Void where prohibited by law.

- The DOUBLER™ circuit card includes high-performance data separator, write precompensation circuits for reliable disk read operations — even on 77-track drives.

Introductory price, including DBLDOS™ and format conversion utility on minidiskette, **only** $219.95. Use the coupon for even greater savings.

Call toll-free, 1-800-527-1592, for the address of your nearest dealer, or to order direct from Percom.

†Percom TFD-200™ drive, OS-80D™ operating system

PERCOM

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

PRICES AND SPECIFICATIONS SUBJECT TO CHANGE WITHOUT NOTICE.

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


ELECTRIC
CRAYON
SHARP
Linytron

80 microcomputing™ DATA

PUBLISHER/EDITOR
Wayne Green
MANAGING EDITOR
Michael Comendul
TECHNICAL ADVISOR
Jake Commander
PRODUCTION EDITOR
Clare McCarthy
NEWS EDITOR
Nancy Robertson
REVIEW EDITOR
Emily Gibbs
ASST. TECHNICAL EDITOR
Chris Brown
EDITORIAL ASSISTANTS
Debra Marshall
Thomas Peabody
EDITORIAL ADMINISTRATION
Cresca Clyne
Nancy Noyd
DESIGN ASSOCIATE
Diana Shonk
DIRECTOR OF MANUFACTURING
Noel Ray Self
ASST. DIRECTOR OF MANUFACTURING
Dion Owens
ADVERTISING PRODUCTION
Robert Drew, Bruce Hedin, John White
MAKE-UP
Michael Murphy, Steve Baldwin, Linda Drew, Kenneth Jackson, Ross Kenyon, Bob Sawyer, Patrice Scribner, Sue Symonds
PHOTOGRAPHY
William Heydolph, Terrie Anderson, Tedd Cluff
TYPESETTING
Barbara Latti, Sara Bedell, Linda Locke
PUBLISHER
Wayne Green
ASSOCIATE PUBLISHER
Edward Ferman
ASSISTANT PUBLISHER
Jeff DeTray
EXECUTIVE VICE PRESIDENT
Sherry Smythe
CORPORATE CONTROLLER
Alan Thulander
ADVERTISING MANAGER
Kevin Rushalko
CIRCULATION
Debra Boudrieau
EXECUTIVE ASSISTANT
Leatrice O'Neil
BULK SALES MANAGER
Ginny Boudrieau
ADVERTISING SALES
(603) 924-7138
Penny Brooks
John Gancarz

Manuscripts are welcome at *80 Microcomputing*, we will consider publication of any TRS-80 oriented material. Guidelines for budding authors are available, please send a self-addressed envelope and ask for "How to Write for *80 Microcomputing*."

80 Microcomputing (ISSN #0199-6789) is published monthly by 1001001 Inc., 80 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. Subscription rates in the U.S. are $18 for one year and $45 for three years. In Canada, $20—one year only, U.S. funds. Foreign subscriptions (surface mail), $28—one year only, U.S. funds. Foreign subscriptions (air mail), $60—one year only, U.S. funds. 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. Australian Distributor: Electronic Concepts, Rudi Hoess, 55 Clarence Street, Sidney 2000, Australia. All U.S. subscription correspondence should be addressed to *80 Microcomputing*, Subscription Department, P.O. Box 981, Farmingdale, NY 11737. Please include your address label with any correspondence. Postmaster: Send form #3579 to *80 Microcomputing*, Subscription Services, P.O. Box 981, Farmingdale, NY 11737.

← MORE PRODUCTS ON PAGE 4

MAKES EVERY BYTE COUNT

IN YOUR TRS-80* MODEL I OR MODEL II DISK SYSTEM

PROGRAMMING TOOLS

SAVE Any 3, $49.95
For Model II........... $74.95

TDAM $19.95
For Model II $29.95
Includes MTC QUE Card!

Having trouble with RANDOM FILES? With MTC's Table-Driven Access Method (TDAM) you'll never fret over FIELDing again. No knowledge of random access files is required. Insert the TDAM "interpreter" into any BASIC program and type in a few DATA statements describing the information in your files. TDAM does the rest! Reads and writes fields and records of any type (even compresses a DATE field into 3 bytes!). Features automatic file buffer allocation/deallocation, memory buffering, sub-record blocking/deblocking, and handles up to 255 fields per record. Super fast and super simple! Complete with TDAM interpreter, instructions and demo program. Requires programming experience.

DIVERGE $19.95
For Model II $29.95

Compares two BASIC program files, showing the differences between them. Identifies & lists lines which have been inserted, deleted, & replaced. Use for version control.

REBUILD $19.95
For Model II $29.95

Reorganize programs for adding program code, faster execution, readability. Much more than simple renumbering. Rearrange groups of statements within a program - automatically updates references to line numbers. Use with SUPERSEDE and MINGLE for maximum effect.

SIFTER...................... $19.95
For Model II.................. $29.95

Twelve in-memory high-speed sorts for use in any BASIC program: stable, non-stable, with/without tags, for numeric or string data. Random File Sort included. Some sorts written in machine code. Includes sort subroutines, demo programs and instructions. Relocate as needed with REBUILD. Requires programming experience.

SHRINK..................... $19.95
For Model II.................. $29.95

Makes Every Byte Count! Make programs smaller and faster! Combines lines & removes unnecessary code including remarks, without altering program operation. Typically reduces program size 25% to 40%.

SUPERSEDE $19.95
For Model II.................. $29.95

A "must have" for the professional programmer or the serious amateur. Probably one of the greatest time-savers available. Write programs in shorthand - change variable names - generate program documentation - use with REBUILD and MINGLE to build new programs from old ones.

MINGLE-II.................. $19.95
For Model II.................. $29.95

Merge up to 14 files (Program or Data) into a single file. Data files may be merged in ascending or descending sequence with the ordering based on a user-specified comparison field. A very handy utility for consolidating data files.

Single sided, Single density, Soft-sectored

DISKETTES

Verbatim 5¼-inch

$23.95 Box of 10

NEW LOW PRICE

Quantities of 10 boxes (each)........ $22.95
Hard-sectored (10-hole), Box of 10........ $26.95
8-inch FLOPPIES
Single-density, Box of 10.................. $29.95
Double-density, Box of 10.................. $39.95
PLASTIC LIBRARY CASES
5¼-inch or 8-inch diskette case............ $3.00
50 (5¼-inch) diskette file box...... $19.95

FACTORY FRESH, ABSOLUTELY FIRST QUALITY
Minimum order 1 box, NO order limit!

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

NEWDOS +

$69.95 by Apparat

40 TRACK VERSION $79.95

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

MTC QUE Card only.................. $1.50

Let Your TRS-80* Teach You

ASSEMBLY LANGUAGE DISK I/O TECHNIQUES

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

REMDISK-1 .... $29.95

Let your TRS-80* Teach You

ASSEMBLY LANGUAGE

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

REMASSEM-1 .... $69.95

The perfect supplement for your NEWDOS, from IJG, Inc.

"TRS-80 DISK AND OTHER MYSTERIES"

by Harvard C. Pennington

132 pages written in PLAIN ENGLISH packed with HOW TO information with details, examples and in-depth explanations. Recover lost files and directories, remove file protection, make BASIC programs unlistable. How to use SUPERZAP, recover from DOS errors and MORE!

TRS-80 DISK $19.95

All products guaranteed for replacement only. Prices, Specifications & Offerings subject to change without notice.

MOST ORDERS SHIPPED WITHIN ONE BUSINESS DAY

QUANTITY DISCOUNT INQUIRIES INVITED

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

- Add $2.50 for standard UPS shipping & handling
- $2.00 EXTRA for C.O.D.
- Ohio residents add 5½% sales tax.

TO ORDER CALL TOLL FREE
1-800-321-3552
IN OHIO call (216)289-7500 (COLLECT)

META TECHNOLOGIES CORPORATION
26111 Brush Avenue, Euclid, Ohio 44132

20

800730

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

80 REMARKS

by Wayne Green

"When Radio Shack announced the TRS-80, Ed must have been one of the first to appear at their door to get one."

Here Come de Juge

For many years, one of the top ham stores in the Southwest was Juge Electronics in Fort Worth. Thus, I've known Ed Juge for many years, as a ham and as an advertiser in my *73 Magazine*.

In mid-1975, my frustration with trying to understand and select a computer for handling magazine subscriptions led me to microcomputers. From this interest I developed a plan to put out a magazine for computer hobbyists. Thus was born the idea for *Byte* magazine.

Once I had located someone to edit the new magazine, there followed a frantic five-week period getting the first issue ready to print. Details such as designing and printing the letterhead and envelopes, composing subscription letters, writing possible authors, getting mailing lists from firms in the business, getting in touch with all of the clubs and their newsletter editors, etc., were almost endless.

Then, once the first issue was out, I grabbed a bunch of copies and flew to visit the major firms in the business to show them my new magazine and get their support. I visited Mits, the pioneer microcomputer firm, then Sphere. On my way down to San Antonio to see Southwest Tech, I stopped off in Dallas and drove over to Ft. Worth to say hello to Ed Juge. I talked with Ed enthusiastically about the future and what I saw as an industry which would have to grow very rapidly.

Apparently my enthusiasm did the trick, for Ed quickly sent away for an Altair and started in microcomputing. When Radio Shack announced the TRS-80, Ed must have been one of the first to appear at their door to get one. By this time he was out of the ham business.

The ham business was a real bummer during the 1964-1974 period, as a result of a proposal by the American Radio Relay League to change the ham regulations in a way that would take enormous frequency bands away from most hams. Sales of ham gear dropped to about 15 percent of their previous level and about 75 percent of the ham stores around the country quit the business. Not long after my visit, Ed got fed up with the slow ham sales and with most of the ham manufacturers going out of business (we lost Hallicrafters, National, Hammarlund, Barker & Williamson, Johnson, Harvey Wells, Multi-Elmac, World Radio, Galaxy—virtually every major ham manufacturer) and folded up his ham business. For a while he was manufacturing a fold-away mobile mount for car antennas and doing well. But the micro bug had bitten.

It didn't take very long for Radio Shack to discover this sharp chap right in their backyard. Eventually Radio Shack lured Ed out of retirement and made him director of computer merchandising. That was one of the wiser moves made by Radio Shack. Ed is one of the reasons that Radio Shack has been able to move a bit faster than most of the other firms in the field.

I'm happy to have Ed writing a column for *80* and thus give all of us TRS-80 owners a look at what is happening in that otherwise very secretive Tandy Tower in Forth Worth. It will be a relief to get some honest information on what is happening and what may be in the works. The general tendency of large firms is to deny all rumors up until a new product is ready for sale. This is frustrating, particularly in a fast moving field such as microcomputers.

Tandy in Europe

Just in case you are under the impression that Tandy is not pushing hard in Europe, here is the back cover of their recent catalog distributed in the Netherlands. That price translates to about $855 in U.S. funds at the current rates, not a lot more than the U.S. price, considering the problems of shipping, duty, service and other distribution costs.

We stopped to visit a major Tandy store in Rotterdam and sure enough, they had a TRS-80 on display. We talked with them about getting some of our TRS-80 programs translated into Dutch and they put us in touch with a local programmer who was most interested. I think we'll be coming out in one more language in short order, if anyone wants to brush up on his Dutch.

The Belgian price translates to about $866 at the current exchange rate, again a very reasonable price for the system when you figure all the expenses involved in supporting the system in a small country like that. The problems are somewhat simplified by the use of French in the catalog, but I suspect that most programs are still in English.

As we were driving through a very small town in Belgium, Malmedy, about the size of Peterborough, N.H. (population 5,000), Sherry spotted the Tandy store and we stopped.

We had a pleasant visit with the chap running the store and found that even in a town this small he was selling an average of two or so TRS-80 systems a month! We showed him some of the Instant Software packages we had

Continue to page 11

Tandy's Catalogs Distributed in the Netherlands and Belgium

INSIDE 80

by Ed Juge, director of computer merchandising, Tandy Radio Shack

"To those among you who didn't like TRS-80's modular design, and said we aren't responsive to our customers—may you byte your tongues."

Since I promised last month to tell you about some new products, you're expecting me to finally admit the TRS-90 rumor, aren't you? We heard so much about that here in Fort Worth that we *almost* used TRS-90 as a code name for one of our new computers.

TRS-80 Color Computer

I'm writing this prior to any announcements; bear this in mind when I say, we are about to announce our new TRS-80 Color Computer! The Color Computer attaches to any color TV—yours or ours. A video modulator is built-in, and radio frequency interference (RFI) shielding is excellent.

The screen is 16 lines deep by 32 uppercase characters across.

Color graphics are available in five formats from 32 times 64 pixels with eight colors, to 196 times 256 pixels with one color plus background. The Color Computer also offers sound.

The keyboard has 53 keys. Although they are smaller than those on professional keyboards, they are in the normal staggered typewriter layout. The processor is an MP 6809E.

The low-end Color Computer comes with 4K RAM and an 8K ROM BASIC. Two joystick ports and an RS-232-C are built-in. Although a recorder is optional, a cassette port is provided for program and data storage at 1,500 baud (not compatible with our non-color TRS-80 software). There is also a port for ROM plug-in Program Paks and GO programs, which can include up to 16K of additional ROM.

Expansion options will include up to 16K RAM, which is required for some of the higher resolution graphics modes, a 16K extended BASIC, the joysticks, communications, printers and a disk drive.

Everything won't be available immediately. For example, the extended BASIC is expected in late November, and the disk drive in early 1981. Joysticks should be available by press time. The existing Quick Printer II already has a serial interface, and its 32-character lines match the Color Computer's format.

TRS-80 Color BASIC is more powerful than Level I and compares favorably with Level II. It includes commands for input from the joysticks, for color video, and for sound output. PEEK, POKE and USR(0) are included for the serious programmer. Accuracy is to nine digits. String handling, arrays and functions (ABS, INT, RND, SGN, SIN, SQR) are also included.

Besides higher resolution in graphics, TRS-80 Extended Color BASIC adds rotating and zoom movement to your graphic blocks. You'll also get editing, tracing, specific error messages, user-definable keys, additional intrinsic functions and real-time clock access from BASIC.

I'll bet about now you're saying to yourself, "Just what the world needs—another thousand-dollar color computer!"

Not quite. The basic computer will sell for only $399. The 16K RAM kit is $119, and the Extended ROM for another $99 plus installation charges. In late November a 16K Extended Color Basic computer will be available.

Because the new color computer utilizes a different kind of chip in ROM, it requires its own software. Initially, eight ROM Program Paks will be available from Radio Shack: Personal Finance, Quasar Commander, Checkers, Chess, Music, Math Bingo, Pinball and a diagnostic pack to check memory and functions. Several more are being developed.

TRS-80 Color Video and Model III

The Model III

The smoke from our Color Computer did a fair job of covering up another new project. To those among you who didn't like TRS-80's modular design, and said we aren't responsive to our customers—may you byte your tongues.

There is now a bridge between the Model I and Model II. There is now a TRS-80 Model III!

A single housing contains a 12-inch high-resolution video monitor, 65-key professional keyboard with a 12-key datapad, computer and power supply (with only one power cord).

Model III will expand quite far without going outboard. You can start with the familiar 4K, Level I system at $699, which is software compatible with existing Level I cassette programs. (The recorder is not included.)

Additionally, Model III Level I includes print commands to take advantage of Model III's parallel port. The processor is a Z-80, at 2.0 MHz.

One kit can be used to upgrade Level I to 14K extended Model III BASIC and increase 4K RAM to 16K. From there, RAM increases in 16K jumps to the maximum of 48K—all internally.

Model III BASIC offers two cassette speeds: 500 baud which is compatible with existing Level II programs; and a new reliable 1,500 baud mode. An expanded character set including upper- and lowercase is standard, as is a screen print function in ROM. Also, the

BREAK key will now exit from a hung-up LPRINT or CLOAD—or any other lock-up.

There is a real-time clock, accessible from Model III BASIC or machine language. Also, an RS-232 board can be added inside the Model III.

Two internal disk drives can be added. The 5¼-inch drives are 40-track double-density drives, giving 178,944 bytes per data diskette (about 134K user space on drive 0). Two more drives can be added externally, for a total of four, and almost 600K of on-line storage. With two drives, single-density Model I disk software can be converted to Model III disks.

All Radio Shack software, and most non-Radio Shack software for the Model I should be compatible. Three Model III versions will be sold: the 4K Level I for $699, the 16K Model III BASIC for $999 and the 32K two-disk RS-232 system for $2495. Other system configurations must be reached by installing upgrade kits.

Model III TRSDOS will contain all of the features of the Model I and many of those of the Model II DOS. It includes a BUILD command to create a DO file for automatic power-up command sequences. HELP gives you detailed instructions on any specified command. SETCOM initializes the RS-232 port, and CONVERT reads a Model I disk and converts its programs to a Model III disk and Model III format.

We've gone overboard to maintain compatibility with Model I programs. We also kept Model I's 16-line by 64-character video format, in the interest of compatibility. There are strong arguments for others formats, but we feel the compatibility of existing software is most important. We expect some vocal objections, but hope our choice reflects the choice of the majority.

The Pocket Computer

This year at Radio Shack good things are coming in threes. Our third new product is the TRS-80 Pocket Computer! That's right—a pocket-sized TRS-80.

It programs in BASIC, has a 24-character liquid crystal scrolling display, features ten-digit numeric accuracy, with exponential notation to plus or minus the 99th power of ten. It includes 1.9K of memory for program and data storage, which stays live even with power switched off. Program steps are stored in a compressed format that packs more commands into available memory than you will believe!

An automatic power saver feature turns the power off after 7 minutes of inactivity in case you have forgotten. An optional cassette interface allows external program and data storage.

The keyboard contains 57 keys, including an alphabetic keyboard in typewriter format, numeric keys in calculator layout and special functions. Eighteen of the alphabetical keys can represent reserved functions when used with the SHIFT key. Four type 675 mercury batteries (non-rechargeable) provide a nominal life of 300 hours.

The Pocket Computer can be used like a calculator. Formulas are input using parentheses and symbols, as you would on paper, rather than one operation at a time in typical calculator mode. After an equation is solved, it can be recalled and edited.


TRS-80 Pocket Computer.

We have seven programs already available for the Pocket Computer—with more to come. Or you can write your own BASIC programs. The Pocket Computer contains 11K of ROM, including a 7K BASIC and 4K monitor.

Pocket Computer BASIC is much like Level I. But it includes 15 arithmetic functions and no graphics. It allows limited string variables (up to seven characters), and limited string and numeric array variables.

The memory, which holds 1424 steps, is automatically partitioned for program and data storage. There is a 26-data element memory and a 48-step reservable memory for storing frequently used functions. This six-ounce beauty is a true TRS-80—not just a programmable calculator with alpha capability. When you've tried one, I predict you'll give your programmable to "Junior" to play with!

Well, I hope I've excited you. Obviously Wayne doesn't want me taking up the whole magazine.

Let me close by saying, once again, that as Radio Shack brings out new products, we don't intend to forget purchasers of our first TRS-80s. We intend to continue to support all models with new software and hardware products. ■

80 REMARKS

From page 8

for his system and he felt that if he had them available to help him, his sales would increase dramatically. Tandy stores are not yet permitted to sell any products not stemming from Tandy. ■

Twentieth Anniversary Issue

With the twentieth anniversary issue of *73 Magazine* (my first publication in the ham radio field) going to press, I find myself reflecting over the ups and downs of the past twenty years.

My ham magazine, like *80*, is for me a labor of love more than a way of making money. Amateur radio is beginning to grow again and there are some exciting prospects. Many of the changes I see coming in radio are tied to microcomputers and the circuits which microelectronics make possible. For those of you who have not yet been bitten by the hamming bug, I might explain that most of the important inventions and pioneering developments in radio communications have been made by amateurs, and there is no reason why this won't continue.

The list of ham developments is a long one—slow scan television, virtually every radioteletype circuit being used today, circular antennas, etc.

The pioneers of commercial television were amateurs too. I worked in the TV broadcasting industry back in the early days (1948–1950) and many of the people I worked with were graduates of W2XEL, an amateur television station out in Queens, NY.

Today, we are in the early days of microcomputers, and we are seeing most of the inventing and pioneering again being done by amateurs. This is one of the exciting aspects of this magazine. It is a medium for keeping everyone in touch with the latest ideas and the state of the art. People who depend on reading books are a year or more behind what is happening because it takes that long before books are published. Meanwhile, the art is staying ahead of the books and is reflected in magazine articles.

The art, in our case, is microcomputing—its hardware and software. In the hardware line we see improvements in the basic computer unit (CPU), in memories, in mass storage equipment such as disks and various tape systems, bubbles, in printers, modems, control systems. In software we are seeing continuous improvements in languages, in operating systems, and in applications programs of all sorts.

The basic purpose of *80 Microcomputing* is to let you know what is going on in all of these aspects. We aim to make the material as understandable as possible, and we are making a major effort to see that we have plenty of information to help the rank beginner grasp the fundamentals. But we also want to act as a medium for the front runners to communicate and let all of us know what is going on up there. ■

80 INPUT

". . . if you have a new set of ROMs at power on. . . The computer will ask MEM SIZE instead of MEMORY SIZE."

Shack Service

I was amused by a recent local advertisement for Radio Shack Computers that said, "We service what we sell." As a dealer in business systems, we are naturally interested in that line of equipment and purchased a Model II, primarily to get acquainted with it and find out just what using and supporting it involved. We found out.

First of all, it was DOA from a bad memory chip. We broke the seal as part of our self-imposed indoctrination and luckily had a chip on hand from a previous job. The next failure involved the CRT sweep circuitry, and we dug into it again.

It turned out that the part we suspected (horizontal drive amplifier) had a house number that did not cross-reference to anything. It was then that we learned that Radio Shack stores are not permitted to stock parts for these computers, and that Tandy's standard advice is to send the entire machine to the nearest service center, roughly 600 miles away. That center verified that they were not authorized to service boards or sell parts.

Another call on the hot-line gained us the names of three other cities having service centers, but no names, addresses, or phone numbers. We did locate them through phone directory assistance, and eventually found one that would accept a board for repair.

The repair was prompt and well done, at a reasonable price. The fix, as nearly as we could determine, consisted of changing the value of several resistors. This apparently prevents instability when the drive is low, which was the case in our machine.

In summary, we can only tell our customers that yes, Radio Shack does have a service capability, but it is somewhat difficult and time-consuming to obtain when you need it.

E. G. Brooner
Lakeside, MT

Satisfied Subscriber

I am just writing to you to express my gratitude for your magazine. It is a truly superb publication, and I plan to subscribe to it for a long time to come. We get a few other TRS-80 related publications, but yours is the best.

Just the other day I received as a gift the Radio Shack Editor/Assembler and I have started learning assembly language.

I looked over my first issue of *80 Microcomputing* and was again amazed at the amount of useful information supplied to the reader. You wrote in that issue, ". . . The editor of *80 Microcomputing* is not trying to impress you with his great knowledge of computers, I just want to provide you with entertainment, a way to learn more about your system, a way to save money on developing it." I believe you have done just that. Please continue to publish the fantastic magazine that you do.

Brad J. Richter
Ridgefield, CT

Multiple INKEY Entry

Entering more than one digit or character by means of the regular INKEY routine is not possible, since the system reacts to the last character entered. Using the INPUT routine displays a query symbol and also requires ENTER to be pressed: In my opinion, this is for the birds in any reasonable operator/machine interface.

The following routine permits two characters to be INKEYed:

```
100 GOSUB 500
110 B$ = A$:A$ = "":PRINT B$;
120 GOSUB 500:PRINT A$
500 A$ = INKEY$:IF A$ = ""THEN 500 ELSE RETURN
```

Line 110 stores the first INKEY as B$, and resets A$ to a null. This requires that two characters *always* be entered, e.g. 01 for 1, etc. The routine can be expanded for additional input if required.

The input can be restricted to digits only by modifying line 500:

```
500 A$ = INKEY$:IF A$ = ""THEN 500 ELSE IF ASC(A$)<
    48 OR ASC(A$)>57 THEN 500 ELSE RETURN
```

Any non-digit entry will be ignored.

Michael Barlow
Pierrefonds, Quebec
Canada

New ROMS

Did you know that Radio Shack has new ROMs in the latest models of the TRS-80? In case you don't I will tell what little I know and problems and fixes I have found.

First, Radio Shack did a software fix for the cassette loading problems. It works wonderfully!! Second, they added a control key which is the shift, down arrow. Therefore, shift, down arrow G will send a CHR$(7) or what is known as a bell. This works for all 26 keys. However, if you are using NEWDOS the shift, down arrow no longer works for the last program line, but shift, down arrow Z will, for the old code for shift, down arrow was 26.

You can determine if you have a new set of ROMs at power on. The computer will ask MEM SIZE instead of MEMORY SIZE. The second message will use R/S for Radio Shack. The keyboard will also have the new keyboard which is a wonderful improvement in itself.

I hope this information may be helpful to you. Also page 0 calls are all the same for other calls I don't know. Radio Shack will only say the calls listed in the editor/assembler are in the same place.

Chess McKinney
Hermitage, TN

Scripsit: Round Three

Perhaps I should not say anything, not having tried the Electric Pencil WP program, but since I saved about $50.00 by buying Scripsit, I figure I have two cents I can spare to stick in here.

In his letter to *80*, Peter Brennan seemed somewhat incensed at Scripsit for a few of its shortcomings. Let me simply remind him in a blanket approach that Pencil was not written for the TRS-80. It is available for many machines and has probably undergone changes (for the better) as it metamorphosed.

More specifically, let me address myself to what he mentions as defects and see if we can't shed some light on them.

I am using Scripsit with a Teletype 43 until my 737 arrives. Mr. Brennan is correct, the program assumes the printer will supply a line feed with the carriage return. The only formatting problems I have run into using my 43 is that double spacing requires using the LS = 3 command rather than LS = 2.

Yes, it does play havoc with page length, but, and perhaps there is some other difference between the printers we use, I find that the page length setting should be full scale at 90, not the 75 he suggests, and the bottom margin somewhere around 86, again, not the 70 he mentions.

The paragraph formatting is simply set to the same dimension as the line space. Granted, one must then print the text using the (P)ause option, but I picked the printer, as I assume he did, and I knew full well what its limitations were in regard to its compatibility with my TRS-80 and the printers Radio Shack suggests for it.

I also have those sheets of paper Tandy claims will enable Scripsit to use "my own driver." I also have no idea of what to do with them. I think his condemnation of Tandy might have been modified to read, "You should have bought a Radio Shack compatible printer," and in this instance they are right. You don't spit in the wind and you don't pull the mask of the old Lone Ranger. That only makes sense.

I can't vouch for the ability of the Pencil to keep up with the pace of typing, but it's definitely not true that Scripsit will keep up with a fast typist. I am not the fastest, but there are times when the program will not keep up with me, and times, since it allows continuous character print by holding down a key, that I get multiple characters when I don't want them.

I have typed some 90 pages using Scripsit, which makes me no expert in its operation, but I think the bulk of Mr. Brennan's comments might be clarified by concentrating on using Scripsit and discovering its finer points, rather than trying to play it off against Pencil. I would have rather he mentioned that selecting a particular spot in the text for printing requires reentering the format instructions. That's an inconvenience, but that's about the only thing I can come up with that bothers me.

Scripsit is easy for me to use. After the first half hour of the tutorial cassette Tandy supplies I shut my deck off and started using Scripsit with the foldout of operating commands in front of me. I guess the decision as to which program depends on the matter of application, of use and of oneself.

William O'Brien
New York, NY

Micro Mystery

I am returning the Microsoft editor/macroassembler/linkage-editor to my Radio Shack dealer because of totally inadequate documentation.

There is absolutely no discussion of concepts. How is the average Level II BASIC midnight coder supposed to know the difference between an assembler, a compiler, and an interpreter? How is he/she supposed to know what a macro is? How is he/she supposed to know the power and flexibility of a linkage-editor?

Being a multi-lingual independent commercial programmer, I have worked on most 3rd generation computers, including everything made by IBM. I am used to IBM manuals that deal with concepts prior to discussing bit-fiddling. Microsoft people have obviously never seen an IBM manual, which has wonderful litle aids like examples, concept discussions, sample programs, graphs, hints, references to other manuals, etc., all of which are totally lacking in Microsoft's piece of literature which I am hesitant to call a manual.

User response prompting is non-existent. When the editor, assembler, or linkage editor are executed, they simply display an asterisk on the screen. If I wrote programs like that in industry, I'd be fired on the spot. How about prompting the user to enter the source file name, the object file name, the load address, the number of object modules to be linked, do I want a cross reference, etc., etc., etc. I'm totally underwhelmed by their kindergarten code.

Can this manual really be from the same folks who gave us the Level I and Level II (w/o index) manuals? Why didn't Microsoft contract with David Lien to write their assembler manual instead of the incompetent trainees they used?

Dave Westfall, Pres.
Aries Computer Software, Inc.
Farmington Hills, MI

You are right, of course. I would perhaps be more critical of Microsoft if I had had any success in the last four years in getting an article written for any of my magazines which covered the material you are asking about. I have been promised dozens of times by writers that they would do this. . . and so far not one promise has been kept.

The difference between an interpreter and a compiler is a very difficult concept for both beginners and intermediate computerists.

This applies to macro anythings, linked editors, and the lot. With the execution of the instructions which come with the Level I TRS-80, I would be hard put to point with pride to good documentation in our field. I can point to any number of ghastly disasters.—Wayne.

Tired Memory

If you're tired of continually having to reserve memory for your commonly used machine language routines, or if you have more than one and have to figure out how much to reserve, you can let the 80 do all that for you by using the method shown in the listing below.

```
50 CLS
200 DEFINTA,I
300 DIMA(8)
500 FORI=0TO7
550 READA(I)
700 POKEVARPTR(A(0))+1,A(I)
800 NEXT
1000 DATA62,49,50,32,62,195,154,10
1200 DEFUSR=VARPTR(A(0)):X=USR(0)
1400 A$=INKEY$:IFA$=""THEN1400ELSESTOP
```

Simply answer the MEMORY SIZE? with an <ENTER> and append the appropriate data statements to your program. The data in the program (line 1000) prints a 1 in the center of the screen.

The key to the program is in line 700 where the data in line 1000 is POKEd into memory, starting at the location where the array A begins. By setting the DEFUSR to that starting location and executing the USR, the 80 will perform the machine language steps and return to your BASIC program. For your routines, just replace the data statement with your data and change line 500 accordingly. Line 1400 was inserted so you can admire your handy work. Note that you can have several machine language routines.

A word of caution is needed. Since the 80 has a habit of rearranging its memory from time to time, the starting location of the A array may change. For this reason, it is necessary to reset DEFUSR each time the USR is called. I would recommend carrying statement 1200 as a compound statement any time you need your machine language routine.

A. W. Maddox
Creve Coeur, MO

Simple Debounce

In the April issue of the magazine, you have an article written by C. W. Anderson which shows a relocation routine for the KBFIX routine. In this article the author makes a request for a debounce routine which only contains relative jump instructions. For some months now I have been using a debounce routine that can be directly located anywhere in memory, but which normally will not require relocation.

In my TRS-80 which does not have an expansion interface, the reserved RAM area from 405E to 407F does not seem to be used by any of the routines in ROM. Thus if a routine can be fitted into this area, then it can remain untouched during execution and the addresses in the high end of RAM can be used for other purposes. Since KBFIX apparently requires 55 bytes it won't fit in here, but there are fortunately less memory-consuming methods available. The method I am using requires 22 bytes of permanently allocated RAM, and will therefore easily fit into this area.

The extra instructions required to initialize the routine are only used once, and can therefore be placed in an area of RAM where they may be erased at some later stage. The most convenient area to use for this purpose is the I/O buffer area from 41E6 to 42E6. As long as the buffer is not used during the initialization process, the area is perfectly safe to use for such purposes.

The debounce routine may seem a little primitive, but during several months of use it has not let me down once. During normal processing it is even slightly faster in execution than the routine in ROM since it does not require as many instructions to determine whether the keyboard status has changed. When the status does change, however, a delay loop is invoked causing a delay in processing.

I can add that when I use the TRS-80 without the routine, there will be a repeated character on nearly every line, despite clean keyboard contacts.

The 22 bytes of permanent code are as follows:

```
406A  118038          LD DE,3880H
406D  213540          LD HL,4035H
4070          NXTCH:
4070  CB03            RLC E
4072  F8              RET M
4073  2C              INC L
4074  1A              LD A,(DE)
4075  AE              XOR (HL)
4076  28F8            JR Z,NXTCH
4078  0605            LD B,5
407A  CD6000          CALL 0060H
407D  C3E303          JP 03E3H
```

The value loaded into B in line 4078 could possibly be changed to 4, 3 or 2 thus reducing the delay time, if this still provides adequate debounce.

The only purpose of the initialization routine is to put the starting address of the debounce routine into the keyboard DCB. Unfortunately this cannot be done in BASIC with POKE since the address is in constant use during execution and should therefore be done in an uninterrupted sequence of instructions or in a single instruction. The following lines can be used, at the same time setting the USR address back to the default value.

```
42D0  216A40   LD HL,406AH
42D3  221640   LD (4016H),HL
42D6  214A2E   LD HL,2E4AH
42D9  228E40   LD (408EH),HL
42DC  C9       RET
```

I use a short BASIC program to POKE these values into memory, call them with a USR(0), and then issue a NEW statement. I was also pleased to see in the April issue that others have found out that BASIC can be used to convert values from hexadecimal to decimal for the POKE statement (MACROPOKE monitor). It certainly saves a lot of conversion work and potential errors. Actually I use a format which includes the address at the beginning of each line, with the data bytes following, for example:

```
DATA 42D0,216A40 221640 214A2E 228E40 C9
```

Intervening spaces can be used to separate the individual instructions, making them even easier to interpret visually.

Obviously, since I have neither expansion interface nor a disk system, I cannot guarantee that the area used for the debounce is also unused when these are added, but for users without these extras (presumably still the majority) it should provide a trouble-free debounce that doesn't require MEMORY SIZE or interfere with other machine language routines.

Arne Rohde
Pilevej 31
7600 Struer
Denmark

Box Confusion

Thought your readers would like to know changes necessary to make "Ball Box" (April *80*) work in Level II, since they are elusive and not covered in text.

1) PRINT AT in Level I can be followed by a comma or semicolon. PRINT@ in Level II must be followed by a comma, NOT a semi-colon. Therefore, add a comma before the semi-colon in lines 2010, 4900, 4920 & 4940.

2) POINT returns −1 in Level II, *not* 1 as in Level I. Therefore, add a minus sign before 1 in lines 4320, 4330, 4410, 4420, 5020 & 5120.

3) No graphics wraparound in Level II, so change line 5001 to Y = RND(48 − 4) + 2 or you will get a FC (function call) error, line 5020 frequently (whenever Y>47).

Fred Blechman
Canoga Park, CA

Like Father, Like Son

I am 13 years old. I think that *80 Microcomputing* is one of the best books about microcomputers. My dad subscribes to it, and we both enjoy it.

I like the articles and columns that you have printed. It has very interesting stories and programs, we both read and like this book very much.

Thank you for making such a fine product.

Martin Eisenhauer
Margate, FL

DEBUg

Error in SCRNPRNT

A small error in my SCRNPRNT program in the May issue of *80 Microcomputing* got past me and effectively disabled the instruction that fools the computer into thinking that its RAM ends just below where SCRNPRNT starts. When working properly, this instruction both protects the program against TRSDOS Disk BASIC's usage of the top few bytes of memory and obviates the need for answering MEMORY SIZE. Unfortunately, I left off the H in the instruction, so the program attempts to change a ROM decimal address instead of the correct RAM hex address. This would not have caused a problem in the program as printed, since it was assembled lower in RAM, but would cause a conflict with BASIC if the program were assembled at the very top of memory.

To correct the error, change the fourth through sixth bytes of the assembled program to 224940 instead of 22D10F, and change the corresponding instruction to read LD (4049H),HL.

My apologies to anyone who may have been inconvenienced.

Louise H. Frankenberg
Pasadena, MD

More Story

The addendum at the end of the text, "Get the Whole Story" (July *80*) is not clear. The programs printed in the article are current, except for the Order Verification Program (Listing 1). This program requires frequent price updates. The cassette/wafer offered for $25 includes the latest high-speed order verification with the prices in data in effect at the time of shipment. Those who order the whole package for $25 may receive updates of the order verification program on cassette or wafer for $10.

Program Listing 3, Bookkeeper, has an omission that will create a serious error. Line 340 should have.......:J1 = 0...... added.

I'm also including a suggested correction to the monthly Gross Profit Statement (Listing 4), which will allow a much simpler method of providing a printer option.

With over 160 magazine articles published, I've never had such enthusiastic reader response as I've had to this one!

Fred Blechman
Canoga Park, CA

```
CHANGE LINE 10 TO......CLS:PRINT:INPUT"PRINTER? (Y/N)":A$
ADD LINE 11............IF A$="N" POKE16423,4
ADD LINE 12............CLS
CHANGE END OF LINE 31 TO......GOTO99
DELETE LINES...................50,60,70,75,86,90,92,93

IF YOU ANSWER  N  TO THE QUESTION IN LINE 10, THE COMPUTER
WILL IGNORE ALL LPRINT STATEMENTS, AND YOU'LL HAVE SCREEN
DISPLAY IN THE ORDINARY WAY. TO ENABLE LPRINT AGAIN, JUST
POKE16423,5.       NOTE: IF YOUR PRINTER NEEDS A SOFTWARE
DRIVER, PRINT PEEK(16423) AND NOTE THIS VALUE WITH THE
PRINTER DRIVER ACTIVE. THIS WILL THEN BE YOUR POKE VALUE
TO ENABLE PRINTER AFTER DISABLING WITH POKE16423,4 .......
```

Monthly Gross Profit Update

Disgruntled Manufacturer

There might have been some confusion about the thrust of John Acres' article, "Saving Money," that appeared in the July issue of 80. The piece raised havoc at Microtek in San Diego.

Acres' piece in no way purported to be a review of Microtek's product or of the Percom drive mentioned. It seemed more of a caution to enthusiastic hobbyists not to let their affections for the hobby elude their better judgement. The Moral: Always read the manual.

Though 80 does not intend to shy away from controversy, we regret the lost energies on an article that apparently was misconstrued by some readers. The intent of the Acres article appears to our editors to satirize his own electronic abilities or lack thereof.

To help set the record straight, we're reprinting the original correspondence from Acres to Microtek, forwarded to us by Dan Obed, Microtek's president.—Eds.

Microtek:

Enclosed are the remains of a once-proud Microtek printer that I killed by my own hand. Bad as that may sound, I am also responsible for the death of two Radio Shack disk drives, one Percom TFD-200 disk drive, a TRS-80 expansion interface and—perish the thought—a TRS-80 keyboard unit.

It was not cold-blooded murder though, it was an accident I swear! In my haste to see the Microtek unit in action, I used a Radio Shack number 26-1401 printer interface cable—the one normally used with the Centronics 779-2.

How was I to know that the Microtek has a separate signal and chassis ground (without having read the manual)?

Unfortunately, the interface cable mentioned above connects pin 17 to the signal ground of the TRS-80 expansion interface while deviously allowing the printer to function, giving no indication of the danger lurking ahead.

Although my intentions were good, I was overcome with the lust of a working system and—in a moment of insanity—disconnected the three-prong grounded ac plug of the TFD-200 from ac power. Somehow in that mysterious electronic twilight zone when an ac plug is somewhere between connected and disconnected, a ground loop (or something!) got onto signal ground and caused 5 ICs in the TFD-200 to explode and several pc traces to vaporize.

Similar damage was done to the TRS-80 components. As for the Microtek printer—alas—it too had its luscious circuitry violated. The pc trace to pin 17 of the parallel interface connector evaporated. And that's not all. The 8155 seems to be very ill. When connected, it pulls the +5 volts to a helpless 3.2. Whatever ails it is contagious. When a known healthy 8155 is plugged in, it too soon develops identical symptoms. Without the squiggly diagrams outlining circuitry connections, further surgery on my part is fruitless.

I appeal to you then: Ye mellow, laid-back gods of California heal this printer and return it to me as quickly as possible. Charge me what you will—but do not make it over a hundred dollars without talking to me first.

If they have not been sent already, please send along a ribbon and spools. They were not included with the printer originally.

John Acres
New Palestine, IN

80 AID

Teletype Interface

I have an old Model-15 Teletype Printer (with a 5-bit code and 60mA current loop) that I would like to use with my Radio Shack TRS-80 Model II. Unfortunately, Radio Shack has nothing that could or would help me concerning this type of operation. Please! Can you help me?

Milton J. Belle
2008 Wayside Street
Compton, CA 90222

Friden Schematic

Could you please print the following inquiry wherever it is appropriate?

I am looking for a schematic or information on the Friden Flexowriter Programatic Typewriter. I have a model FPC-5P which has a 5 level baudot paper tape punch/read assembly. I would like to interface this for computer or RTTY.

Robert G. Gilman
Box 103
Hellertown, PA 18055

Cassette Confusion

I have a problem with which I hope you, your staff or *80*'s readers could help me. I have a copy of the excellent tape Micro Music. Unfortunately, it is very annoying to have to pull the cassette plug out, plug it into a speaker, then plug it into the cassette player to load another piece of music.

Since I own an expansion interface, I know that you can have two cassette recorders attached, and if you store one in address 37E4H all I/O will be done on cassette two. This works great with Babybeep (*80*, April, 1980), but it will not work with Micro Music.

Since I know little Assembly Language I can't change this. I would like to be able to load the data on cassette one and play it out of cassette two. Can this be done? If so, I would like to know about it.

Also, if anyone knows how to put any Radio Shack machine language tape on disk, I would appreciate it if he would drop me a line.

Tarus P. Balog
Asheboro, NC

Morse Code Mod

My Morse Code program in your July 1980 issue is correct for TRS-80s without the Radio Shack E-Z cassette load modification. This modification is installed in all TRS-80s sold after early '79 and was made available for no-cost installation in mid '79 to all earlier models. If you have this modification installed, change the following eight lines to the Receive Mode section to read:

```
1290 IFA<200.....
1310 IFA>200.....
1330 IFA<200.....
1360 IFA>200.....
1380 IFA>200.....
1410 IFA>200.....
1430 IFA>200.....
1470 IFA>200.....
```

A special thank you to Gene Steele K5EVE, in Orangefield, Texas for this suggestion. It has been mailed out to all the 2000+ purchasers of the W4UCH TRS-80 Morse Program and should prove useful to your readers that have the E-Z cassette load modification installed.

Also, users should be reminded that a minimum of one volt peak-to-peak audio is required during receive without any RFI (radio frequency interference) going into the cassette input line from the station receiver, as the program obviously cannot discriminate between a valid Morse code signal and interference from the TRS-80. Good quality coax cable and a remote antenna at least 60 feet away from the TRS-80 is an absolute must.

Robert M. Richardson
Chautaqua, NY

Input to Page 40

LOOKING FOR TRS~80* BUSINESS SOFTWARE? WE HAVE HUNDREDS OF QUALITY BUSINESS PROGRAMS IN STOCK! AT PRICES YOU CAN AFFORD.

‹WHERE YOUR TRS~80* MEANS BUSINESS›

For the first time you can fill most of your software needs with one telephone call. Whether you are trying to find a specific program, custom software or just help with your system—give us a call. We have the people that count!

Invoicing • Inventory Control • Accounts Payable • Accounts Receivable • Payroll • General Ledger • Letter Writer • Word Processing • Mailing • Manufacturing Inventory • Cost Accounting • SalesReporting • Stock Market • Business Statistics • Statistical Analysis • Data Base Systems • Medical Billing • Dental Billing • Special Industries • Advanced Accounting • Income Tax • Language • Personal Finance • Technical Programs • Insurance • CPA • Law Office • Asset Depreciation • Job Cost • Utility Programs • Education • Games • Home Programs • Loans • Credit Bureau • Electronics • Test Systems • Sports • Art • DOS Systems • BASIC lessons • and hundreds more!

Send for our free catalog or give us a call today. We also do custom programs as well as buy top quality programs.

Summer Special:

Complete business system $299.95

80 REVIEWS

edited by Emily A. Gibbs

"He assumes the reader has a fairly good understanding of Level II BASIC, although all his tests. . . operate on a 4K machine."

TRS-80 Interfacing
Johnathan Titus
Blacksburg Continuing Education Series
Howard W. Sams & Co., Inc.
$8.95

by Ed Neister

In *TRS-80 Interfacing*, Titus provides the reader with a practical knowledge of input/output circuitry and its operations. It is a basic book that should be in the library of all hardware designers who wish to work with the TRS-80 system for input/output control.

The book is written for a general reader. Titus does not go too deeply into the basics. He assumes the reader has a fairly good understanding of Level II BASIC, although all his tests and experiments are designed to operate on a Level II 4K machine with read/write memory and data storage capability.

Operation Control

Titus introduces the reader to the control signals one by one and helps the reader grasp the purpose for this control system and how to use it. He then goes into the operational control of these signals by using Level II instructions in computer programming steps. Titus leads the reader from the initial concept to a point where the reader can write the programs and generate the control signals he desires. Titus and the Blacksburg Group have developed a circuit which allows the reader to do all the experiments presented in the book.

Command of the Language

In the first chapter, Titus discusses the Z-80 microprocessor, its memory capability and the setup and use of the diode devices used through the Z-80 processor. The information is presented quickly and in great detail.

In Chapter one, Dr. Titus talks about the simple diode commands, as well as how software commands and interface circuits are used. He briefly discusses the PEEK and POKE commands in relation to interface circuits. He also goes over assembly language vs BASIC programming, as far as data transfer and control are concerned.

TRS-80 interfacing and its hardware are presented in more detail in Chapter two. Diode device addressing and decoding, as well as how gates are used for addressing and decoding are discussed.

Chapter three contains a detailed discussion of the output and input ports, device interfacing and memory mapping. Chapter four focuses on flags. The input/output device synchronization, logical operations, software detection of flags and complex circuit flags are considered.

Readers are introduced to the breadboard circuit design in Chapter five. This allows testing of the logic probe device, memory coder bus buffers and control circuitry. Next Titus considers circuit construction.

Interface Experiments

Chapter six is the central theme of the book and delves into detailed discussion of interface experiments. After an introduction, eighteen experiments are presented to help the reader test and understand the concepts of input/output transfer. The experiments reinforce the concepts presented throughout the book.

For the size of the book, much information is presented. Several appendices are given. One presents logic functions concerning lamp monitors, logic switches and pulses. Another gives the parts required for the experiments. The last three appendices give the Z-80 microprocessor, the TRS-80 interface breadboard parts and PC artwork, so that the reader can easily build his experimental circuit.

The first eleven experiments provide a set of interfacing and programming investigations for readers interested in interfacing concepts. The last seven experiments provide additional lab investigations of advanced topics and projects to supplement the experiments. They cover logic probe, device address decoding, device selecting pulses, constructing an input port, handling multi-byte input port information and input port applications, constructing an output port, I/O interactions, data logging and display.

An analog-to-digital converter is maneuvered by the computer in Experiment 11. When the program is run, the operator should be able to make the meter needle swing between two preset voltages.

Experiments 12 through 18 get more involved with device address and decoder circuits, output ports, BCD and binary coding, using the output ports and a simple program as a traffic light controller circuit.

Experiment 15 is an important one, used to demonstrate how logic device testing can be done. Simple flag circuits are handled in Experiment 16. Experiments 17 and 18 are more detailed in working with programmable interface chips and interfacing the A/D converter to the TRS-80.

Experiment 18 is very interesting from the standpoint that the resistance capacitance (RC) curve can be observed on the TRS-80. An RC curve is typically used to measure pressure, temperature, speed, conductance or many other physical perimeters.

The reader can obtain an overview of the subject in a quick reading. Next the breadboard circuit can be assembled, and the TRS-80 should be put to use.

The most dramatic thing about the book is that it gives the reader practical experience and guidance with the TRS-80 with program examples and actual hardware. When the reader has completed the book, he should have a good understanding of I/O devices and interfacing. ■

Computer Games for Businesses, Schools and Homes
J. Victor Nahigian and William S. Hodges
Winthrop Publishers, Inc.
Cambridge, MA
Softcover, 157 pages
$10.95

by Len Gorney

Are you looking for a good computer games book that will give you a solid core of coding from which to develop your own personalized computer games? Do you welcome the challenge of converting a program from any version of BASIC to your own machine's BASIC?

Do you enjoy debugging others people's programs? Do you like to play computer games just for the fun of it? If you answer yes to any or all of the preceding questions, you are a prime candidate to purchase *Computer Games for Businesses, Schools and Homes.*

In his foreword, Mr. Weinberg introduces *Computer Games* as Winthrop's "first sally into new territory—the personal computer." The programs contained in this 157 page book are for the personal computerist. However, the particular version of BASIC, which runs on PDP type machines, used in these programs is not within the grasp of the average personal computerist. Assuming you don't have a DEC-type BASIC or DEC machine, there are a few changes you must make before you power up.

The backslash is used as the multiple statement per line delimiter in these programs. You will have to replace it with whatever character your BASIC uses as a separator, or code each statement on a separate line. You may have to change the PRINT statements to LPRINTs, if you want a printout. Using INPUT$ or INPUT in place of GET and PUT statements will eliminate some problems in your INPUT and PRINT statements.

A listing of the variable names and their uses within the programs follows a short description and history of each game. This is a nice touch, especially if you are the type who likes to tear apart a program. A typical sample run and a program listing in an easy to read 5 × 7 dot matrix round out the games' introductions.

More Winners Than Losers

As with any collection, some of the games are losers. "Bullet" is the old—click, click—computerized version of Russian Roulette. "Poem" randomly generates one, but with little literary value.

"Psycho" is a minor league attempt to output your personality profile based on a design. Other poor selections include a Star Trek game, "Taplab" and "Tic-Tac-Toe."

What remains, however, are 21 winners that can be divided into five general categories: calculation games, casino games, dice games, grid/math games, and the ever present miscellaneous category.

One of the more interesting calculation games is called "Date" which, when given today's date and any other date, calculates the number of days between today's date and the second date. A variation of this same logic is used in the next two programs.

"Biorythm" is a graph of those critical days! "Compat" is advertised as a biorythmic compatibility calculation for two people. Interestingly enough, for two people to be 100 percent compatible, they should either be born on the same day or their ages should differ by 59.86 years.

Casino games are the downfall of most personal computerists. *Computer Games* has "Blackjack", "Draw Poker" and "Jackpot", but done rather well. In addition to a fairly good description of each game, the authors have included probability tables which for "Blackjack", give the combinations and odds for hitting any particular count; for "Draw Poker", give the odds for being dealt each type of poker hand; and, for "Jackpot", a payoff table listing the odds for spinning a winner.

There are three interesting dice games, "Kismer", "Notone" and "Rollon"—if you go for the computer tossing dice, that is.

The grid/math/number games include "Boggle", a number-based Mastermind; "Escape", a grid game which pits you against five enemy attackers on a simulated battlefield; "Golf", a computer simulation; "Gunner", the old artillery game which teaches you a bit about trigonometry; "Skeedoodle", a math game; "Stock", a good stock market simulation.

Luck and Skill

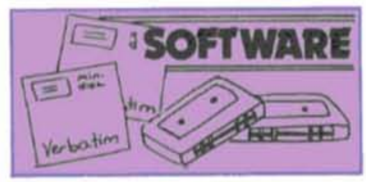
The remaining four games require an equal amount of luck and skill. "Basket" is a 2-player basketball game. "Dogs" simulates a dog race for up to 13 bettors. This program even skims off 17 percent of the wager pool before it pays the win, place, show and perfecta wagers!

"Horse" pits up to 10 players, each having 12 horses. The object is to get one of your horses over the finish line before any of your opponents. "Tennis", anyone? Input your position, the type of shot, and the area your shot should go to on the opponent's court and settle back.

The book is a good buy for anyone willing to accept the challenge of converting and debugging programs to expand his or her game library. It combines a learning experience with pleasure, and this, according to Weinberg, is the goal of *Computer Games.* ■

Planetary Lander
Instant Software, Inc.
Peterborough, N.H.
$7.95

by John Warren



Space games are one of the most durable commodities of the microcomputer boom, and spaceship landing programs are a mainstay of the genre. One of the best of these is Planetary Lander by Instant Software, Inc.

While the early lander programs were limited to number displays, graphics are now in. Planetary Lander shows the position of your ship in relation to an altitude "yardstick," the scale of which changes as your ship approaches the planet's surface.

Next to the altitude display is a control readout that gives lapsed time, speed in meters per second, thrust and other indicators. Two of the readouts use horizontal bar graphs that give speed as a percent of terminal velocity and the amount of fuel remaining. These displays are updated at one second intervals.

Instead of limiting itself to a single planet, Planetary Lander allows the player to attempt a landing on any of the nine planets (and the moon) in the solar system.

Both the thrust and its duration are input as a single statement eliminating the need to repeatedly enter identical inputs—an annoying feature of some other lander games.

Although Planetary Lander is only available in Level I, Radio Shack's conversion program transforms it to Level II by altering only two PRINT AT statements (lines 300 and 330).

Combined with Stellar Wars, a space warfare game, in a two program package (Star Trek III), Planetary Lander is an outstanding value. ■

Interlude: The Ultimate Experience
Syntonic Software Corp.
Houston, TX
$14.95 tape, $17.95 disk

by Chris Brown
***80* Staff**

Silicon pundits have told us for a long time that computers will play a major role in our changing lives. We will see the advent of electronic mail, electronic money, even electronic crime. It seems to me that if the devilish device is ever to be mainlined into our cultural envelope, it must service more basic needs than game playing and accounts payable. Something really basic, for example, like sex.

The time has come. The era of home computer sex is finally at hand, brought to you by Syntonic Software, a forward-looking subsidiary of Software Technology, Inc. These people are not above pandering to your prurient instincts for a buck, or fifteen bucks, actually.

Billed as the ultimate experience, the Interlude package is a curious piece of software for several reasons, not the least of which is the unmitigated gall the folks at Syntonic were able to muster to market this X-rated program in the first place.

What exactly is Interlude? Well, it isn't hard core, high resolution graphics, so if you're into blue movies, forget it. You won't get that sort of display with Interlude.

In essence, Interlude is a well-structured interview designed to plumb the depths of your sexual preferences in ten probing questions, after which it refers you to a book of "interludes" that accompanies the software.

The paper-bound, 96-page volume of interludes contains 106 elaborate sexual scenarios, written with acute attention to detail. Titles indicate content. For instance, "Wet Fun On A Hot Summer Night," "Good Vibrations," "Fellatio By Firelight," "Caveman," "My Way," "Satin Chains," "Macho Man." The list goes on.

If the computer-suggested interlude suits your fancy (or fantasy), you may opt to act out the scenario. In addition, certain interludes are stored in the computer and do not appear in the book. Getting one of these gems is apparently cause for self-congratulation. Your answers must have been outstanding!

This computerized extension of spin the bottle is not a revolutionary premise but is, one has to admit, an unusual application for the home computer and certainly not what the gang at Tandy had in mind.

The software algorithm of this machine language program is straightforward. The interview format is logical, and all questions require a numerical answer, indicative of the degree of enthusiasm the respondent demonstrates. Some of the more mild examples are:

How complicated do you want this interlude to be?

1. Back to basics.
2. Keep it fairly simple.
3. I'm not choosy.
4. Maybe a small production.
5. I'm ready for a really big show!

—or—

If you were a movie, which would you be?

1. Pillow Talk
2. Tom Jones
3. Gone with the Wind
4. Superman
5. A Streetcar Named Desire
6. Fellini's Satyricon

After answering several of these thought-provoking questions, the computer gets down to the nitty-gritty and quizzes you on specifics, hopefully picking up on your mores, folkways and taboos. Your scalar answers are then processed, and an interlude indicative of your final tally is suggested.

The program is fairly interactive. Suzy, one of our foxiest secretaries here at *80 Micro*, displayed so little enthusiasm during one interview session that she was directed to interlude number 29. The scenario for this interlude instructed her to stay home alone and curl up with a good book.

If the interlude suggested is not quite what you had in mind (e.g., an acrophobic being assigned an interlude involving a high-flying swing), you may request an alternate interlude.

You may choose from about six alternates before the computer assumes you are a spoiled brat and gives up.

The disk-based version of Interlude we've been using even bombs out at this point, painting the screen with 2-byte stack symbols as the video memory fills. Although this seems like a bug to me, it does get the point across. Syntonic uses their own DOS boot, which may still contain a few bugs related to program termination.

For all its daring, the Interlude package is fairly tame in its attitude toward sexual encounters. The maximum number of participants in any interlude is limited to two, so no free-swinging, group gropes are possible. Furthermore, all interludes are strictly heterosexual. No gay or even bisexual interludes are allowed.

Syntonic hints that this is just the beginning. The last page of the book suggests that Interlude II is coming and requests that users send their favorite interlude to Syntonic for inclusion in the next edition. No mention of royalties is made, but the concept is certainly democratic. We may even feature the Interlude of the Month in *80 Microcomputing* someday.

Syntonic's marketing has caused raised eyebrows in the industry, and their full-page living color ads have elicited both pro and con responses from readers. A few irate subscribers have taken offense at the quasi-naked lady approach Syntonic uses in their promotions and canceled. Generally, most readers seem curious or amused, rather than indignant. This may be just another indication of the decadence of the electronic age.

There you have it—Interlude: The Ultimate Experience. The first microcomputer-based adult computer program. Will it be a milestone in microcomputer software, or just another trendy diversion? Time will tell.

What's next? A microprocessor-controlled, RS-232-compatible love doll, perhaps. Welcome to the micro-millenia. ■

ISAM
Johnson Associates
Redding, CA
$50.00

by William L. Colsher

ISAM (Index Sequential Access Method) is a set of BASIC subroutines and utilities that allow the programmer to create and maintain direct access files using keys, rather than the hardware-dependent numbers used by Radio Shack. For example, a program can access a library catalog by the title of a book, rather than by some arbitrary number. ISAM requires 32K and is available for the Model I or Model II.

The ISAM routines which must be merged into the application require about 4.5K. There are also minor restrictions on variable names used by ISAM and, of course, the line numbers used by the routines.

Efficient Disks

In return for these restrictions, ISAM pays off with greatly increased disk space efficiency. The Radio Shack random access file method wastes a large portion of space if your file has only a few records in it. If, for example, the file contains records numbered 1 and 100, you have actually grabbed space for 100 records, even though the file contains only two!

ISAM files with two records take up space for only two records.

In addition to BASIC subroutines, several utilities are provided on the distribution disk. INIT is used to create new ISAM files, REORG is used to free space taken up by deleted records and sorts the file in key sequence, and ISAMPRNT enables you to list the entire file in key sequence on the printer or screen.

Most companies are content to simply provide the routines and utilities, but Johnson has included a demonstration program—a mailing list system. It may not be the most terrific mailing list program in the world, but it is more than adequate for my personal use and, I suspect, for some small businesses.

Suitability

ISAM does everything it promises very well. Anyone who has suffered the agony of trying to fit an existing numbering or naming system into Radio Shack's random access method, with everything but a shoehorn, should look into this package.

Finally, one of its best features is Johnson's delivery. I got my program six days from the day I mailed my check. ■

Z80ZAP/CMD
Org-Tex Industries
Lewisville, TX
$29.95

by Bill Vick

Z80ZAP/CMD makes lost file recovery, file patches and file maintenance a real joy, primarily because of its Hash Index. Z80ZAP calculates the hash code for any filespec instantly.

The Z80ZAP/CMD diskette comes with Backup on auto when it is booted up. This is a subtle way of telling the user to make a copy before he experiments with it. The diskette also contains a write protect tab, apparently one of those "protect us from ourselves" safeguards.

Automatic or Self-prompting

The documentation is contained in four brief, but comprehensive pages. Most of the commands are either automatic or self-prompting, making Z80ZAP a real breeze to use. Written in machine language, the run time is almost instantaneous in all modes, including power up.

When you load the program it waits two seconds for you to enter a command.

In the R (read) mode, having entered your drive, track and sector numbers, a well organized display appears on screen. The first column of numbers on the left side of the screen represents the drive number; the second column, the track number; and the third column, the sector you selected. Next is the column containing the byte numbers, or the location of the first byte of each row in the display. All columns are separated by a blank column for easy reading.

In the middle of the screen is a hex dump display of the sector you selected. To the right of the screen is an ASCII dump of the same sector. Each dump has its own flashing cursor. The cursors move relatively, indicating the ASCII byte that corresponds to the hex byte.

The most fascinating aspect of the program, the Hash Index Code, can be demonstrated while the display is on the screen. First, read a sector from the directory track, (11), sector 2-9. Choose a program listed in the directory and move the cursor down over the first letter of the filespec name.

By pressing H, a new column instantly appears on the screen between the hex and ASCII dumps. At the top of this column will be the Hash Code number for the filespec. Directly below it will be the words Hash Code spelled out vertically, with an arrow pointing to the

✓Reader Service—see page 226

code at the top. Below this will be the track number, sector number and byte location to install the code to recover a killed file. By pressing the minus key you are able to page back to the first sector in the directory track, where you may apply the code to recover the file. The code, as well as all other information, remain on the screen as you page.

While in the display mode, pressing M, modify, allows you to change any byte in the display.

While in the modify mode, the hex cursor increases in size. Any key pressed (0-1, A-F) modifies the byte at the current location of the cursor. After each modification, the cursor advances one position.

While in the modify mode, you have access to the W, write, command. The W command does not immediately write to disk. Instead, the prompt comes up, asking you if you desire to write to a drive other than the one you read in. If the destination is the same, simply press ?ENTER? and the write takes place. You then return to the display mode.

Z80ZAP even makes sector compares easier by using the C key. Pressing C gives you a byte-for-byte comparison of any sector on any drive to the sector on the display. This function is nice to have, if a voltage spike or some other little gremlin got into your diskette. An example of this is a diskette that does not boot up because of a zapped byte in BOOT/SYS. The compare function compares this sector to a good diskette and pauses at each byte that is not the same. Pressing ?ENTER? continues the comparison.

Another invaluable function is the F, find, command. With a sector of data displayed on the screen, press F, followed by a two-digit hex byte. The flashing cursor goes off to look for the byte and resumes flashing directly over it—if it exists. If the sector does not contain the data, the cursor returns to the first location in the display. This function is great for apply patches or fixes.

Using the L, locate, function lets the program move the cursor for you. Press L, followed by the byte number you wish to pinpoint, and the cursor moves automatically over the location. This function also comes in handy in applying patches.

To escape any of Z80ZAP's modes, press X. This will return the flashing cursor to the first byte of the display.

Following are some of the other important functions of Z80ZAP:

- \+ or – Pages forward or backward, one sector at a time.
- Z Zeros out a sector, handy for cleaning up cluttered sectors on your diskettes.
- J Jumps to DOS READY. A word of caution here: After using Z80ZAP, always reboot the system, as the author resets the stack pointer to a different location than DOS does.
- D Puts you in debug: "G" [ENTER] returns you to Z80ZAP.

The documentation also includes easy instructions for removing passwords from all files, as well as extremely easy directions for recovering a killed file. ■

Model 800 Printer
Base-2
Fullerton, CA
$499

by Milan D. Chepko

Shortly after adding a disk drive to my TRS-80, I decided it was time to turn it into a decent business system by adding a hard copy device.

I considered Radio Shack's price of over $1000 for their line printer to be out of the question, so I started looking around in the computer journals. There are several companies producing less expensive line printers compatible with the TRS-80, but I finally settled on the Base-2 Model 800.

Among its many hardware and software features, perhaps the best is the low price—$499 for the standard model with friction-feed paper drive.

The printer arrived about six weeks later than initially promised, and the paper rack arrived even later.

Lighter Than Centronics

Weighing 10 pounds and measuring 14 by 10 by 3 inches (six with tractors), the Base-2 printer is considerably smaller and lighter than the standard Centronics 779 printer from Radio Shack. It prints 60 lines per minute bidirectionally. Its ribbon cartridge is supposedly good for five million characters.

All the component subassemblies use connectors on the cables, so replacement should be relatively straightforward. The construction is generally sturdy and, though it should tolerate considerable use before needing repairs, I believe the Centronics 779 would stand up better under heavy day to day use.

The Base-2 printer uses an 8085 chip and two ROM chips, along with up to four RAM chips to achieve some sophisticated functions. For example, six different character densities are available per line (64, 72, 80, 96, 120, and 132), and can be selected by a back panel DIP switch or by a software control program.

The DIP switch on the back panel is used to establish a standard print density. This can be changed at any time. I generally leave mine in the 80 character position. This generates an automatic line feed after a carriage return, which is needed with the TRS-80.

In addition, the characters can be printed in double width in any of the densities, giving you 12 different type sizes. The double-width characters can start and stop in the middle of a line for highlighting. Also, lowercase characters can be printed by generating the proper ASCII code. (On the TRS-80, this is done by holding down the shift key while typing the letters.)

The printer comes with two different sets of characters—standard and APL, both in ROM. There is room for three more character sets in ROM (although details are a little sketchy in the instruction book). A sixth set (maybe the TRS-80 graphics?) can be loaded into the on-board RAM by program software. Characters are printed in a 5 by 7 dot matrix.

Changing the different character sizes and fonts is relatively easy. Issue an escape character, followed by the control code describing the desired function. With the TRS-80, the sequence would be:

```
LPRINT CHR$(27);CHR(X);
```

The codes are all described in the book that accompanies the printer.

Three options can be added to the standard Model 800. Option S controls the paper advance from one dot to more than full-character height. It permits rapid paper advance without forcing the print head to traverse the paper, which is important in printing graphics. Since this is a bidirectional printer, there is some variance in character position from line to line. Option S prints in one direction, which eliminates most of the wobble in the printhead-drive mechanism.

Option T is the tractor feed and I recommend it based on my limited experience with friction feed. Option M adds extra memory, enabling the printer to accept 1920 characters from the CPU in a burst, then print them out while the processor is doing something else.

Each of these options adds an extra $50. But, in order to add the tractor feed (T), you have to add paper advance control (S). The tractor feed version ends up costing $100 more than the standard model. However, this is still $300 to $400 less than the comparable Centronics model, which lacks many of the Model 800's features.

Versatile

Unlike many printers, this one is versatile with a capital V. In addition to the Centronics port required by the TRS-80, there is a serial port handling baud rates up to 19,200 and accepting RS232 or 20 mA current loop signals, along with an IEEE 488 port that should be compatible with Pet systems.

In general, the Base-2 printer is an excellent product, and should provide good service for most TRS-80 owners. As an added gift, it is quiet. Unlike the Centronics 779, the Base-2 printer is completely silent when it is not printing. If your printing workload is light to medium, the Model 800 could be just what you're looking for! ■

120 CHARACTERS/LINE

132 CHARACTERS/LINE

ELONGATED

Sample printout from the Model 800.

✓Reader Service—see page 226

80 ACCOUNTANT

by Michael Tannenbaum C.P.A.

"For the professional accountant or executive. . . (Visicalc) is a pure gift."

One of the most frequent questions that I have been asked is how the reliability of the Model II compares with the Model I. Until recently I was unable to give a reply. My Model II has performed virtually without a flaw since I acquired it. That is, until last week.

All of a sudden the program I was working on dropped out for no reason. When I attempted to reboot, the operating system gave me a memory fault error message. After several more trys, I gave up and brought the computer to my neighborhood computer center.

After a brief trial, the salesman agreed that the computer was not functioning properly and turned it over to the capable hands of the repairman. Twenty-four hours later it was returned properly repaired. Frankly, while I was not overjoyed to have the computer break down after it was out of warranty, the experience was not unpleasant. When the Model I had problems, the computer took so many trips to the repair center I thought they owned the machine.

Pioneered with Model I

Of course, to be fair, I was a pioneer with the Model I. Not only did I purchase one of the first 16K Model Is, but I also had one of the first interfaces and disk drives. When the machine broke down, the local repair center was unequipped to deal with it.

A major difference between the two computers lies in the diagnostics that the Model II automatically performs on power-up. This prevents you from using an improperly performing device.

Unfortunately, it is quite easy to use a sick Model I. Because much of its operating system is in ROM, a memory fault could remain undetected until a debugged program bombed. During the time the chip was inoperative, the owner could be quite unaware that the machine was causing problems. If the computer was just used for playing games, such a situation need not be fatal. However, if the Accounts Receivable or the General Ledger file was destroyed as a result of the flaw, recovery would certainly be painful.

I recommend that Model I owners acquire any of the diagnostic programs that are available and run them on a regular basis.

Well, now that we have covered the "unthinkable" system failure, let's get down to the subject of this month's column. . . Accounts Payable. Accounts Payable recordkeeping is a requirement for almost all businesses. Bills must be accumulated and paid for supplies, services and merchandise. Since this type of activity usually involves the analysis of invoices, coding for subsequent posting to the general ledger, posting to a vendor's payable ledger, preparation of a check and preparation of a remittance advice, much work is required. In fact, much of the data has to be written over and over again.

The tedious nature of this work has resulted in the development of many different Accounts Payable systems. One of the most common is the so-called One Write system. In a One Write system, a series of forms is designed so that the repetitious data is reproduced on all the documents by means of carbons with a single entry.

Another widely used method is the accounting machine, which, like the One Write system, uses special forms and requires careful positioning when posting data.

Recent models of electronic accounting machines are computers in almost every sense of the word. These electronic accounting machines, like their mechanical predecessors, maintain data on individual ledger cards, much like the generations of bookkeepers before them. Unfortunately, most computer systems do not generate a ledger-card type of record.

A ledger card contains not only the detail of open transactions, but also a record of historical closed transactions. Because a microcomputer does not have the capacity to retain closed transactions, a ledger card cannot be produced. If the lack of a ledger card makes it difficult for your staff to function, I'd advise against a microcomputer payable system.

At the end of last month's column, I indicated that the Radio Shack General Ledger system appeared to be the first element of a fully integrated accounting recordkeeping system. The Accounts Payable system catalogue #26-4505 is the second element. This system can stand alone or be used with the General Ledger. If it is to be integrated with the General Ledger, a second drive is required.

Prior to using the system, an initialization program must be completed. You are required to specify the firm's name, address, system password, internal ledger codes (GLC)s and accounting method to be used.

The GLCs are used by the payable system to identify ledger accounts. They are not compatible with the codes used in the General Ledger system. For example, an expense account requires two codes, one for Accounts Payable and one for the General Ledger. *Both* codes

continued to next page

EDUCATION 80

by Earl R. Savage

Education 80: The role the 80 plays in teaching/learning/educational experiences. This is the world of the classroom *AND* the homeowner learning to repair a leaky faucet; the driver learning to tune up his car; the new employee learning his job; the old employee learning new skills for a promotion.

Well, you have the idea. The young, old and in between are involved in teaching/learning/ education at school, at home, work, play—everywhere. And the 80 can be the most effective aid to those learning situations if we can enlist its help.

That is exactly what we will be discussing in this space each month: using your 80 in all kinds of instructional situations. We will exchange ideas on planning, programming, evaluating and improving. We'll get into software, both home-grown and commercial, and even into hardware.

Sharing Techniques

In this column, we'll share thoughts and techniques for making the 80 a better teacher. That's right—*share*. We will discuss what I have found and the findings of others, including you. Each one of us knows something about the 80 and about a skill or process that others want to learn. Now is the time to begin to put it all together.

Me? Well, I've been in education and electronics more years than I care to admit. As far as the 80 is concerned, I am far enough "into" programming and computer assisted instruction (CAI) to have gained a bit of skill and glimpse its potential.

I am convinced that we are on the threshold of a real revolution in both formal and informal learning. The microcomputer will bring changes beyond our current imagining. Teach-

continued to next page

80 ACCOUNTANT

From page 26

must be available to Accounts Payable, if automatic posting is desired. Failure to do this accurately will automatically abort the General Ledger update at the end of the month.

When using an integrated Payable and General Ledger system, it is a good practice to specify a Miscellaneous Account. In the event that the specific code is not known, the operator of the system will have an account to use which is acceptable to both the payable and ledger systems.

Looking over the sample reports supplied with the system, I was quite pleased to note many "big" system features. In addition to an Invoice Listing, Posting Report and an Aging Summary, there are Cash Requirements Reports, Check Preview Reports and a Discounts Lost Report. Invoices are specified where a discount could be lost through failure to make a prompt payment. This feature, if properly utilized, can easily pay for the whole system.

The information package includes a sample check and an order form for supplies. I do not see any way to customize the check printout, so you will probably be forced to order a supply of window envelopes. Using this envelope eliminates the need for addressing envelopes to the vendor.

The screen formatting and the controls available to the user are quite advanced. I especially like the file maintenance screens. If you cannot remember the number of a vendor, you can search by name. If you type just a few letters, the computer will automatically put you into the proper range for retrieving the information. You can then use the "previous" or "next" options to locate the precise number.

The system has a capacity of up to 500 vendors and 3,000 unpaid invoices can be accommodated. As indicated previously it will run on a one drive Model II.

Visicalc for Model I

Just before I completed this article, Radio Shack announced the availability of the Visicalc Software package for the Model I. This software package was originally developed for the Apple II and has proved so desirable that many Apples were sold on the strength of it.

For the professional accountant or executive who is responsible for planning and projections, this program is a pure gift. It turns the TRS-80 Model I into a multi-column, multiline worksheet. Each column and line can be programmed so that a change in any one figure can automatically change all other related figures. For example, if you are budgeting a 12-month period, and each month's activity is related to a prior month, you can project an entire year's activity by simply entering the first month's data.

The program is designed so that the work area can be scanned either horizontally or vertically in columns one to sixty, or lines one to two-hundred-and-fifty-six. Any portion of the work area can be printed and the results of the calculations stored on the disk for recall at any time. I urge you to see a demonstration of this software. I'm sure you will be as enthused as I am.

In the several months that I have been preparing this column I have attempted to mix a little accounting theory with program reviews and recommendations. I am not sure how well this format has succeeded to date. I would certainly welcome any comment on how the column can be made more interesting. If you wish to contribute to the column, I will be glad to include your comments with the appropriate credits.

Since many of the TRS-80s currently being sold are disk systems, I am certain that there are many people who are using them for business purposes. Sharing experiences with others can be an excellent way to increase the value of the machine to all. I look forward to hearing from you. My address is: Michael Tannenbaum, CPA, 42 Bulaire Road, East Rockaway, NY 11518. ■

EDUCATION 80

From page 26

ing/learning will never be the same. We have barely scratched the surface.

All we have to do is apply our programming skills to create situations that help others learn most effectively.

You are invited—no, urged—to participate. I welcome your questions, comments and suggestions about this column. I and other readers will be most interested in *your* ideas, software techniques and hardware designs.

What have you found that works well? That doesn't work? The rest of us would like to know. Write me in care of *80 Microcomputing*, and they will send you letter along. If you would rather "write" on a cassette, that's all right, too.

Oh, yes; I will answer as many of your letters as time permits, especially if you have included a self-addressed stamped envelope.

Recording Progress

Suppose that you have an instructional program that the user studies independently in school, office or wherever. Further, the "instructor" wants to keep a record of the user's progress.

Often in such cases, the user and/or the instructor simply writes down the final score from the display. It is much more satisfactory to have a printout of that information and that can be done with the simplest printer.

The following few statements will do the job on the little RS Quick Printer II. (Change the commands to suit your printer.)

```
1030 PRINT "PLEASE SWITCH ON THE PRINTER AND
     THEN PRESS ENTER."
1035 B$ = INKEY$ : IF B$ = ""THEN 1035
1040 LPRINT "NAME
1045 LPRINT "PROGRAM : AMATEUR THEORY II"
1050 LPRINT "SCORE : "R" RIGHT OF "T"
     ATTEMPTED"
1055 FOR X = 1 TO 4 : LPRINT : NEXT
1060 END
```

Certainly, the variable names will have to be made to agree with those in the program. Everything should be clear, except perhaps line 1055, which simply runs the paper up four lines so that the printout clears the tearbar.

This small section prints the essential information, but is easily expanded to include whatever you want on the record. Best of all, not only can you put it in programs that you write, but it can be inserted easily into any you wrote and any that you've bought.

Program Modifications

A well known expression states, in one form or another, that one should not waste time reinventing the wheel. So it is with programs. If there is an existing program that meets your needs, why spend time writing your own? Better to use it, while writing one that does not exist.

There are instances, however, when a program does part of what you want. It may be a game which uses the information you wish to teach. It may be an instructional program but written for the wrong level—too easy or too hard for the student(s). It may have too much or not enough explanatory material.

My own approach in these cases is to modify the existing program rather than begin at zero. Let's look at an example.

Suppose you want a tutorial program on Newton's Laws of Motion. Suppose, also, that you have one of the many forms of that popular game in which the player attempts to land a space vehicle on a planet. The game provides excellent application and practice in using Newton's Laws. Your best bet is to use that program and add the instructional part.

Here are some of the elements you would consider adding:

- Title introduction including personalization;
- Tutorial material, *per se*, on the level and to the depth appropriate;
- Introduction to and rules of the "game";
- Choice within limits of the number of times to play;
- Reward/reinforcement frames for success;
- Review frames for failure;
- Scoring and score-keeping;
- Summary review and conclusion.

As you see, even a game of the entertainment type can be turned into a valuable instructional program. There are many such possibilities. Often, too, relatively minor modifications will turn a poor program into an excellent one.

Don't re-invent the wheel. Get in there and apply it in a different way. In later columns we'll discuss the elements listed above and find ways to make them easy to add.

I'll meet you here next month for more Education 80. ■

80 APPLICATIONS

by Dennis Kitsz

"As you may have discovered when trying to install the high-speed modifications, Radio Shack has made some changes."

The gentleman in the back. Yes, in the green shirt. What was that? Customize your TRS-80? Sure, folks, just a few simple do-it-yourself changes can turn your computer into a funny-looking, but more powerful machine. Step right up, heat that soldering iron. It's time for the Autumnal Equinox Hardware Festival!

This month we will be adding a second video jack, putting a radio frequency (RF) modulator into an expansion box, getting rid of the RESET lockup in the expansion box, bringing RESET up front, adding Level I and Level II, adding a keyboard expansion socket, eliminating the LPRINT/LLIST hangup, and doing some preliminary thinking about real-time clocks. And you can get nearly everything you need at your local Radio Shack.

In Preparation

First I have some mixed news for those of you with newer TRS-80s.

As you may have discovered when trying to install the high-speed modifications, Radio Shack has made some changes. The two-chip ROM set is slow, and takes up both available ROM sockets; the cassette load improvement modification ("XRX") prevents data I/O at high speed; and the new expansion box is somewhat locked into standard speed. Also, the updated no-bounce keyboard style makes it harder to install top-mounted items such as a control key, and so forth. Radio Shack has gone through so many alterations to the computer that models now hardly look like my own antique machine.

Let me offer a few precautions before starting. Open the case carefully, setting aside each part in order (six screws and six white keyboard grommets). Work on a large surface, and set the unit on a towel to protect its finish. Flex the keyboard's cable as little as possible, and put absolutely no stress on it. And most of all, always make any changes with the power off *and disconnected.*

Let's start with an easy one, the installation of an extra video jack. You'll need a subminiature plug/jack pair, a video jack, wire, solder and cosmetic tools.

You can always use this extra video jack, especially if your machine makes frequent appearances before the general public. You may add almost any sort of jack for which you have abundant matching connectors; I used a simple coax jack. Since long wires will tend to make your video signal look messy, choose a point near the normal video outlet to mount the new jack.

The plastic used for the TRS-80 case is soft and very pliable, almost enjoyable to work with. Exact-o knives, drills, wood or linoleum cutting tools, and even screwdrivers can be used to cut the plastic to the right shape and smooth out a hole.

Open the case and select the placement of the jack carefully so that the 80 will reclose without striking or binding on the new hardware. Cut or drill a mounting hole, and install the video jack. Be sure to blow out the bits of plastic, because these chips, which will cling by static, can cause havoc underneath keycaps.

Now examine the TRS-80's normal video output (DIN) jack; it has five pins, three of which are used. Fig. 1 shows the video area with the cover removed.

You will need only two pins because the five-volt output pin is used for activating the optical isolation circuitry in the video monitor. Isolation is necessary because, unlike most commercial monitors, this one has "hot" parts uninsulated from potentially dangerous AC power.

On Fig. 1, locate the video and ground traces (D and B) where the DIN jack connects to the circuit board. Obtain a miniature audio connector, and mount it on the board, soldering the center (hot) lug to point D, and the ground lug to point B.

Solder a short piece of coax cable to the mating connector, and run this to your new video jack. I used phono cartridge mounting hardware for this process. In any event, if ease of getting in and out of the TRS-80 is important to you, then mating connectors will allow you to work on the computer without having to clip, solder, or disentangle anything.

Insert the connectors, snap the case together, hook up your second monitor, and power up. Ahhhh.

Picture and Sound, Will Travel

Do you carry your video monitor everywhere you go with your TRS-80? There's no need to do all that lugging. Using a simple RF modulator available in kit form, you can put computer power in your back pocket (if you have very big pockets). The materials you'll need are the modulator, a case, two five-pin DIN jacks, one DIN cable.

Start with the RF modulator—Radio Shack sells a $15 Project Board and complete surplus units are easily found. Assemble it according to the directions. Hook it to an ordinary television, and temporarily run computer video (from the new output jack) to it. The picture quality will be underwhelming, but that is mostly a function of the limited bandwidth of commerical TV receivers.

For simplicity, you can put the whole assembly in a small equipment box, driving it with either your new video output jack or the built-in one. For the modulator's case, I used the empty slot reserved for the RS-232 board in the expansion interface, although any classy box will do.

The center pin of the video DIN jack (pin C in Fig. 1) in the TRS-80 is not used. To it, attach a wire from the cassette port's audio output line (pin G on the cassette connection shown in the

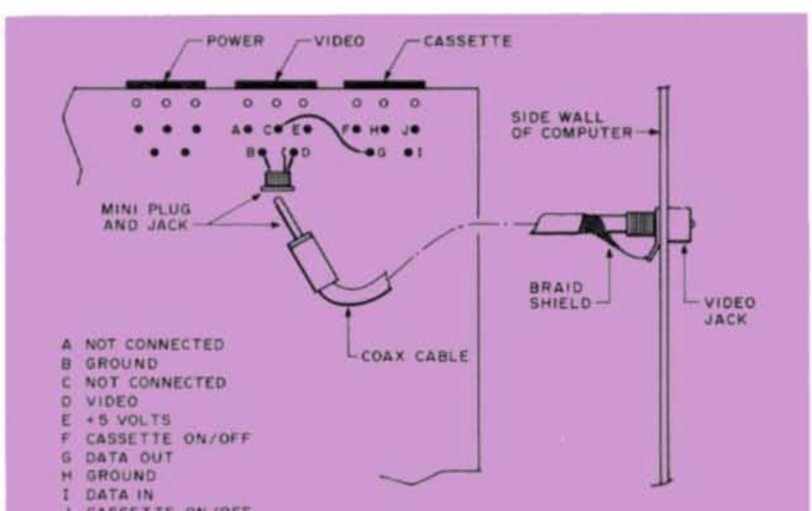
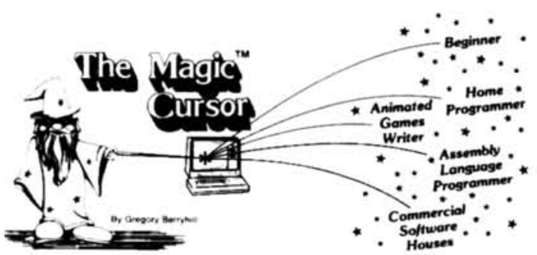


Fig. 1

TRS-80* OWNERS:

- **Let the computer write your "Basic" program for you!**
- **Draw pictures, animated figures, data forms!**
- **Create a library of display forms!**
- **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 or 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 MAGIC 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 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 Disk) **$149.95**

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

WRITE FOR OUR COMPLETE SOFTWARE CATALOG!!

Call 713/474-2484 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. Some programs supplied on cassette tape. For disk versions, the cassette supplied will automatically create a disk file.

NEW RELEASES FROM CCC!!!

The Supply Room

A sophisticated program that is easy to operate. This unique system helps you manage your retail or wholesale inventory of up to 1000 line items. Cost may be fractions of a cent, item quantities may be decimal values. Automatic cost averaging of updated items and suggested sales prices based on gross profit. Transaction reports produced during posting functions. Printed reports include:

1. Transaction Listings.
2. Complete Inventory Listing.
3. Listings by Prefix.
4. Items on Order.
5. Items Out of Stock.
6. Suggested P.O.'s.

Requires Mod 1, 32K, 2 Disk **$250.00**

The Filing Cabinet

Take that collection of Job Orders, Personnel Records, Reports, Etc. and get them organized. You define up to 14 data input fields specifying field type - alpha/numeric or $$$, field length and field separators - such as slashes, etc. During data input, the cursor provides for character input and skips over the field separators.

But that's only half the story. Output reports are automatic formated and program automatically requests column headings.

This means you can customize your own data base manager, complete with rapid data input and selective output reports. Comes complete with documentation.

Requires Mod 1, 32K, Disk & Printer **$100.00**
Or Mod 2, 64K, 1 Disk & Printer **$150.00**

The Restauranteur's Consultant

by Ty Halderman

This food and beverage management tool dramatically reduces the human factors in food cost analysis. Those tasks required to effectively operate any restaurant or food service business. Over a half dozen reports give uniform, accurate and up to the minute information for profitability. And, handling of daily cost changes requires only minutes per week instead of hours, because the Consultant makes all the necessary conversions from your case prices. Reports include:

1. Menu Recipes.
2. Ingredient Listing.
3. Supplier Master File.
4. Complete Listing of Food Cost Analysis.
5. Food Cost Summary.
6. Input Data Sheets.
7. Batch Update.

Mod 1, 32K, 2 Disk, Printer or Mod 2 .... **$750.00**

*Trademark of Radio Shack, a Tandy Co.

Send Check or Money Order Payable to:

CUSTOM COMPUTER CENTER, INC.
Attn: Jim Martens
P.O. Box 58042 / Houston, Texas 77058
or call: (713) 474-2484

QUAN.	DESCRIPTION	SYSTEM (Size: Tape or Disk)	PRICE
	Handling Charge: $1.50 Texas Res. add 6% Sales Tax	TAX & HANDLING	
		TOTAL	

MASTER CHARGE / VISA — **SEND INFORMATION ONLY.**

Card # ____________ Expires ____________

NAME ____________

ADDRESS ____________

CITY ____________ STATE ____________ ZIP ____________

Reader Service—see page 226

The Radio Shack catalog numbers, descriptions and prices for all the parts needed to compute the modification are shown below:

General stuff		
Wire-wrap wire	278-501-2,3	1.99
Solder-up	64-2090	1.49
Added video jack		
1 F-61 video connector	278-212	2/.89
1 Miniature plug/jack set	274-283	2/1.99
Added RF modulator		
2 5-pin DIN plugs	274-003	1.49
1 RF modulator (parts extra)	277-122	16.95
1 Case (optional)	270-232	2.49
1 DIN cable	42-2151	4.99
Modified E/I RESET		
1 Miniature SPST switch	275-612	1.69
1 10K ohm resistor	271-1335	5/.39
Added RESET button		
1 Pushbutton	275-1547	5/2.49
1 Miniature plug/jack set	274-283	2/1.99
Level I Addition		
1 Miniature DPDT switch	275-614	2.19
2 1K ohm resistors	271-1321	5/.39
1 Level I ROM set	(see text)	$10 bribe
Keyboard addition		
2 16-pin wire-wrap sockets	276-1994	2/1.39
1 16-pin solder-tail socket	276-1998	2/.89
1 16-pin jumper cable	276-1976	3.99

Parts List

@	A	B	C	D	E	F	G
H	I	J	K	L	M	N	O
P	Q	R	S	T	U	V	W
X	Y	Z					
0	1	2	3	4	5	6	7
8	9	:	;	,	—	.	/
ENT	CLR	BRK	UPAR	DNAR	LFAR	RTAR	SPACE
SHIFT							

Table 1

figure). Inside the RF box, wire two DIN jacks in parallel, hooking them up to the RF modulator as in Fig. 2.

A short DIN cable connects the computer to the RF box, the TRS-80's original monitor plugs into the box's other jack, and the RF output goes to an ordinary television. See Fig. 3. Picture and sound can now travel with me and my TRS-80.

Unlocking RESET

Speaking of the expansion box, I'm amazed that Radio Shack has let users suffer with a RESET button that locks up the computer when an E/I is connected. There's no good reason for leaving it that way, and fortunately the solution is very simple. You'll need a single-pole, single-throw switch and a 10K ohm resistor.

Open the expansion interface, and locate Z32 (near the power switch). If you have a newer expansion interface, this will be marked Z39. In either case the part is a 16-pin IC, type 74LS155. Now find pin 4, and identify the trace that leads underneath the IC and out its other side. Look carefully, using an ohmmeter to be sure it's the right trace.

Got it? Okay, take a sharp blade and cut through it. Run a 10K ohm resistor from the far side of the trace to pin 16 of Z32 (or Z39), and solder it carefully. Finally, run a pair of fine wires, one from each side of the cut trace. Solder one wire to each connection of the miniature toggle switch. When the switch is on, the cut trace is bridged, and the expansion interface acts normally. When the switch is off, the RESET button again works correctly.

Why did the expansion box hang up the computer in the first place? When you power up or hit the RESET button, the CPU clears a few internal items, and then sends out a signal to hex address 37EC, asking for information about the disk system. If it doesn't get anything (no expansion box), it continues to READY (or MEMORY SIZE? at power up). But if it does find something (the disk chip in the expansion interface responding with the proper value), it sits and waits for the disk to start rolling and the data to come in. No disk . . . no data.

When we cut that single trace, we took the disk system's answer from the circuit. The disk controller is still sending a signal, but the computer doesn't see it. The CPU goes happily on to READY. As for the 10K ohm resistor, it acts like an electronic rope, restraining the answer circuit from triggering because of the high-speed electronic madness inside the expansion box.

Even disk owners will find this modification valuable, because it allows the disk drives to be disabled without flipping a glitch-inducing power switch. Also, readers using the LNW research expansion board may make the same modification. The IC is marked U19, but the rest of the process is identical.

An Easy-to-reach RESET

Where is your RESET button, anyway? In frustration, have you torn off the computer's little silver port cover? Maybe bent the cable and crashed the system when reaching for RESET? Well, then, bring the button up front!

Following the same carpentry precautions used for the extra video jack, install a momentary-on pushbutton (get them in packs of five from Radio Shack) at the left side of the computer. Just two wires are needed, as only half of the TRS-80 double-pole RESET button is actually connected in the circuit. See Fig. 4.

As before, it is wise to use some sort of mating disconnect plug so that access to the machine innards is simplified.

Don't put the button *too* close to the top of the keyboard, as many an Apple user has shed tears over hitting its badly positioned RESET button. Imagine being on the 7,000th word using Electric Pencil, and accidentally swatting RESET!

Level I BASIC on Level II

Your kids are having trouble with triple-dimensioning strings? They can't make sense out of ?DD ERROR and holler about having to type long lines of command words? Then why not use that nice and easy starting language—Level I BASIC?

But you just upgraded to Level II, you say! Fear not, for the virtues of both are available to you for the price of a switch and two resistors. Let's hope you've saved your Level I ROM . . . you paid for it the first time. If not, seek out a cooperative ($10 usually encourages cooperation) Radio Shack manager and obtain a Level I ROM set from the upgrade pile in the back room.

Column 1 — R8 — @ H P X 0 8 ENTER SHIFT
Column 2 — R5 — A I Q Y 1 9 CLEAR
Column 3 — R3 — B J R Z 2 : BREAK
Column 4 — R2 — C K S 3 ; UP-ARROW
Column 5 — R7 — D L T 4 , DOWN-ARROW
Column 6 — R1 — E M U 5 — LEFT-ARROW
Column 7 — R4 — F N V 6 . RIGHT-ARROW
Column 8 — R6 — G O W 7 / SPACE

Table 2

Row 1 — Z1 PIN 8 — @ A B C D E F G
Row 2 — Z1 PIN 2 — H I J K L M N O
Row 3 — Z1 PIN 10 — P Q R S T U V W
Row 4 — Z2 PIN 2 — X Y Z
Row 5 — Z1 PIN 6 — 0 1 2 3 4 5 6 7
Row 6 — Z1 PIN 4 — 8 9 : ; , — . /
Row 7 — Z1 PIN 12 — ENT CLR BRK UPAR DNAR LFAR RTAR SPACE
Row 8 — Z2 PIN 4 — SHIFT

Table 3

The ROMs you want are marked National, and are identified by the numbers M2316E/MMS258ET R/N and S/N, or Motorola, marked 7807 and 7804. Best of all is the single-chip Motorola ROM, marked 7809. You do *not* want chips marked Intel; these were very early ROMs, and wiring is complex.

Check to see if you've got a two or three-chip Level II ROM set. If you've got the three-chip set, there will be a connector cable running to a separate board. If not, both ROM sockets will be filled, and that means no room for Level I.

Handle the ROMs carefully (use special black foam or aluminum-foil-covered vegetable trays). Mount a double-pole, double-throw switch conveniently, but not where you are likely to flip it while using the computer.

Open up and turn over your TRS-80. The Level II interconnect cable will be plugged into one of the two sockets, and several additional wires will be connected from the Level II board to various parts of the main circuit card. Find the following locations:

- The green wire from the Level II board, connected near the underside of dip shunt Z3.
- ROM Socket Z33 or Z34 (whichever is empty), pins 18 and 20. See earlier "Applications" columns if you need to know how to read pin numbers.

Cut the traces leading from pins 18 and 20 of the unused ROM socket. Add a short length of

Enjoying 80 MICRO? then read on...

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 TRS-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 $18, 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 $2.50 at your computer store, that's $30 a year. For $18 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 $2.95 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 $45. 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.

*TRS-80 is a trademark of Tandy Corp.

wire between pin 20 and the *other* end of the cut trace from pin 18. Solder a long white wire to pin 18.

Now remove the far end of the Level II ROM board's green wire from its connection point near shunt Z3, and solder a red wire in its place. A blue wire goes to the five-volt supply. I used Z57, pin 14. Run the other ends of the white, green, red and blue wires to the switch following Fig. 5.

Note also that two 1K ohm resistors are necessary to tie up the ROM chip-select leads when they are switched off. The resistors hold the chips inactive.

Now here comes the hard, or fun, part if you don't have the single-chip Level I set. Find the notch at the top of each ROM chip, and line up the two chips. Now *piggyback* one atop the other, and solder *all 24* pin pairs, so that you've got a single, integrated circuit. Do it carefully, and be sure to keep the bottom IC anchored in some conducting material, like aluminum foil. The foil also acts as a heat sink, so work patiently and accurately; use good, fine solder, and always keep some Solder Up or Solder Wick handy in case a blob of solder fuses some neighboring pins together.

Insert this Level I hulk, or the single-chip ROM, in the empty socket. Snap the computer back together, and power up. You should either get MEMORY SIZE?, if it's switched to Level II, or a simple READY in Level I. You cannot switch back and forth; the languages get confused and "hang up." You must power up to the language you want.

Need a Second Keyboard?

It's an action game, and the youngsters are crowded about the keyboard. One youngster (balding, with grandchildren) complains, "Hey, I'm not close enough to the computer! It's rigged in his favor!"

Need another keyboard? No problem! You'll need two high quality wire-wrap integrated circuit sockets and one low quality solder-tail socket. All three are 16-pin types. You will also need some fine wire (wire-wrap is best), a 16-pin jumper cable, and a keyboard.

The type of keyboard is up to you. Perhaps you'll want a complete alphanumeric type (available in the $40 range) or merely a $10 numeric keypad. Whatever you choose, it should consist of individual keys, each with a single pole single throw (SPST) contact pair. Small calculator keyboards usually have matrix arrangements which aren't compatible with the TRS-80. Its keys are arranged in eight columns and eight rows, as shown in Table 1.

Depressed keys are identified in the TRS-80 software by column and row. Letter T, for example, is column five row three. That simplicity makes adding a keyboard a straightforward task, since no special encoding is needed. To identify any letter, you only need to know which column-row combinations produce it.

There is enough room on the far left side of the keyboard for a 16-pin socket in which to plug a jumper cable leading to the added keyboard. Inside the computer, this location is directly above a blank part of the keyboard's circuit card.

Use a flat screwdriver to snap out the black portion of the cover, and mark precisely where the free area is found on the baseboard. Using needlenose pliers, pull out all the pins from the solder-tail IC socket, and use the socket as a guide to mark the baseboard.

With a small hobby drill, make 16 holes in the board to match the 16 pins of the sockets. Slide one of the wire wrap sockets into the disemboweled solder-tail socket, using the latter as a special-purpose grommet. Feed the wire-wrap pins through the circuit board, and fasten the sockets in place with fast-drying glue.

Now remove the entire cover, turn the board over, and identify pin 1 of the socket. This will attach to column one. On most versions of the 80, you can use the keyboard's resistors to identify columns. See Table 2.

Match column one with socket pin 2. Match column two with pin 2, etc. Solder a wire to the resistors, and wire-wrap (or solder) the other end to the socket pins. Be sure to solder to the ends of the resistors closest to the keys. You will see that the other ends of the resistors are all tied together.

The keyboard rows are found at the input pins of specific IC's on board, but there have been so many versions of the 80 keyboard that this sequence is far from consistent. The technical manual (good for better than half the machines in use, but not mine) identifies the rows as shown in Table 3.

It's better to check it yourself. Look for the traces connecting @, A, B, C, D, E, F, and G. That is row one. Solder a wire to some point in this row, and run it to pin 16 of the socket.

Locate row two, and solder to socket pin 15 and so forth down to row eight, soldering it to pin 9 of the socket. When viewed from the top, the arrangement will be as depicted in Fig. 6.

Once you have completed the wiring, clip the socket pins short, turn the board over, and place everything back in order. Leave the cover off, power up, and run through a few of the usual motions to make sure the computer is functioning properly.

Now take a short length of wire and, at the newly-installed socket, jumper each row across to each column. You should produce all the non-shifted keyboard characters, including the four arrows and the cursor.

Take a second length of wire and jumper row eight with column one. This simulates the shift key. Repeat the column-row test, and note that shifted characters appear. Any bizarre behavior, such as repeated letters, indicates that a wire may be shorted or attached to the wrong column or row.

Last of all, there are the cosmetics to deal with. Cut a rectangular hole the size of the

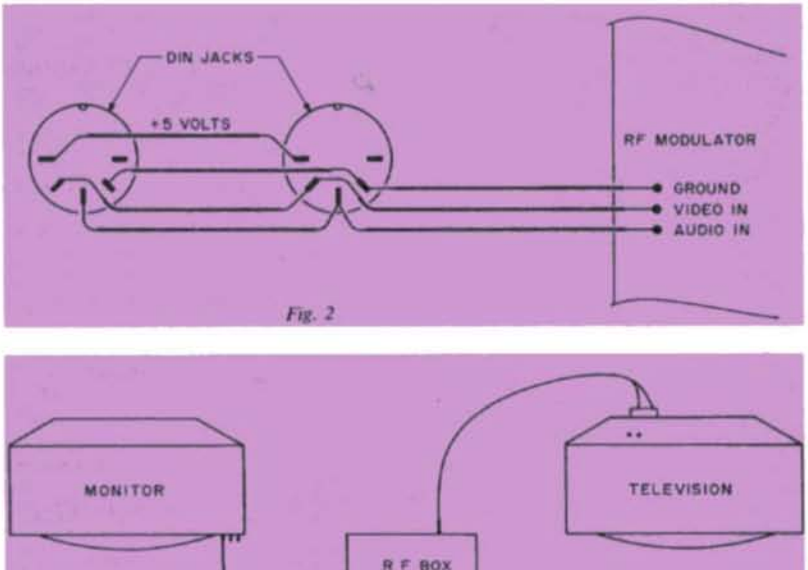


Fig. 2

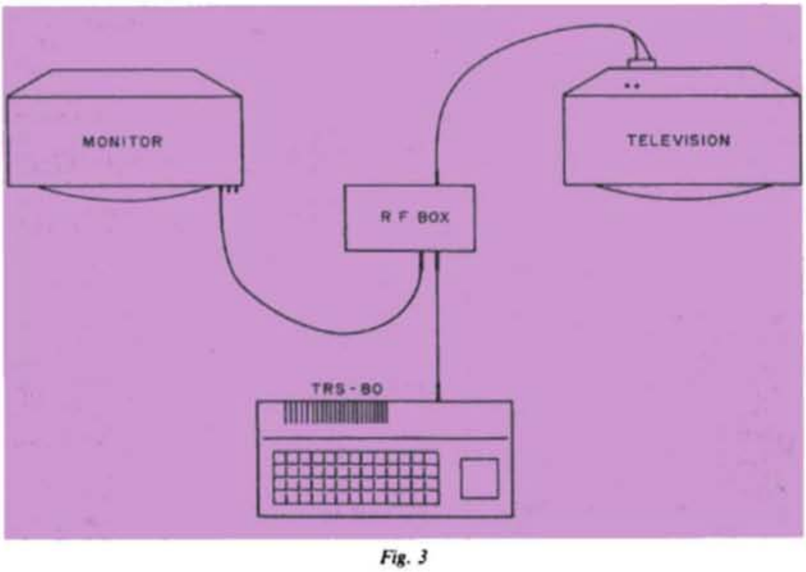


Fig. 3

16-pin socket in the black plastic cover using a hot, sharp knife or razor blade. Work slowly and carefully to achieve a factory finish (whatever that is).

Piggyback the remaining wire-wrap socket into the first one, and snap the cover back on. The socket should fit perfectly, rising about 1/16 of an inch above the surface of the cover. The cable plugs in at a comfortable distance from the keycaps.

For each keyboard you wish to add, work out a row-column key matrix to match the TRS-80's, using Table 1. You can either make the jumper cable an integral part of each added keyboard, or include a socket on it as well. Better yet, put a cable *and* socket on each extra keyboard, so that they can be chained.

No-printer Lockup

If you've got the time and patience, there's a good hardware fix to cure the LPRINT/LLIST no-printer lockup. It involves a few ICs, an edge connector, and lots of wire. Instead, do this:

```
POKE 16422, PEEK (16414) : POKE 16423, PEEK (16415)
```

Now do a few LPRINTs and LLISTs. See, it sends your mistaken Ls to the screen. Where's the hardware?

The previous little item was included to make a point. With computers, there are many ways of achieving very similar goals. Software can run a printer without the mess of hardware used in the expansion box. Although a separate data generation circuit could have been used for cassette output, data is output via software instead. The whys and wherefores are rooted in economics, elegance and versatility. It was not too long ago that I built a nine-channel multimedia controller for a museum out of dozens of discrete logic chips and transistors. A very simple CPU board and some software could have done 30 times that task. Time and technology make changes.

There are those computer users who, if they do not fear working with hardware, at least sneer at its apparent clumsiness. On the other hand, some folks cannot begin to understand the contortions (and labor-dollars) programmers go through to accomplish something in software that a 15¢ chip could do with ease. ■

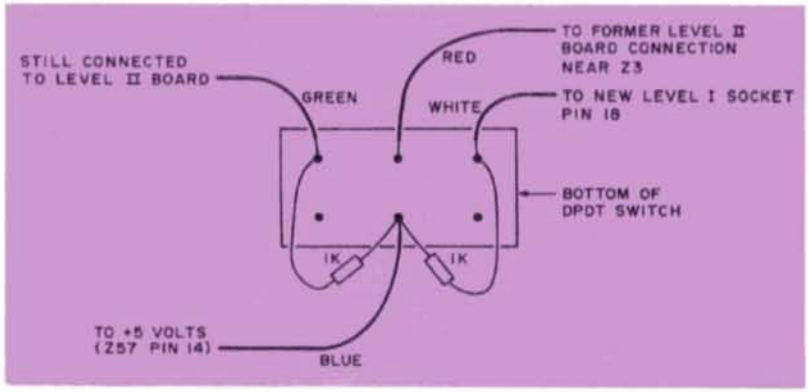


Fig. 5

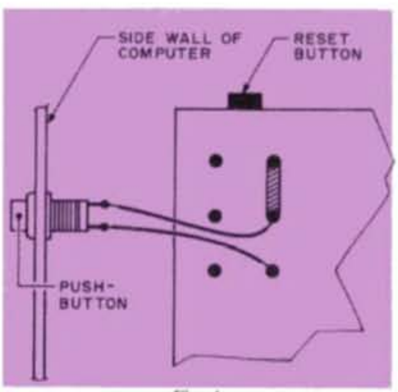


Fig. 4

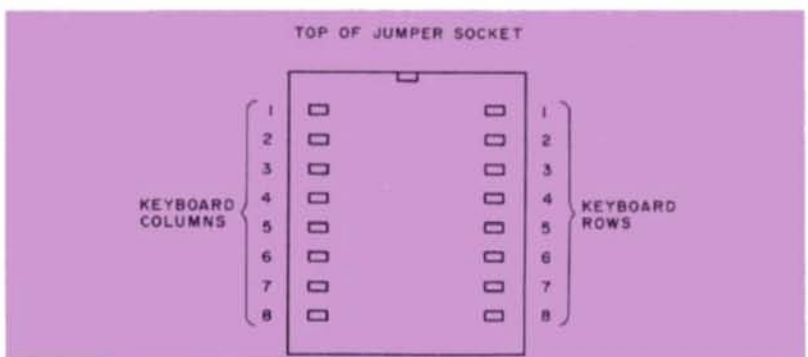


Fig. 6

New Releases for the TRS-80*

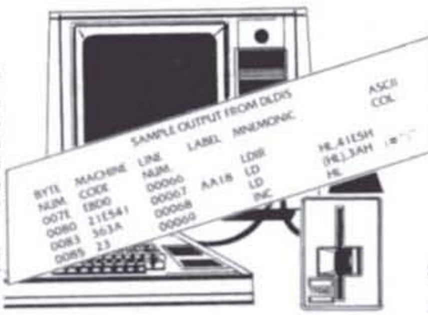
Utilities

We're proud to present three disassemblers for the TRS-80. For speed and simplicity, we recommend The Disassembler. For complex disassemblies, especially if you wish to make alterations, you may prefer one of our Labeling Disassemblers, either TLDIS or DLDIS.

TLDIS & DLDIS

You've bought a super machine-code program, but now wonder how it works. Maybe you even used a quick PEEK routine to glance through it when it was in memory. If so, you definitely noticed the complete lack of comments in the code, making it almost impossible for you to decipher and understand it.

Well, Instant Software's Labeling Disassemblers are the answer to your problem.



TLDIS (Tape-based Labeling DISassembler) and DLDIS (Disk-based Labeling DISassembler) are three-pass, label-assigning disassemblers which assign labels (where appropriate) to the routines in a machine-language program. Their output is almost identical to that of a hand-assembled source code.

You can send the disassembly to a lineprinter (Radio Shack parallel port) for either TLDIS or DLDIS. (The difference between these utilities is the storage mode of the disassembly.)

TLDIS can send the disassembly to cassette tape, DLDIS can send it to disk; both send it to the video monitor. The stored disassembly from TLDIS may be reassembled with Radio Shack's EDTASM(TM)—the disassembly from DLDIS, with Apparat's extension of EDTASM(TM).

Because of the use of labels, it is a simple matter to change any object code program by disassembling it and then making changes to the resultant source code, without losing track of jump/load addresses. Labels start with "AA00" and increment up, in even numbered steps (AA02, AA04, etc.). The odd numbers (AA01, AA03, etc.) are left for you to use for the source code during reassembly.

The printing of the disassembly may be temporarily halted by using [SHIFT] @ (just as in BASIC) or it may be ended by pressing the [BREAK] key. It also has a comments column to display ASCII charachters used in a LD or CP opcode.

Because TLDIS and DLDIS work only on in-memory programs, they may be relocated in memory to avoid conflict with the program you disassemble.

The next time you need to "climb inside" a machine-code program, take DLDIS or TLDIS with you. We promise that it will be an easier journey.

Order No. 0230R (TLDIS) $14.95
Order No. 0231RD (DLDIS) $19.95

The Disassembler

This is a single-pass, hex-notation disassembler that will send its output either to tape or to a lineprinter (Radio Shack parallel port). The tape output is directly compatible with Tandy's EDTASM(TM). Thus, you can take an object code tape, disassemble and output it to tape, then use EDTASM(TM) to add, delete, change and even re-assemble your new version.

In addition, it displays the *displacement* and *absolute* address of any relative jumps made by the disassembled program. It also displays any ASCII characters used in a LD or CP opcode.

Sample output from the Disassembler

BYTE NUM.	MACHINE CODE	LINE NUM.	MNEMONIC	COMMENTS COLUMN	
706E	22057B	00053	LD	(7B05H),HL	
7071	183B	00054	JR	$+3DH	;70AEH
7073	FE52	00055	CP	52H	;="R"
7075	2007	00056	JR	NZ,$+09H	;707EH
7077	CD8F70	00057	CALL	708FH	

H means the number is HEX
$ means current location counter.

Since the Disassembler works only on in-memory programs, it has been made relocatable so that you may move it around in memory to avoid conflict with the program you wish to disassemble. As an added option, you may also jump to memory locations and transfer control between Disassembler and other utility programs in your computer.

The Disassembler, use it to examine and analyze *any* machine-code program!

Order No. 0232R $9.95

Terminal-80

The Terminal-80 package lets your TRS-80 communicate with the rest of the world. These programs give you control of the RS-232 port of your Expansion Interface.

You can connect one or more serial terminals to your TRS-80. Your computer will accept input from the RS-232 port just as if it were entered from the keyboard. Thus, you can use your computer from a remote terminal without having to move your equipment.

The TRS-80 can also be transformed into a "dumb" terminal. You can use it in a time-sharing situation to talk to "big" computers via a modem. All data that you type in will go out through the RS-232 port and all incoming data will be displayed on the screen.

You can transfer programs over the phone lines. Just load a program into the TRS-80. The LPRINT/LLIST command will transfer the program to a receiving computer via the RS-232 port.

Using the upper/lowercase modification of the TRS-80 is simplified. (You must have the modification kit installed first or follow the detailed instructions included in this package.) Control characters in Level II and Disk BASIC will be properly displayed and all functions such as CHR$ will work correctly.

This package even includes a BASIC program to set the baud rate. You won't have to tear apart your Expansion Interface if you use more than one configuration.

There are thousands of TRS-80 computers in the world. Let's get together and talk to each other—with the Terminal-80 from Instant Software.

This package requires the following minimum system:

1. A TRS-80 with 16K of memory.
2. An Expansion Interface.
3. An RS-232 Serial Interface (e.g., Radio Shack's No. 26-1145 or the equivalent).
4. An optional upper/lowercase modification kit.

Order No.0130R (cassette-based) $24.95.

WRITE FOR OUR NEW CATALOG!

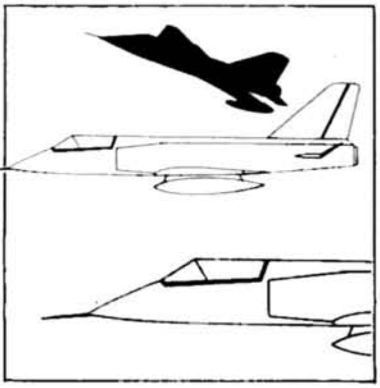
Instant Software™

*A trademark of Tandy Corporation

PETERBOROUGH, N.H. 03458
603-924-7296

✓2

Great Simulations



Jet Fighter Pilot

The Jet Fighter Pilot package takes you as close to real combat flying as possible . . . without pulling G's.

In this brilliantly realistic simulation, you become the pilot of a high performance, twin turbo-jet fighter. Total control of the aircraft is yours.

At the start of your mission, you'll go through an entire engine start procedure before your flight (provided your ground maintenance is up to par). Your takeoff will be from either the deck of an aircraft carrier (via a steam catapult) or from an airfield.

All controls respond the same as they would on a real jet fighter. You'll have to constantly monitor your display and make adjustments to your throttle, flaps, rudder and air spoilers. You decide when to retract flaps, landing gear and release the auxiliary fuel droptanks.

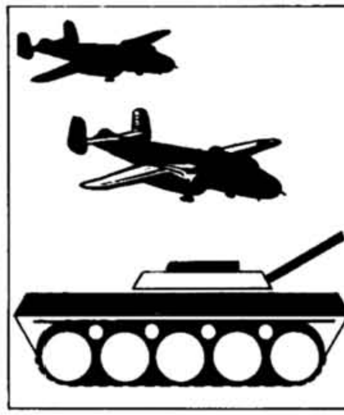
Your on-board navigational computer will direct you to your selected airport. The Glideslope/Localizer information will aid you in approaching and landing on an aircraft carrier deck or airfield.

The Weapons Control Computer will arm your missiles, provide you with the range and bearing to a target, and tell you when to attack. And, if things should get a little too hot, you have an ejection seat command for egress.

For a carrier-based landing, you'll have to deploy your tail hook. For a land-based landing, you'll need reverse thrust and your drag chute.

After you've flown a few missions with the Jet Fighter Pilot package, you'll *know* you've earned your wings.

Order No. 0159R $14.95



Battleground

It is late 1944, and the Allied Forces are sweeping toward Berlin. As General in command of your sector, you study the map. At your command, are tanks, planes, artillery, infantry, engineers and vehicles—an awesome array of fighting men and the machines of war. From Intelligence reports you know that the enemy General is a shrewd tactician, not to be under-estimated. It will take planning and strategy to outwit this wily old campaigner.

The battle map of your sector will fill with markers, each showing the deployment of your forces. You and another player will slip into the roles of opposing German and American commanders as yet another battle unfolds.

Battleground allows you to experience the awesome responsibility of a battle-area command. It will be up to you to deploy your tanks, planes, vehicles, weapons and men. On your shoulders rests the decision, whether to call for direct artillery gunfire, or to order your planes into the air. You will constantly be watching for an enemy airdrop, always carefully maneuvering your forces.

The stark reality of World War II comes alive in Battleground.

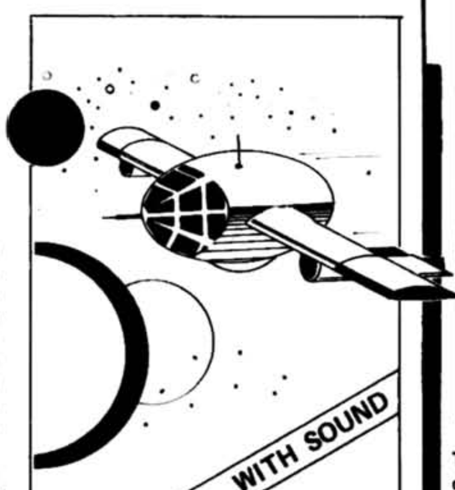
Order No. 0141R $9.95

Cosmic Patrol

WARNING: PLAYERS OF THIS GAME SHOULD BE PREPARED FOR A STATE OF REALISM HITHERTO UNAVAILABLE ON THE TRS-80

The Cosmic Patrol program puts you in the command chair of a small interstellar patrol craft. Your mission is to defeat Terran space and prey on the Quelon supply ships which carry essential parts and lubricants for that implacably hostile robotic force. The drone freighters are fairly easy pickings for the accomplished starship pilot, but beware of the I-Fighter escorts. They're armed, fast and piloted by intelligent robots linked to battle computers. They *never* miss.

The Cosmic Patrol program is not just another search and destroy game. With its fast, real-time action, impressive sound option and superb graphics, this machine-language program is the best of its genre.



Don't keep putting quarter after quarter into arcade games or spending big bucks for video games cartridges. Get Cosmic Patrol from Instant Software—and get the best for less!

Order No. 0223R $14.95

TO ORDER: Look for these programs at the dealer nearest you (see list of dealers on page 199). If your store doesn't stock Instant Software send your order with payment to:

Instant Software
Order Dept.
Peterborough, N.H. 03458

(Add $1.00 for handling) or call toll-free 1-800-258-5473 (VISA, MC and AMEX accepted).

Instant Software™

Prices subject to change without notice.

PETERBOROUGH, N.H. 03458
603-924-7296

THE ASSEMBLY LINE

by William Barden, Jr.

"Numeric computations are very tedious in assembly language, especially when floating-point numbers must be handled."

This month we're going to discuss interfacing assembly language code to BASIC—why it is desirable, what techniques are involved, and what kinds of problems will be encountered when it's done. We'll also be presenting the winner of the Great Second Assembly Line Programming Contest.

A Symbiotic Relationship

A program made up of both BASIC code and assembly language code takes advantage of both types of programming. BASIC can be used for all of the I/O bound processing, while assembly language can be used for the compute-bound processing, which must be as fast as possible.

An example of I/O bound processing is the portion of a mail list program that accepts user input for mail list entries. In this case, the high-speed processing of assembly language is unnecessary, as the data is being input at speeds no greater than 30 or 40 words per minute, and can easily be handled by BASIC code. A similar example is printing output, which proceeds at speeds in the area of 100 characters per second.

Assembly language code can be used in the compute bound, or "number crunching" portions of the program, where BASIC code is very slow. We've all seen or heard about mail list programs that take hours (or days) to sort entries. If the sort routines are coded in assembly language, this time critical portion of the program can be improved by a factor of 100 or so. Other examples of processing where assembly language can be used are high-speed graphics, searches, and array operations.

When one opts to use assembly language code, there are other trade-offs. BASIC code can be developed and debugged much faster than assembly language code. Numeric computations are very tedious in assembly language, especially when floating-point numbers (those larger than integer values) must be handled. String operations are not easily performed in assembly language.

However, in spite of its negative aspects programs of any significance can benefit greatly by having portions of the code in assembly language.

Level II BASIC and Disk BASIC allow BASIC code and assembly language code to be run together. The mechanism for this is the BASIC USR call. Let's illustrate the use of USR in Level II BASIC and Disk BASIC by some simple assembly language code. This code draws a line across the screen and is shown in Program Listing 1.

For reasons which we'll discuss later, the assembly language code here is relocatable. That is the machine code end result of the assembly (the hexadecimal data in the second column) can be moved anywhere in memory and will continue to draw that line on the screen. Non-relocatable code can only execute in one area in memory, the area for which it was assembled, controlled by the ORG pseudo-op in the assembly language statements. (For a more complete discussion of relocatability, see this column in the May, 1980 issue.).

Level II BASIC and Assembly Code USR Calls

Suppose that the 12 machine code bytes have been put into RAM memory at locations 8000H through 800BH. How do we execute the assembly language code at those locations while in the middle of a BASIC program? In Level II BASIC, the procedure is this:

1. POKE the address of the start of the machine language code into locations 16526 and 16527. The low order byte of the start address must be put into location 16526, and the high order byte must be put into location 16527. In this case 00H must be put into location 16526, and 80H must be put into location 16527. Locations 16526 and 16527 are simply a variable in BASIC that has been set aside specifically to hold the address of the machine code to be executed when a USR call is made.

2. At the point in the BASIC program at which you want to execute the "draw line" code, use the BASIC statement X = USR(M). When Level II BASIC encounters this statement, it takes the address value in locations 16526 and 16527 and calls that location. At the end of the machine language, a RET instruction pops the return address off the stack and causes a return to the BASIC interpreter, which picks up the BASIC code at the next statement after the USR call. This CALL and RETurn sequence is illustrated in Program Listing 2.

3. We've ignored variables X and M in the last two steps. If no arguments are to be passed, any dummy value can be used for M and variable X can be ignored. It's often necessary, however, to pass one or more arguments from the BASIC code to the machine language code and back again. For example, we might want to modify the assembly language routine to draw a line at any of the 16 screen lines, passing the line number of 1-16 from BASIC to the draw-line code. Conversely we might want to pass back to the result of a high-speed multiply routine to BASIC.

The USR call has a built-in provision for passing one argument to the machine code and for transferring one argument back from the machine code. The M in the statement X = USR(M) may be any variable or expression that resolves down to an integer value from −32768 to 32767. This argument is passed to the machine code in the HL register pair only if the assembly language code contains a CALL 0A7FH instruction. When the machine code executes the CALL, a ROM routine in Level II BASIC transfers the argument to HL.

To transfer an argument value of −32768 to 32767 back to BASIC, the value must be in HL and a JP made to ROM routine 0A9AH at the end of the machine language code; note that this "JP 0A9AH" replaces the normal RET.

Mulipie Arguments and Multiple Routines

Multiple arguments may be passed from the BASIC code to the machine language code in a variety of ways. The arguments may be at some predefined location known to both the BASIC program and the machine language code, for example a list starting at location 9000H. A second way to pass arguments is to pass a pointer in HL; this pointer identifies the argument list, which has a predefined order. The pointer may

```
100 'LEVEL II CALL AND RETURN SEQUENCE
110 CLS                    'CLEAR SCREEN
120 POKE 16526,0           'LS BYTE OF 8000H
130 POKE 16527,128         'MS BYTE OF 8000H
140 X=USR(0)               'CALL SUBROUTINE
150 GOTO 150               'LOOP HERE
```

Program Listing 2. Level II Machine-Language Calling Sequence.

```
               00100 ;ALL CODE HERE IS RELOCATABLE - NO ORG NEEDED
0000 0640      00110 DRAWL   LD      B,64            ;64 CHARACTER POSITIONS
0002 21003E    00120         LD      HL,3C00H+512    ;START
0005 3E8C      00130         LD      A,8CH           ;LINE
0007 77        00140 DRA010  LD      (HL),A          ;STORE SEGMENT
0008 23        00150         INC     HL              ;BUMP POINTER
0009 10FC      00160         DJNZ    DRA010          ;GO IF NOT 64
000B C9        00170         RET                     ;RETURN TO BASIC
0000           00180         END
00000 TOTAL ERRORS
```

Program Listing 1. Line Drawing Program

```
            ØØ1ØØ ;ALL CODE HERE IS RELOCATABLE - NO ORG NEEDED. THIS CODE
            ØØ11Ø ;ASSUMED TO BE AT 8ØØØH.
ØØØØ CD7FØA ØØ12Ø MULT2  CALL  ØA7FH      ;GET M INTO HL REGISTER
ØØØ3 29     ØØ13Ø        ADD   HL,HL      ;MULTIPLY BY 2
ØØØ4 C39AØA ØØ14Ø        JP    ØA9AH      ;PASS BACK AS X
ØØØØ        ØØ15Ø        END
ØØØØØ TOTAL ERRORS
```

Listing 3A

```
2ØØ 'LEVEL II BASIC HERE. ASSUMES MACHINE LANGUAGE
21Ø 'AT 8ØØØH.
22Ø POKE 16526,Ø                'LS BYTE OF 8ØØØH
23Ø POKE 16527,128              'MS BYTE OF 8ØØØH
24Ø DEFINT M                    'DEFINE M AS INTEGER
25Ø INPUT M                     'INPUT INTEGER VALUE
26Ø X=USR(M)                    'MULTIPLY BY 2
27Ø PRINT "X=";X                'PRINT RESULT
28Ø GOTO 25Ø                    'GO FOR NEXT
3ØØ 'DISK BASIC HERE. ASSUMES MACHINE LANGUAGE
31Ø 'AT 8ØØØH.
32Ø DEFINT M                    'DEFINE M AS INTEGER
33Ø DEFUSR5=&H8ØØØ
34Ø INPUT M                     'INPUT INTEGER VALUE
35Ø X=USR5(M)                   'MULTIPLY BY 2
36Ø PRINT "X=";X                'PRINT RESULT
37Ø GOTO 34Ø                    'GO FOR NEXT
```

Listing 3B

```
1ØØ 'DATA VALUES TO PREASSIGNED AREA METHOD
11Ø 'MOVE DATA HERE
12Ø FOR X=32768 TO 32774
13Ø READ A
14Ø POKE X-65536,A
15Ø NEXT X
16Ø DATA 2Ø5,127,1Ø,41,195,154,1Ø
2ØØ 'LEVEL II BASIC HERE
21Ø POKE 16526,Ø
22Ø POKE 16527,128
23Ø DEFINT M
24Ø INPUT M
25Ø X=USR(M)
26Ø PRINT "X=";X
27Ø GOTO 24Ø
3ØØ 'DISK BASIC HERE
31Ø DEFINT M
32Ø DEFUSR5=&H8ØØØ
33Ø INPUT M
34Ø X=USR5(M)
35Ø PRINT "X=";X
36Ø GOTO 33Ø
```

Program Listing 4. DATA Values to Preassigned Area Example.

be one established by the BASIC VARPTR command, and the pointer may point to an array or string containing multiple arguments. Another method is to pack arguments into HL. Four four-bit values could be put into HL, or one eight-bit argument could be in H and another in L.

More than one assembly language routine can be used with a Level II BASIC program; it's only necessary to put the starting address of the machine language routine to be called into 16526,7 before each new USR call is executed in BASIC. If there is only one machine language routine, its starting address needs to be placed into 16526, one time only, at the start of the BASIC program.

The calling sequence for Disk BASIC is somewhat more straightforward. Disk BASIC also uses the USR function, but it has a slightly different format. The format is X = USRn(M). The small letter n stands for any digit 0 through 9, which references one of ten possible machine code addresses. To call machine code routine 7, for example, the statement might be X = USR7(M). As in Level II BASIC, M may be used to pass one 16-bit argument, or it may be a dummy and X may be used to pass back one 16-bit argument, or it may be ignored.

Rather than POKEing the address of each machine language routine into 16526,7, Disk BASIC permits the addresses to be predefined by a new function, DEFUSRn. To define machine code routine 8 at 8000H, the statement DEFUSR8 = &H8000 is executed sometime before the call X = USR8(M) is made. Everything else works as in Level II BASIC. CALL 0A7FH is optionally made to pick up the M argument from BASIC, and a JP 0A9AH is optionally made to pass an argument in HL back to BASIC as variable X.

Another sample program with versions for both Level II BASIC and Disk BASIC is shown in Program Listing 3. This program takes one argument M from BASIC, multiplies it by two in the machine language code, and passes it back to BASIC as variable X. It illustrates all of the concepts we've been talking about except for multiple argument passing.

There are basically two methods for interfacing machine language programs to BASIC programs. The first is the two program approach. The second is the embedded machine language approach. We'll discuss the techniques and advantages of each.

The Two Program Method

In the two program approach, the BASIC program contains no machine language code. The machine language code is separately assembled and loaded. To integrate the BASIC program and the associated assembly language program for a cassette based system, the following steps are performed:

1. Execute Level II BASIC. For MEMORY SIZE?, type in a memory address corresponding to the start of the assembly language program. Normally this would be in high memory, as everything above this address will not be used by the Level II BASIC interpreter.

```
1ØØ 'DUMMY STRING USING CHR$
11Ø A$=CHR$(2Ø5)+CHR$(127)+CHR$(1Ø)+CHR$(41)+CHR$(195)+
    CHR$(154)+CHR$(1Ø)
2ØØ 'LEVEL II BASIC HERE
21Ø DEFINT M
22Ø B=VARPTR(A$)
23Ø POKE 16526,PEEK(B+1)
24Ø POKE 16527,PEEK(B+2)
25Ø INPUT M
26Ø X=USR(M)
27Ø PRINT "X=";X
28Ø GOTO 22Ø
3ØØ 'DISK BASIC HERE
31Ø DEFINT M
32Ø B=PEEK(VARPTR(A$)+1)+PEEK(VARPTR(A$)+2)*256
33Ø IF B>32767 THEN DEFUSR5=B-65536 ELSE DEFUSR5=B
34Ø INPUT M
35Ø X=USR5(M)
36Ø PRINT "X=";X
37Ø GOTO 32Ø
```

Program Listing 5. Dummy String Using CHR$ Example.

```
100 'DATA VALUE TO DUMMY STRING
110 A$="DUMMY S"
120 B=PEEK(VARPTR(A$)+1)+PEEK(VARPTR(A$)+2)*256
130 IF B>32767 THEN B=B-65536
140 IF B>32767 THEN C=B+6-65536 ELSE C=B+6
145 IF B>32767 THEN D=-1 ELSE D=1
150 FOR X=B TO C STEP D
160 READ A
170 POKE X,A
180 NEXT X
190 DATA 205,127,10,41,195,154,10
200 'LEVEL II BASIC HERE
210 DEFINT M
220 B=VARPTR(A$)
230 POKE 16526,PEEK(B+1)
240 POKE 16527,PEEK(B+2)
250 INPUT M
260 X=USR(M)
270 PRINT "X=";X
280 GOTO 250
300 'DISK BASIC HERE
310 DEFINT M
320 B=PEEK(VARPTR(A$)+1)+PEEK(VARPTR(A$)+2)*256
330 IF B>32767 THEN DEFUSR5=B-65536 ELSE DEFUSR5=B
340 INPUT M
350 X=USR5(M)
360 PRINT "X=";X
370 GOTO 340
```

Program Listing 6. DATA Value to Dummy String Example.

```
100 'DATA VALUE TO ARRAY
110 DIM A%(6)
120 FOR X=0 TO 3
130 READ B,C
135 A=B+C*256
140 IF A>32767 THEN A%(X)=A-65536 ELSE A%(X)=A
150 NEXT X
160 DATA 205,127,10,41,195,154,10,-1
200 'LEVEL II BASIC HERE
210 DEFINT M
220 B=VARPTR(A%(0))
230 POKE 16526,B-INT(B/256)*256
240 POKE 16527,INT(B/256)
250 INPUT M
260 X=USR(M)
270 PRINT "X=";X
280 GOTO 250
300 'DISK BASIC HERE
310 DEFINT M
320 B=VARPTR(A%(0))
330 IF B>32767 THEN DEFUSR5=B-65536 ELSE DEFUSR5=B
340 INPUT M
350 X=USR5(M)
360 PRINT "X=";X
370 GOTO 340
```

Program Listing 7. DATA Value to Array Example.

2. Use the SYSTEM command to load a previously assembled machine language program into the protected memory area. Rather than typing *slash* to start executing the machine language code, hit BREAK to go back to the BASIC interpreter.

3. Execute the BASIC program with its USR calls to the machine language code. The machine language code loaded by the SYSTEM tape may contain many different routines, each of which can be specified by the proper POKE commands to 16526,7 and a subsequent USR call.

To utilize the two program approach with a disk based system, follow these steps:

1. LOAD the machine language program into memory. The program must have been previously assembled with an object file to disk (Apparat assembler) or have been previously loaded and saved on disk (Radio Shack Macro Disk Assembler).

2. Execute Disk BASIC by the usual BASIC input. Protect the machine language code by entering the starting address of the machine language code for "MEMORY SIZE".

3. Execute the BASIC program with its DEFUSRn and USRn functions.

This method works well, but is best used for lengthy assembly language programs that cannot use the procedures for embedded code that we'll outline shortly. Since it is a two step loading process, it is somewhat awkward to use. Its main disadvantage is that the BASIC code must deal with absolute addresses; if the assembly language portion is reassembled, the addresses of its routines may change, and the corresponding addresses in the BASIC program will also have to be changed.

The second method of interfacing is the embedded machine code approach. This approach is best used when the machine code consists of short routines. The data representing the machine code instructions can be incorporated into the BASIC program directly, as DATA values, CHR$ strings, or dummy strings. The advantage of this approach is that the BASIC program, complete with its embedded machine code, can be loaded or edited in one fell swoop.

The disadvantage is that some conversion is involved, to change the machine code from an assembly listing to corresponding decimal DATA or string values. There are a number of ways to embed the machine code; we'll discuss the most popular.

DATA Values to Preassigned Area

This is similar to the two program method in that the machine code is at a predefined area and should be protected by a MEMORY SIZE value. In this approach, the machine code in a DATA table is moved to its area sometime prior to the USR call. Since the area is predefined, the machine code does not have to be relocatable, but simply assembled to run at the specified memory area. An example for both types of BASIC is shown in Program Listing 4.

A dummy string consisting of CHR$ values can also hold the machine code. The location of the string is found by the VARPTR function. As one reader pointed out, this appoach has a potential problem. When the infamous "string garbage collection" mode is entered in BASIC, the location of strings may change. The garbage collection mode is entered when all string space has been used, and the interpreter must go back and clean up the string allocation area to create additional room. (Garbage collection is used in lengthy BASIC programs that manipulate many strings.)

When using this method, find the VARPTR location immediately prior to the USR call; it will work without problems except a 255-byte limitation on string size. Relocatable code must be used. This approach is shown in Program Listing 5.

DATA Value to Dummy String

```
8000                    ORG     8000H
          ;FAST MULTIPLY BY STEVE NELICK, WASHINGTON, DC.
8000 58        MUL      LD      E,B           ;MULTIPLICAND
8001 1600               LD      D,0           ;NOW IN DE
8003 62                 LD      H,D           ;ZERO HL
8004 6A                 LD      L,D
8005 07                 RLCA                  ;ROTATE M'IER BIT TO CY
8006 3001               JR      NC,$+3        ;GO IF M'IER BIT = 0
8008 19                 ADD     HL,DE         ;ADD M'CAND TO TOTAL
8009 29                 ADD     HL,HL         ;SHIFT RESULT ONE BIT POS LEFT
800A 07                 RLCA                  ;SHIFT M'IER BIT
800B 3001               JR      NC,$+3        ;GO IF M'IER BIT = 0
800D 19                 ADD     HL,DE         ;ADD M'CAND TO TOTAL
800E 29                 ADD     HL,HL
800F 07                 RLCA
8010 3001               JR      NC,$+3
8012 19                 ADD     HL,DE
8013 29                 ADD     HL,HL
8014 07                 RLCA
8015 3001               JR      NC,$+3
8017 19                 ADD     HL,DE
8018 29                 ADD     HL,HL
8019 07                 RLCA
801A 3001               JR      NC,$+3
801C 19                 ADD     HL,DE
801D 29                 ADD     HL,HL
801E 07                 RLCA
801F 3001               JR      NC,$+3
8021 19                 ADD     HL,DE
8022 29                 ADD     HL,HL
8023 07                 RLCA
8024 3001               JR      NC,$+3
8026 19                 ADD     HL,DE
8027 29                 ADD     HL,HL
8028 07                 RLCA
8029 D0                 RET     NC
802A 19                 ADD     HL,DE
802B C9                 RET
0000                    END
00000 TOTAL ERRORS
```

Program Listing 8. Fast Multiply Winner.

This approach avoids the garbage collection problem by moving DATA values to a dummy string made up initially by legitimate string values. The location of this type of string is always within the statement itself, rather than being in the separate string area. (You may want to verify this by some experimentation with the VARPTR.) The size of the string must be at least equal to the number of machine code bytes to be stored. The 255 byte maximum size limitation remains. Relocatable code must be used. This approach is shown in Program Listing 6.

DATA Value to Array

A final method was suggested by Charley Butler of the Alternate Source. In this case, the DATA values are moved to an array. As arrays are allocated once and are not reshuffled, there is no garbage collection problem. Also, the arrays may be as large as is necessary, and any size machine language program may be used. Relocatable code must be used as the array location is not known beforehand. This approach is shown Program Listing 7.

The Great Second Assembly Line Programming Contest

There was an excellent response to the second contest problem posed in this column in June: Write the fastest subroutine possible to multiply two eight-bit unsigned numbers in the A and B registers with the result to appear in HL with A and B preserved.

Entries were in four categories; those that used huge tables to precompute the result, those that used repetitive addition (too slow), those that used standard eight-iteration loops (some very elegant), and those that wrote in-line code to repeat the computations eight times, avoiding the loop overhead.

As the decision of this blear-eyed judge is final, the winner belongs in the last group, having the fastest worst-case multiply that did not use extensive pre-computed tables in memory. He is Steve Nelick of Washington, DC and his fast 157.85 microsecond multiply is shown in Program Listing 8. He will shortly receive a copy of the new computer science bestseller *Godel, Escher, Bach* by Hofstadter and a copy of my new Radio Shack fiction book *Programming Techniques for Level II BASIC*.

Honorable mention goes to Foulk, Van Pelt, Mignery, Wallen, Lee, Craig, Thomas, and Smith.

I appreciate all of the entries. Some of you went to elaborate documentation and I am in awe of the amount of work you performed.

Many have asked me to keep the problems coming, so I am duty bound to pose a third problem: Write an assembly language subroutine to draw a line between any two screen character positions. (Not points! There are 16 lines of 64 character positions.) There will be two winners, one for the most elegant version, and one for the fastest version. Again, token prizes will be awarded. Send entries to the address at the end of this column.

Next month we'll discuss the mysteries of the Radio Shack Disk Assembler and review some assemblers for the Model II. ■

William Barden, Jr.
28182 Palmada
Mission Viego, CA 92692

80 INPUTS

from page 16

Double Size Graphics

In Bertram Thiel's article, "Double Size Graphics" which (June *80*), states that the only way to escape the double character mode is with a CLS, or as he goes on to describe later, a series of PEEKs. Actually, he is in error, the TRS-80 provides an escape from this mode via the CHR$(28), which returns the cursor to home, and resets the screen back to 64 characters. It is the exact opposite of CHR$(23) and can be used as such.

Jeff Eisen
Huntington Valley, PA

Super Program

I would like to commend Delmer Hinrichs for his super BASIC Word Processor program that appeared in the May issue of *80 Microcomputing*.

The program runs perfectly with just minor changes to accommodate my printer and provide for a stop between printed pages so that single sheets can be used. The changes I made are:

1. Use LPRINT" " at the end of line 1690
2. Add line 1675 to provide top margin of one inch on single sheets and paper advance on rolled paper.
   1675 FORZ1 = 1TO6:LPRINT CHR$(138):NEXTZ1
3. Delete LPRINT CHR$(28) and GOTO1680 from line 1740
4. Add new line 1745:
   1745 INPUT"NEW PAGE—PRESS ENTER WHEN READY";Z1:GOTO1675

There is one apparent misprint in line 880 (a lowercase 'P"). I'm guessing, but I think that the misprint should be replaced with a zero.

Morris L. Krome
Owings Mills, MD

Recorder Variation

For the past three years Grant Union High School in Sacramento California has been developing a TRS-80 based microcomputer program as part of our math and science curriculum. During this period we have continued to acquire systems and build an extensive library of programs on tape and disk. Many of our programs were CSAVEd on the CTR-41 recorder. Most of our recorders are now CTR-80s.

We developed an equation to convert reference numbers from one tape recorder to another. We checked our results on several different tape recorders and found the maximum variation to be one to two percent.

The equation relating the two systems is as follows:

CTR-41 (#) = (1.66)(CTR-80(#))
CTR-80 (#) = (.6)(CTR-41(#))

Madeleine Fish (Teacher)
Steven Emert (Student)
Sacramento, CA

Word Processing HMCT Style

Using the best in word processing software **the Magic Wand™**, the best in letter quality printers **the NEC Spinwriter™**, and the well-known **TRS-80™ MOD II.** You have looked and looked for the best and now it is available from the **Complete Computer Company, HMCT.** Word Processing at its best from $7500.

Magic Wand available separately for the MOD II and other computers $400.

HMCT The Complete Computer Company
5313 Bissonnet Bellaire, Texas 77401 (713) 661-2005 ↙33

Magic Wand is a trademark of Small Business Applications, Spinwriter is a trademark of NEC, and TRS-80 is a trademark of Radio Shack.

"The two existing and complex sets of standards have led to classification chaos of the buyer."

COBOL on the 80: Prospects & Pitfalls

If the item about Tandy's new COBOL (COmmon Business Oriented Language) compiler didn't catch your attention in "80 News" last month, take another look. And take a peek at June's issue where you'll find the Nevada COBOL compiler. Then there's May with Microsoft's COBOL announcement.

Chairman of the American National Standards Institute (ANSI) COBOL Committee James Panttaja explains that the introduction of COBOL for microcomputers "may put micros in a different market place, taking them beyond the home, where they started their existence, into the business world."

One of the first things anybody hears about COBOL is that there are hundreds of thousands of business programs written in the language and hundreds of thousands of programmers trained in COBOL. Like Panttaja, many people believe that the TRS-80 compilers, which convert COBOL to machine code, will open the door for 80 owners to the wealth of tried and true COBOL business programs.

COBOL was first introduced to the market in 1959—about one and a half decades before the advent of microcomputers. The language was designed to be self-documenting. It is written in full English words, in statements and paragraph-like formation. The idea was that a group of people could develop programs jointly, with each program continually stating its objectives and progress in plain English.

Although COBOL originated from an effort to organize national libraries, Microsoft's Mike Orr explains that, "Over the years, COBOL has been tuned to be just what people want it to be." It began with good text movement capabilities and now exceeds BASIC and word processors in screen formatting, generating rows and columns of data, report generation, Indexed Sequential Access Method (ISAM) filing and graphs.

But because of its vocabulary and syntax, COBOL is a bulky language that can take up a great deal of memory. And it is a slow process to develop COBOL programs. Programming in BASIC is both faster and easier. Yet, BASIC cannot be applied to business applications as thoroughly.

The situation is not simple. In its short history, BASIC has already been written in many versions. There are BASICs for every level of the TRS-80, the Apple and the Pet. There are BASICs for disk operating systems and others for cassette systems. In the long history of COBOL, it is hard to say how many versions of the language exist. Because of these differences in COBOL, programs will need to be translated, not only from machine to machine, but from one version of COBOL to another, to operate on the 80.

Languages dubbed COBOL range from scant, specific application languages to gargantuan languages that can operate in 12 modules serving countless applications. The price range of the three COBOLs with compilers that have been listed in "80 News" is indicative: Business Microproducts' Nevada COBOL for TRS-80 Models I and II with CP/M costs $99. Tandy's TRSDOS COBOL for the Model II costs $299. Microsoft's TRSDOS COBOL-80 compiler costs $750.

"Programs will need to be translated, not only from machine to machine, but from one... COBOL to another."

In an effort to limit the confusion caused by the proliferation of COBOLs, ANSI established the first COBOL standards in 1968. They were updated in 1974, and a corollary set, the Federal Information Processing Standards (FIPS), was issued in 1978.

Deciphering COBOL types and standards is undoubtedly more difficult than programming in COBOL. The importance of this standards maze is its impact on the translation of the thousands of COBOL business programs already in existence.

The ANSI-74 standards divide COBOL into 12 possible modules and require only sufficient performance in the first three to classify a language as COBOL. These three modules are termed Nucleus, Table Handling and Sequential I/O. The complete ANSI standards are explained in the ANSI X3.23 manual. The 1978 FIPS standards separate COBOL into four classes: low; low intermediate; intermediate, and high.

ANSI's Panttaja asserts, "If the levels of FIPS or the modules of ANSI-74 are the same, very little rewrite is necessary to write a comprehensive and comprehensible translation from any machine to another." Without the standards, COBOL translation can be just as difficult as any other machine translation.

The two existing and complex sets of standards have led to classification chaos for the buyer. Further, any distributor of COBOL will be quick to tell you that the standards do not account for non-standard extensions of their language. There are also factors of speed and run time programs to be considered.

When and if you get the standards sorted out, there's still a hitch. As Panttaja explains, "There's no means of verifying ANSI standards." And although the government, through FIPS, has devised tests to validate COBOL language compilers, they are administered by manufacturers. There is no government grading or verification of the test results.

New ANSI standards are being developed now and are expected to be released in the next few years. The *COBOL Information Bulletin*, citing what can be expected of the future standards, is available from the ANSI COBOL Committee, 1430 Broadway, New York, NY 10018.

Despite the lack of software, both Microsoft, Bellevue, WA, and Business Microproducts, Livermore, CA (who have had their separate TRS-80 COBOL compilers on the market for several months), are satisfied with their sales. Business Microproducts' Jim Smith says that they have found a market among high schools "and even universities" that are teaching COBOL programming. Mike Orr of Microsoft explains that despite a low advertising take off, their $750 compiler has "sold in the dozens in its first two months on the market."

But unless the COBOL standards are quickly clarified, it's doubtful that the advent of COBOL for the TRS-80 will have the immediate and dramatic impact on the business market that many people are expecting. Those thousands of existing business programs can't be run on the TRS-80—yet. ■

By Nancy Robertson
80 Staff

Confused by the New Level II Two-chip ROM?

Know what? Radio Shack is selling a revised two-chip version of ROM for Level II BASIC machines. These new 6K chips have given some users problems because the new ROM does not function in quite the same manner as the old.

In their monthly newsletter, *Microcomputer News*, Radio Shack ran a list of command corrections for the new ROM, and some other interesting tidbits. They are rehashed here in an effort to clarify the confusing situation which has resulted from the introduction of this new two-chip ROM.

First, the new power-up query is: MEM SIZE? The BASIC sign-on prompt is: R/S L2 BASIC. The memory space saved by condensing these messages is now devoted to a keybounce fix. The old KBFIX keyboard driver is no longer necessary and will not work with the new ROM.

Second, free memory space has been reduced by two bytes resulting in a 15570 answer to the MEM? query in a 16K machine.

Third, one can no longer CLOAD from either tape input when an expansion interface is bing used. Only tape drive number one may be used. CSAVE#-N procedures still allow the operator to select the destination drive in multi-recorder set-ups, however.

Fourth, pressing SHIFT and the down arrow simultaneously will result in control characters. You are in the control mode as long as both keys are depressed.

Fifth, you can now perform multiple PRINT@ operations by simply constructing your print statements as follows:

```
PRINT@10,"A",@50,"B",@2000,"c"
```

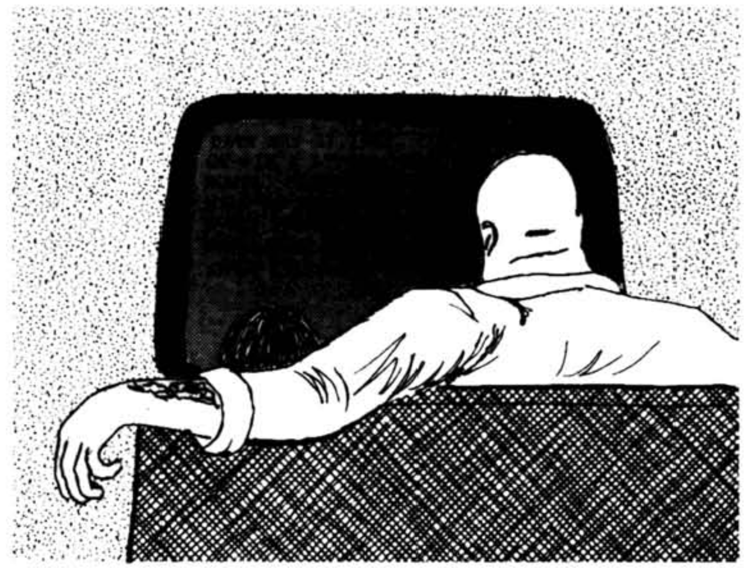
In addition, about ten of the Shack's tapes will not CLOAD with the new ROM. If you encounter this situation, a new tape will be supplied by Radio Shack.

This ROM is not interchangeable with previous ROMs. If you want it, be prepared to shell out $150 (Level II) for a complete conversion kit. The kit is only available as an installed modification and, as one might expect, only certified Radio Shack service centers will be doing the installations. Installation is extra by the way, but how much extra is not mentioned. The model number for the new Level II ROM is #26-1120, and it is available now. ■

By Chris Brown
80 Staff

Please Be Patient

At 80 Microcomputing *we are undergoing some staff changes that have resulted in a backlog of mail and manuscripts. We are reading manuscripts as quickly as possible. The editors apologize for any delay.*



The Infancy of Electronic News Delivery

July 1, *The Columbus Dispatch* became the first newspaper in the country to offer electronic home delivery. Owners of microcomputers can now dial their phones, connect their modems and read the evening news from their video screen.

The Dispatch is one of 11 Associated Press (AP) newspapers which will be offering electronic home delivery through the CompuServe, Inc., Columbus, OH, data network. The other papers which will participate in electronic delivery are: *The Washington Post; The New York Times; Chicago Suntimes; The St. Louis Post-Dispatch; The Minneapolis Star and Tribune; The Norfolk Virginian-Pilot and Ledger-Star; San Francisco Chronicle* and *The Middlesex News* (Framingham, MA).

CompuServe's president, Jeffrey Wilkins, explains that the 11 papers were chosen from more than 40 that were interested. Several of the publications are expected to join *The Dispatch* in electronic delivery sometime in September.

July 14 Knight-Ridder Newspapers, Inc. also began electronic delivery of their paper on an experimental basis in 30 homes. Calling the project Viewdata, Knight-Ridder is working in conjunction with American Telephone and Telegraph.

However, because microcomputers do not need to be dedicated to data sharing only, the prospects for the future of electronic delivery seem more likely to pivot on the reception of the CompuServe program. There are 1,300 daily papers and 3,500 radio and television stations who own AP. Most of them are curious: Does the public have any interest in microcomputer, electronic delivery of the news?

As it stands now, electronic delivery through CompuServe begins with a local phone call from the subscriber, answered by a local computer, which in turn communicates with CompuServe's data base in Columbus. Via telephone wires and modems, news enters the home microcomputer at 300 words per minute. A directory or menu allows users to scan news sections and headlines. Stories can be called, read, saved or printed from each section.

However, computer subscribers will have to say good-bye to Mary Worth, Steve Canyon and Charlie Brown. The comics, political cartoons and photographs will not be transmitted.

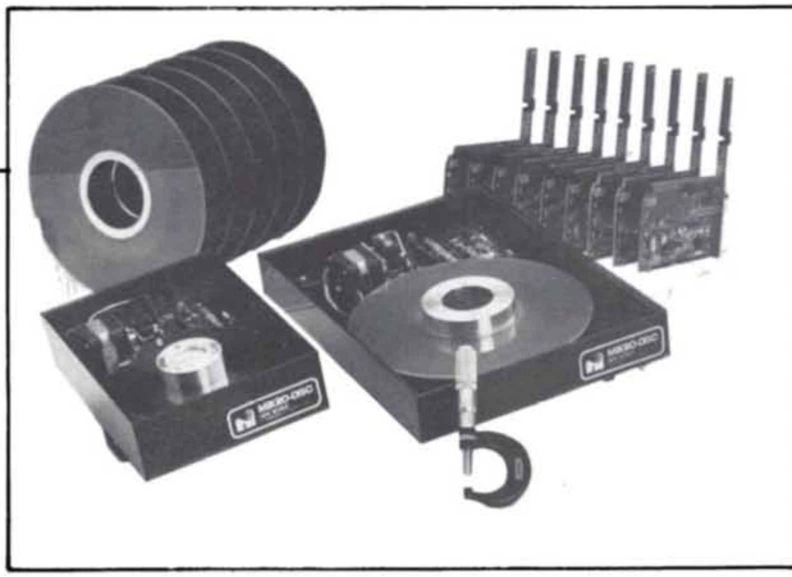
For the time being, advertising is also absent from computer editions. Advertising generally accounts for about two-thirds of newspaper revenues. Although electronic delivery may prove less expensive than paper printing, it is argued by many people in the business that new concepts in advertising will be vital to the survival of computer news delivery.

The Dispatch has already made plans to incorporate classified ads in their CompuServe edition. They are also working on an advertising index system that will make it possible for space advertisers to sponsor several articles.

Philip Meyer, marketing director of Knight-Ridder's Viewdata project, does not believe computer delivery will completely replace paper delivery. "The newspaper is not an inefficient information retrieval device," he said. "You can scan it very quickly. The main threat to newspapers is not competition for information, but competition it (data sharing) provides for advertisers."

Publishers and advertisers may adjust to electronic delivery of the news, but what will become of Sundays without the Sunday comics? ■

NEW PRODUCTS



Mikro-disc Disk Drives

Utilities for Businesses

ACCT-M2 from Micro Architect, Inc. consists of five programs that carry out the on-line accounts receivable functions of a small business or a medical clinic.

The three basic functions are initialization, data base management and report generation. Initialization allows users to specify the system parameters, such as company name and address, late charge policy, etc. It also can sort the customer names or change most of the system parameters. The data base management lets users add, inquire, delete and adjust transactions and customer information.

Reports consist of sales journal, receipts journal, aging analysis, end of period processing, data base listing, labels, statements and deleted customer reports. A consistency check is included.

ACCT-M2 is priced at $149, including a disk and user's manual. The user's manual is available separately for $5. A simplified version of the program is available for the Model I and costs $69.

L216 is another business package from Micro Architect for systems with at least 16K and Level II BASIC. It consists of the following programs: a cassette data base manager, a word processor, an inventory control system, a stock management program, a check balancing program, a label printer, a deposit calculator, a statistics program, a sort utility and a key-access utility.

The complete package costs $59.

For more information on either of these products, contact Micro Architect Inc., 96 Dothan St., Arlington, MA 02174.

Reader Service ✓ 174

Color Graphics Circuit Board

Integrated Service Systems, 1011 W. Broadway, Minneapolis, MN, 55411 is selling a TRS-80 color graphics circuit board. It is designed for use with a standard color television receiver and a 4K to 48K Level II. The board is sold in two different packages: Model C-2000 and Model C-1000. Both include all electronic components. Minor assembly is required to complete the RF modulator section.

Models C-2000 and C-1000 are fully buffered and expanded to ten display modes. ASCII and eight-color semi-graphics can be mixed on the screen. Five four-color graphics modes and four two-color graphics modes are included.

COLPRT is a machine language program that allows ASCII and semi-graphics information to be displayed by LPRINT. COLSCREEN is the program which transfers the TRS-80 graphics to a color screen. (COLSCREEN is not available yet, but is expected to be on the market by October.)

Model C-2000 includes the circuit board, COLPRT and power supply for $189.95. Model C-1000 costs $129.95 for the circuit board and software without the power supply.

Reader Service ✓ 333

80 Music Synthesizer

Software Affair, Ltd. is selling Orchestra-80, a TRS-80 music synthesis system written for a 16K Level II TRS-80. The system consists of software and hardware.

The software is a five part machine language program: a digital synthesizer which produces four simultaneous voices in a six-octave range; a music language compiler, a full function text editor with blinking cursor, and a file manager.

The hardware is a single 1½ by two inch PC board, completely assembled and tested, which plugs into the expansion connector of the TRS-80 keyboard or the screen printer connector on the expansion interface.

Tape and disk versions are supplied on cassette, with sample music programs. For more information, contact Software Affair, Ltd., 473 Sapena Court, Suite 1, Santa Clara, CA 95051.

Reader Service ✓ 340

Production to Begin on New Winchester Drives

New World Computer Company, Inc., 3176 Pullman St., Suite 120, Costa Mesa, CA 92626, has added the Minimikro-disc V Series of 5¼-inch fixed disk drives and an enhanced double-density system called Mikro-disc VIII-1TF to its product line.

According to the company, standard 5¼-inch floppy disk drives have an average access time of 298 milliseconds. The Minimikro-disc V-ITF can access data in an average of eight milliseconds.

The original Mikro-disc 211, has doubled storage capacity and has been renamed the Mikro-disc VIII-ITF.

The Minimikro-disc V-1TF and Mikro-disc VIII-1TF are priced at $700 and $1100 respectively, in dealer quantities. Production is scheduled to begin by the end of the year.

Reader Service ✓ 328

VTOS 4.0 Plus Manual

Virtual Technology, Inc., Dallas, TX 75234, has a new disk operating system available, the VTOS 4.0.

VTOS 4.0 will support five inch (35, 40 or 77 tracks), eight inch (single or dual density), or ten M hard disks. Speed-up kits that are on the market will also be supported. Backup is simplified so that users only need to indicate the type of drives used. Improved chaining offers 15 chaining commands that can handle most

I'M A BELIEVER !!

"I Love it !!. . .It's really a incredible O/S. It' just great! Now I see why people who have seen it say they are now believers. I know I am." *LANCE MICKLUS*

1) Large (8") drive support.
2) Double Sided drive support.
3) Double Density drive support.
4) 80 Track drive support.
*NOTE all above drives may be mixed on any one system and can be configured at Sysgen time or during any Backup!
5) Winchester technology fixed drive support.
6) Supports any combination of the above drives up to a max. of 8 drives.
7) Supports double-speed processor clock modifications. (Archbold for example)
8) FASTER! —— Improved overlay structure using ISAM accessing techniques improves loading times by up to 1400%.
9) General purpose output spoolers of a true, symbiont design provide simultaneous output and program execution without any user intervention.
10) Keyboard Type-Ahead feature permits you to enter keystrokes before your programs need them.
11) User definable keys, all 26 letters.
12) Built in Graphic string packer lets you enter graphic symbols into a BASIC program from the keyboard through the use of the (Clear) key. The (Clear) key is simply held down (just like the (Shift) keys) during other keystrokes and viola...graphics!
13) Dated files. — All files are accompanied by the date of their last modification (creation or write).
14) Marked files. — All files are accompanied by a 'mark' if they have been modified since they were last backed up. This permits the BACKUP utility to copy only those files which have actually been updated since a previous backup.
15) File transfer by class. Allows transferring of all files of a similar directory classification such as /CMD, /BAS, /PCL, etc.

VTOS 4.Ø

VTOS 4.Ø

Operating System Diskette with Operator's Guide

$99.95

VTOS 4.Ø

Master Reference Manual

$29.95

VTOS 4.Ø

Combination – 4.0 disk, Operator's Guide, and Master Reference Manual

$125.00

16) Built-in SYSTEM command contains lower case display driver, screen print, break key disable, blinking cursor, disk drive stepping rate and motor-on delay modifications, and more.
17) User may SYSGEN a custom VTOS system configuration containing special I/O drivers, device LINKing and ROUTEing, SPOOLing and DEBUG tasks, etc. which will be automatically loaded during the BOOT process without requiring a more lengthy AUTO and CHAIN procedure.
18) Non-BREAKable AUTO and CHAIN commands.
19) Wild-card DIRectory. Permits you to locate all files of a certain classification such as '/BAS'. Uniformly indicates file size in K (1024 bytes) regardless of drive type. "DIR D" would give you all your files that start with "D".
20) Dynamic file name defaults in APPEND, COPY, and RENAME commands allow you to specify only minimal information about file names.
21) COPY and APPEND commands execute up to 300% faster.
22) ALLOCate command for pre-allocation and non-releasibility of file space. File space will never shrink if this option used.
23) MEMORY command for directly setting upper memory limit.
24) Variable Length file support is incorporated which automatically blocks short user data records both within a sector and across sector boundaries thereby taking maximum advantage of disk file space.
25) No security disk needed to make backups or to run the system!
26) Though many O/S bear his design and code VTOS 4.0 is the only Fully Aproved Operating System by Randy Cook! And it is FANTASTIC!
27) Endorsed by Scott Adams and Lance Micklus!

VTOS and VTOS 4.0 are registered trademarks of VIRTUAL TECHNOLOGY, INC. - Dallas, Texas 75234

Available from the following distributors or your local computer store. *DEALER INQUIRIES INVITED.*

5% Discount Just For Mentioning This Ad. (Valid month of this publication ONLY)

QS QUALITY SOFTWARE DISTRIBUTORS
11234 Park Central Pl Suite C
Dallas Texas 75230
(214) 692-1055
Micronet - 70130,203
SOURCE - TCC293

Adventure INTERNATIONAL ✓97
ADVENTURE INTERNATIONAL
Box 3435, Longwood, Fla. 32750
(305) 862-6917 - Voice
after 8:00 - same number
as FORUM 80. (SOURCE - TCC957)

SMALL BUSINESS SYSTEMS GROUP
Main St. and Lowell Road
Dunstable, Mass. 01827
(617) 692-3800 - Voice
(617) 692-3973 - FORUM 80
Micronet - 70310,236

Prices subject to change without notice.

types of routines, including timed functions.

There is a built-in graphics screen packer. Graphics can be designed on the screen, listed and combined with BASIC LOOK-AHEAD.

Printer and keyboard speed buffers make it possible to use the system for two functions at one time. For instance, it is possible to operate the printer and run a program simultaneously.

The *VTOS 4.0 Master Reference Manual* claims to offer "the detailed handholding information that so many people have been asking for."

Together, the manual and VTOS 4.0 are sold for $125. The manual is available separately for $29.95.

Reader's Service ✓ 330

Speed Loading Cassettes

Personal Micro Computers, Inc. (PMC), 475 Ellis St., Mt. View, CA 94043, is selling Fastload to input cassettes to Level II computers at one kilobyte per second. PMC claims this is 16 times the normal speed. Any cassette saved at normal speed (500 baud) can be loaded with Fastload at 800 baud.

Unlike other high speed program loading devices, Fastload does not require transferring programs to another media first. This PMC product is also able to search for BASIC programs at high speed, but it does not provide for high speed recording of cassettes.

The Fastload cassette interface plugs into either the back of the TRS-80 Level II 16K keyboard or expansion interface. PROM memory is located above ROM and below RAM. It also contains a keyboard debounce program.

A CTR-41 tape recorder must be used with Fastload. Modifications must be made on standard CTR-41s with instructions provided; a new recorder with the modifications is sold by PMC for $95. The Fastload cassette interface is sold for $188.

Reader Service ✓ 349

Model II System Monitor

RSMII is a relocatable system monitor for the Model II sold by Small System Software, P.O. Box 366, Newbury Park, CA 91320.

RSMII includes commands to insert breakpoints, dump memory in hex and ASCII, test, search, modify, verify, zero and fill memory, etc. These features are also included in the company's Model I system monitors.

Features that are new with the RSMII include a video editor to modify both memory and disk sectors. Disk commands can access four drives and can read and write single and double-density disks. There are controls for split screen scrolling and adjustable scroll speeds.

RSMII will operate printers through either parallel or serial printer ports at baud rates from 110 to 9600. An image of the screen may be printed at any time. Page-length control and RS-232-C handshaking are supported.

The system monitor comes on a self-booting disk with a relocater. It is sold for $39.95 with an instruction manual.

Reader Service ✓ 167

General Ledger System

Sturdivant & Dunn, Inc., 124 Washington St., Box 277, Conway, NH 03818 is selling Full Charge Bookkeeper, an advanced general ledger system for small businesses.

It gives users an efficient method of posting income, expense and adjustments to obtain financial statements plus cumulative detail on an annual basis. It can handle 12 characters of alpha and 4520 posted entries in up to five departments.

Full Charge Bookkeeper is written for a TRS-80 Level II with 48K, three disk drives and printer. The package, which includes manual and programs is priced at $199.95. It will be sold for $129.95 through September as an introductory offer.

Reader Service ✓ 181

Fastload Interface and Cassette

Radio Shack's Self-teaching Guide to Programming

RS Game and Guide

A new game from Radio Shack, Dancing Demon, displays an animated figure dancing to music that can be heard by connecting the computer to an amplifier and speaker. The program is supplied on cassette and includes two preprogrammed selections. The user can also program other tunes of up to 248 notes and choreograph dance routines to go with the music.

The program is designed for a Model I Level II TRS-80 with at least 16K RAM. It is priced at $9.95.

Radio Shack also has a compact 200 mW speaker-amplifier that can be used to produce the musical accompaniment for Dancing Demon. The Realistic Micro-Sonic Speaker-Amplifier is priced at $11.95.

TRS-80 Level II BASIC, a self-teaching guide to programming and using a Level II microcomputer, is another new product from Radio Shack. Each chapter is composed of short, numbered sections which present ideas or topics on BASIC, the TRS-80, or a program that is being developed.

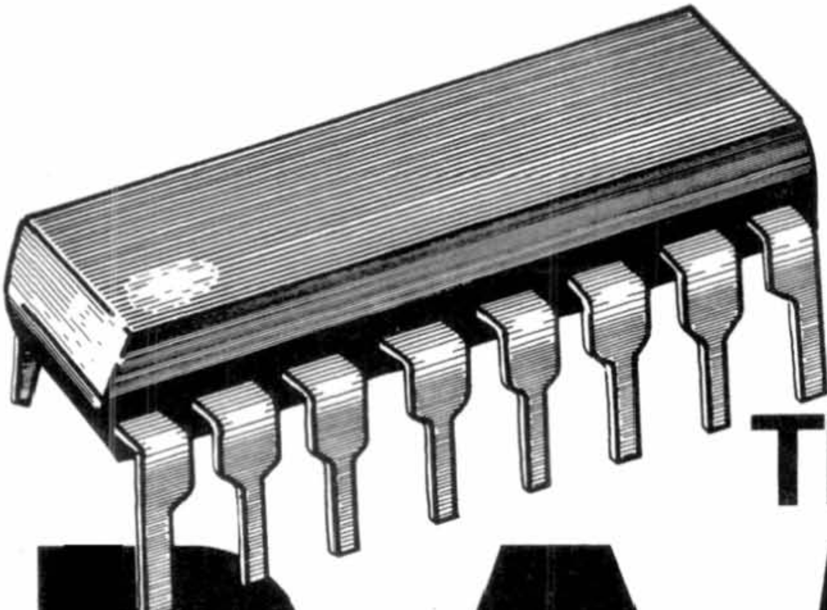
TRS-80 Level II BASIC, is priced at $9.95.

Reader Service ✓ 326

Time-sharing Peripheral

The Micromint Inc. is selling a new data communications product, the Chatterbox. The Chatterbox is a combination of the presently available Comm-80 I/O interface for the TRS-80 and an acoustic modem, to turn TRS-80 into a timesharing terminal.

The Chatterbox includes a built-in programmable 50-19200 baud serial port, a Centronics compatible parallel printer port, a 300 baud acoustic originate modem, and a spare TRS-BUS expansion connector.



TRS-80*

RAM

EXPAND YOUR MEMORY AT AN INCREDIBLY LOW PRICE

16K RAM - $59.50
32K RAM - $109.50

- Pretested
- One year warranty
- Available now

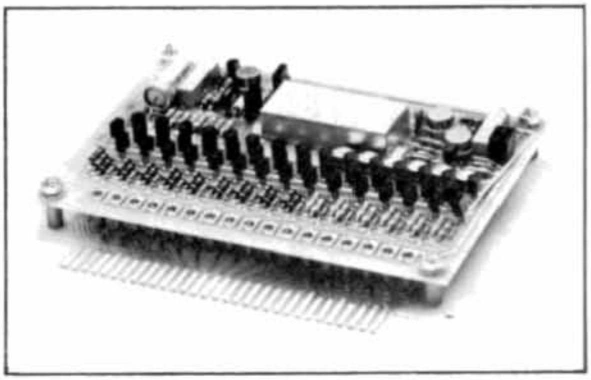
NEW FROM

MICROTEK inc.

9514 Chesapeake Drive
San Diego, CA 92123
Tel. (714) 278-0633
TWX 910-335-1269 ✓362

*Trademark Radio Shack, Div. Tandy Corp.

Comm-80 of Chatterbox Package



Model 1555S Analog-to-digital Converter

It comes with power supply, connection cable, user's manual, and terminal software for immediate operation. It is hardware and software compatible with existing TRS-80 products and connects either to the keyboard connector or screen printer port on the RS expansion interface. It does not require the RS expansion interface for operation.

The Chatterbox is sold for $259.95 from The Micromint Inc., 917 Midway, Woodmere, NY 11598.

Reader Service ✓ 329

TRS-80 to H-14 Interface

The Model PTS-3 (for use with an expansion interface) and the Model PTS-4 (for use without an expansion interface) are printer interfaces from Multi Media Systems. They make it possible to connect the TRS-80 microcomputer with the H-14 Serial Printer.

The PTS-3 plugs into the parallel printer port of the TRS-80 interface. The PTS-4 can be used with the PTS-3 in systems that do not include a Radio Shack expansion interface. The H-14 is connected to the DB-255 connector of the PTS-3.

A software driver is not needed with this system, because it will support all printer commands in BASIC or machine language.

The PTS-3 and the PTS-4 are each priced at $73.45 for an introductory period from Multi Media Systems, P.O. Box 41081, Indianapolis, IN 46241.

Reader Service ✓ 338

E.S.P. Lab for Research

E.S.P. Lab has been designed by Manhattan Software, Inc., P.O. Box 5200 Grand Central Station, New York, NY 10017, as a program for research into possible extrasensory phenomena, as well as for casual testing of the possibility of telepathy, clairvoyance, precognition and telekinesis.

Based on Duke University experiments, E.S.P. Lab selects randomly from among a set of five symbols (square, triangle, cross, wavy lines and oval), presenting one symbol at a time on the screen for telepathy experiments. All symbols are programmed in machine language and appear on the screen instantaneously. For clairvoyance and precognition testing, the program selects the symbol before or after the response, prompting only with a question mark on the screen.

Complete records are kept in memory of each symbol and response for a series of ten trials of 25 symbols each. Tables of results are constructed, with analysis of matching symbols and responses. For TRS-80 Level II 16K, E.S.P. Lab costs $9.95.

Reader Service ✓ 339

Packing Program

Data Assoc., Box 882, of Framingham, MA 01701, has released Pack8, designed to pack BASIC programs for faster loads and runs. Pack8 can reduce memory as much as 40 percent.

Pack8 is written for 32K and 48K Model I computers using one disk drive. All the lines in the program can be compacted, or a block of inclusive numbers. At the end of the packing, a summary is presented of the number of bytes and lines in the original program.

Pack8 is provided on cassette with instructions, and is sold for $19.95.

Reader Service ✓ 169

1555S Analog-to-digital Converter

Tustin Electronics Co. has added the Model 1555S to its series 1500 analog-to-digital converter product line.

The new model provides a complete conversion of an analog voltage to a 15-bit digital number in less than five microseconds. It is a monotonic unit.

The Model 1555S can be used by itself or with other analog modules to form a complete system. The converter price is $1500 from Tustin Electronics, 1431 E. St. Andrews Pl., Santa Ana, CA 92705.

Reader Service ✓ 327

Software Programming Tools

PROgrammer by Rational Software, 963 E. California Blvd., Pasadena, CA 91106, provides professional-quality programming tools to users of Level II BASIC.

After a short machine language routine is read, five functions are continuously available. A single keystroke directs keyboard input, which interprets and executes the command line. After execution, control is automatically returned to Level II BASIC.

The program also includes a keyboard debounce routine. PROgrammer costs $25.

Reader Service ✓ 336

Property Management System

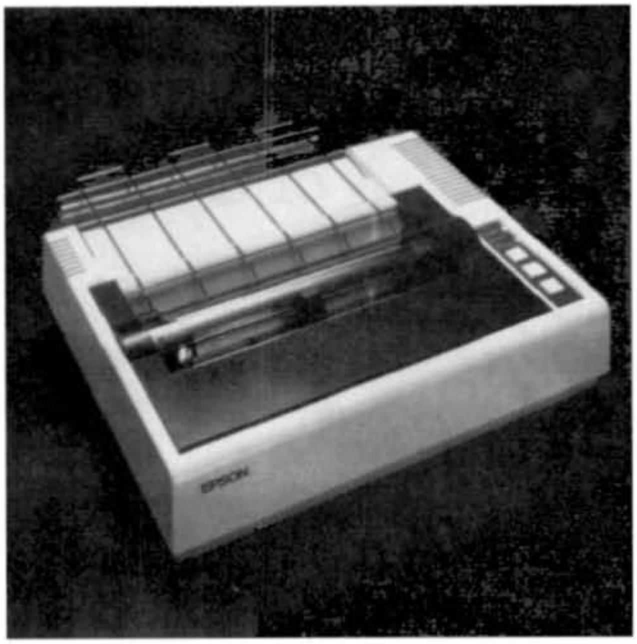
A-T Enterprises has a new income property management software package for TRS-80 Model II. The Property Management System (PMS) is designed to meet the Institute of Real Estate Management recommended computer system capabilities.

It is suitable for all types of income properties including apartments, mobile home parks, office buildings, warehouses, etc. The system is a full general ledger system that keeps track of all income and expenses providing formatted financial statements, management reports and exception reports upon request.

The PMS software operates on the Model II with 48K of RAM, two eight inch disk drives and a printer. It is written in CBASIC, runs under CP/M and is also compatible with most Z-80 and 8080 systems.

PMS, including one year maintenance, costs $650. A demonstration disk is available for $35 from A-T Enterprises, 221 North Lois, La Habra, CA 90631.

Reader Service ✓ 332



If you just bought another printer, boy are you gonna be sorry.

Epson.

The Epson MX-80. It's not just another worked-over rehash of last year's model. It's our top-of-the-line 80-column printer. It's new. From the ground up. And it's the most revolutionary printer to hit the market since Epson invented small printers for the 1964 Olympics in Tokyo. Don't take our word for it, though. Compare. There simply isn't a better value in an 80-column printer. Period.

But here's the fact that's going to stand the printer world on its ear. The MX-80 sports the world's first *disposable* print head. After it's printed about 50 million characters, you can throw it away. Because a new one costs less than $30, and the only tool you need to change it is attached to the end of your arm.

Now that's revolutionary, but that's only the beginning. The MX-80 also prints bidirectionally at 80 CPS with a logical seeking function to minimize print head travel time and maximize throughput. It prints 96 ASCII, 64 graphic and eight international characters in a tack-sharp 9x9 matrix. And it provides a user-defined choice of 40, 80, 66 or 132 columns and multiple type fonts.

We spent three long years developing the MX-80 as the first of a revolutionary series of Epson MX Printers. We employed the most advanced automatic assembly and machining techniques in existence to produce a printer that is incredibly versatile, remarkably reliable and extraordinarily inexpensive. It's a printer that could only come from the world's largest manufacturer of print mechanisms: Epson.

If it sounds like we're proud of the MX-80, we are. Not only does it do things some of the world's most expensive printers can't do, it'll do them for you for less than $650. That's right. Under $650.

And if that isn't revolutionary, we don't know what is.

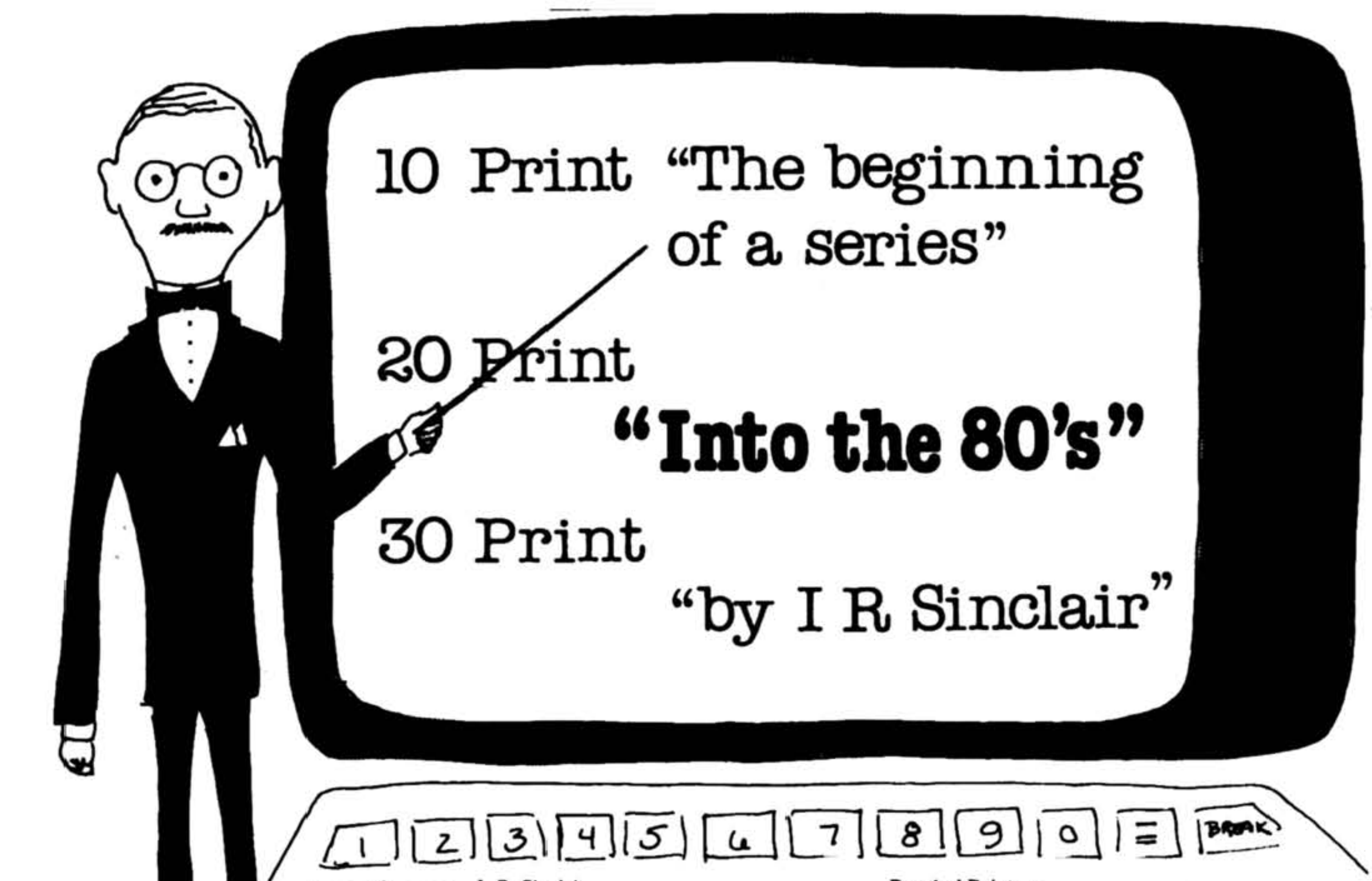
The world's first disposable print head. It has a life expectancy of over 50 million characters, yet it's so simple, you can change it with one hand. And it costs less than – repeat less than – $30.

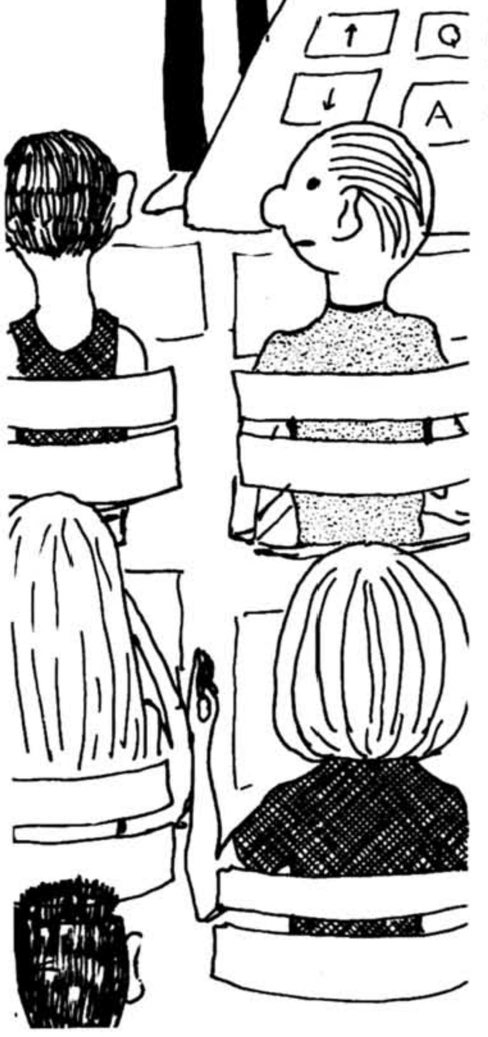
✓404

EPSON

EPSON AMERICA, INC.

23844 Hawthorne Boulevard, Torrance, California 90505, Telephone (213) 378-2220





I. R. Sinclair
89 Alexandra Road
Sible Hedingham
Halstead, Essex CO9 3NP
England

Many articles are written for those of you who have owned a TRS-80 for some time and know what programming is all about. So, this series is dedicated to the thousands of readers who are just about to buy a TRS-80—or have just bought one—and want to know what they've gotten into.

Throughout the series, we'll be talking about the Model I Level II TRS-80. Level is Tandy's word to indicate the complexity of programming language its computer is able to understand. We're dealing with the Level II only for two good reasons. First of all, Level II is much more useful. And, secondly, the Level I computer is already accompanied by an excellent manual for the beginner.

One measure of a computer's power is its memory. Memory dictates the amount of information (data) a computer can store and is measured in units called kilobytes, shortened to K. The 4K computer can store four kilobytes of data. Nothing in this series will cause you to run out of memory space on a 4K TRS-80.

Before you proceed, be assured of one point. The computer does exactly as you instruct it, nothing less, nothing more.

If you have not put in the instructions to print letters or numbers on the video screen, then these letters or numbers just don't get printed. One of the humiliating things about being a computer owner is knowing that whatever goes wrong is your fault.

Practical Pointers

Take a look at the sockets at the back of your 80. They are European DIN-type sockets and match five-pin plugs. There are three of them, one each for power in (from the transformer unit), cassette in/out (I/O) and video out.

Before you start using your 80, take the advice of an old hand and label these plugs and sockets with differently colored tapes. I use red for the power plug, green for video and yellow for the cassette. If your 80 sits in the same place and you never unplug it, this isn't significant. The odds are, however, that some day you'll want to shift it, and you could easily end up with plugs in the wrong sockets, since they are identical and easily confused in poor lighting conditions.

Keeping your system cool is another useful tip. Try not to have a desk lamp shining on the keyboard, for instance. It doesn't help, either, if you've used the computer all day with bright sunlight heating up the keyboard casing and the electronics inside. High temperatures also damage cassette recordings.

I've found that a normal room temperature of 70–75 degrees won't cause the TRS-80 any distress, even if you use it all day. It's another story if you have an expansion unit attached, but we won't go into that. Just make sure that that little black box which is the transformer unit is on the bench or on the floor, with room for air to circulate around it; don't put it inside a box or surround it with books.

Power It Up

You're ready to power up. Plug all the line plugs into the wall sockets and switch on. DON'T switch on the keyboard first—always have power on the cables before you turn on the

"If you have not put in the instructions to print letters or numbers on the video screen, then these letters or numbers just don't get printed. One of the humiliating things about being a computer owner is knowing that whatever goes wrong is your fault."

units, because the switch for the keyboard does more than just switch power.

Start the countdown by switching on the monitor. (On the one I use, the brightness control must be pulled to switch on.) Let the monitor warm up for a minute, then switch on the computer keyboard. The ON/OFF switch is at the back, next to where the power supply plug enters the casing. It's deliberately made a bit hard to find, because when you switch off a computer, all the program material you had stored in it is lost, gone forever, unless you recorded it on a cassette previously.

As you press the ON/OFF switch, you'll see the video screen suddenly filled with a mixture of numbers, letters and odd shapes. That's "garbage," caused by the computer memory being activated. Each little cell of memory can store a bit of information; zero voltage represented by a zero, or +5V represented by a 1.

Upon power up, when all of these cells are activated, some come on as 1s, some as 0s. About 8192 of these memory cells send signals to the video screen, the cells which are set to 1 cause parts of the video screen to light up, and the cells which are left at 0 keep the screen dark.

The result is a display of light and dark pieces at random or almost at random. Circuits inside the computer force these light and dark places into patterns, the patterns which we call letters, numbers and graphics blocks, and this is the pattern we see just as we depress the ON/OFF switch. When you release the power switch, the garbage clears, because the switch operates a memory clear routine for the video screen and memory—that's why it isn't a good idea to switch the keyboard on before plugging in the line and powering up.

The Mystery Message

Shining on the video screen in all its glory is the message MEMORY SIZE ? It has floored many a beginner. Did no one back at Fort Worth tell the machine what its memory size is, you ask? Ignore it for the moment, press the ENTER key, and these more reassuring words appear:

```
RADIO SHACK LEVEL II BASIC
READY
>
```

Your computer is prepared for programming in the BASIC program language.

What about the MEMORY SIZE? message? Well, as it happens, a lot of computer tasks can be performed faster and more efficiently by giving instructions directly to the microprocessor chip inside the computer. This chip needs special instructions, called machine code or object code, and these instructions can't be loaded into the computer in the same way as an ordinary BASIC program.

Unless you're going to use machine code programs right away, though, you can ignore the MEMORY SIZE? question.

The one time you can't ignore it is when it appears while you're running a program. When that happens, it's an unwelcome sign, called a re-boot, that something is very wrong with your program. The computer has started its power-on sequence again. There's no harm in it, but you will have lost your program unless you saved it on cassette earlier.

The READY signal is an invitation, but unless you know what it's inviting you to do, you can't take advantage of it. At the READY signal, you can either load a program from a cassette or you can type one yourself. Since you'll learn more about the 80 from writing your own programs, however simple, we'll start there and leave the frustrations of cassette loading for a later date.

READY is an invitation for a BASIC program. BASIC is an acronym for Beginners All-purpose Symbolic Instruction Code, and it's the easiest of programming languages to learn.

Why should we have to learn BASIC?

It's all bound up with the way computers work and are designed. The fastest and most efficient programs are written in machine code, but learning and using machine code is a painful business, and writing machine code is a frustrating experience.

For these reasons, computer designers have continually sought to make it possible to give instructions in simpler forms, using English words (or Spanish, French, Italian and others) and stringing them together in a way which is reasonably simple to understand.

If you've looked at some of the programs printed in the back of the TRS-80 Manual or published in *80 Microcomputing*, you might not quite believe this last statement, but compared to most other computer languages, BASIC is reasonably easy to understand. We can devise simpler languages, but the penalty for using a simpler language is either that it doesn't do as much as we would like, needs more memory or takes longer to run. Right now, your TRS-80 comes with its BASIC language built in.

Like any other language (and I've had to cope with Latin, French and Greek in my time), BASIC is best learned by using it. Unless you intend to use your computer simply to run programs written by other people and obtainable on cassette, you haven't much choice—learn BASIC!

Unlike other computer languages, BASIC isn't standardized. A program written in BASIC for another computer may not run on the TRS-80, and vice versa, unless you make a few changes.

It is possible to write BASIC programs that will run on any computer equipped with the BASIC language (the Adam Osborne programs are good examples of this), but you can get a lot more out of your 80 if you know the peculiarities of its dialect of BASIC.

This dialect, incidentally, is one of the most advanced BASICs fitted to a small computer. We can't hope to show all the features of BASIC programming in this series, but we can try to fill the gap between elementary BASIC textbooks and Radio Shack's *Level II Reference Manual* that comes with your computer.

Program Proverbs

If you've never programmed a computer before, learning BASIC on your own can create as many ulcers as guarding a bank during a revolution. Computers are fussy about the way you use the language. If the manual says that a word must be followed by a comma, then it really must be followed by a comma, or your program won't run.

If, on the other hand, a word needs a semicolon after it, you can't get by with just a comma or a colon. This punctuation is translated into instructions by the computer, and different marks denote different instructions.

Translated? That's just what happens. There are about twelve thousand bytes of memory inside your TRS-80 which you can't alter. The professionals call it ROM, or read-only memory. It reads your BASIC program and converts each letter or word into machine code instructions to the microprocessor. Because each instruction is converted and then carried out, the instructions are much more long-winded than those in a machine code program.

Bigger computers can do what's called compiling, which means translating the entire program into machine code in one run, and then running the machine code. The TRS-80, like all small computers, only interprets—it converts each instruction in the BASIC program into code, runs it and then goes on to the next part. The difference is between dictating directly to a secretary, with all the um's and er's of speech and delivering an edited tape to an audio typist.

Let's start again with the READY signal staring us in the face. Your first response toward writing a program is to type a line number. The line number is a tag we can attach to an instruction so that we and the computer know where to find that instruction.

Take any number between 1 and 65529. Normally, you start at 10 or 100. It doesn't matter what numbers you choose, since you don't use more memory by numbering lines 100, 200, 300 than by numbering them 10, 20, 30. Important to remember is that the program should flow from the lowest numbered line to the highest.

Since we have to start somewhere, type 10 and a space. Your program now starts at the line numbered 10. Since it's a bit depressing to stare at a blank screen, we'll get the computer to PRINT my name.

```
10 PRINT "IAN R. SINCLAIR"
```

Remember, the quote marks must be in place, because they indicate to the computer that the words between them are to be printed on the video screen and are not part of an instruction. Marks that divide one kind of word from another are called delimiters, and the quote marks are the easiest of these delimiters to work with. Make sure that anything you want printed to the video screen starts and ends with quote marks.

Made a mistake? We're not all trained typists, so the TRS-80 is very forgiving. Use the backspace, the arrow which points to the left, on the key next to the @. Each time you press this key, one letter of your instruction is wiped out, and you can type another one. Keep on until the line is perfectly typed.

Now that you've typed the line, you must press ENTER to make certain that it's planted in the computer's memory. Even if you forget the line number, you still get a result—the words which were inside the quote marks will be printed, but you don't have a program because it won't repeat, and you must start again. If your line was entered correctly, it is now a very simple program. We can run the program by typing the word RUN and then hitting ENTER again.

All right, so it's not impressive, but each time you type the word RUN and hit ENTER, you'll get my name printed on the screen. Makes me feel good at least.

Try typing the instruction with no space between the 10 and the PRINT, so that it reads 10PRINT "IAN R. SINCLAIR". Does it make any difference in the way the program runs? Did I say that computers were fussy?

Now type LIST10 and hit ENTER. This prints your instruction line on the screen again. Notice anything about it? Right, the space has been put in again between the 10 and the PRINT. You can't cheat your TRS-80 this way!

Now try some more deliberate errors. It's just as well to know at this stage what effects they have, rather than be tangled up with an unfamiliar error message later on. For a start, leave out the first set of quote marks. Then try typing PPRINT or PRIINT instead of PRINT. Doesn't run, does it?

Instead, you get the words SN ERROR. SN is a shortened version of SYNTAX, a word that language teachers use to mean the way in which a language is constructed. If you say in English, "I am here since yesterday," your syntax in English is poor, but the same phrase in French is grammatically correct.

Syntax is a matter of rules, and a SN error means you have broken a rule of BASIC, by leaving out a delimiter, or misspelling an instruction word. Some newer computers, incidentally, reject a syntax error the moment it is entered, but the TRS-80, like its entire generation of home computers, doesn't spot the error until you try to RUN.

✓Reader Service—see page 226

How much can you PRINT in one instruction? Try it! Type 20 PRINT ". . . . . and then type as many words as you like. You'll find that the words don't run off the edge of the video screen, but form new lines of text on the monitor. The video monitor accepts only 64 characters/line, while the computer allows up to 250 characters/line. The computer no longer responds to keyboard input when the 250 character limit is reached.

PRINT Plus Spaces

Now try the one-liner below:

```
10 PRINT"      SPACED ALONG"
```

Because we've labeled this line as line 10, it has wiped out the old line 10. Notice that the space between the first set of quote marks and the first letter of the word affects the way it prints. This is one way of adding spaces.

Another way is by indicating tabs. Tab means the same as it does on a typewriter, tabulation. The width of the video screen is divided into 64 starting points, numbered from 0 to 63. Using TAB( ) selects one of these as a starting point for what you print.

```
10 PRINT TAB(25)"IN THE CENTER"
```

Notice the syntax—parentheses after TAB enclose the number, between 0 and 63. The quote marks surround the words to be printed.

If you had put the first set of quote marks between PRINT and TAB, you would have printed, at the left-hand side of the screen, the phrase TAB(25) IN THE CENTER, which isn't exactly what we wanted.

Why TAB(25)? Well, I counted the letters and spaces in IN THE CENTER and made it 13 characters. I rounded that to an even number, 14, then subtracted from 64, leaving 50. Half of 50 is 25, and that's the tab number. Why? Well, the number of letters and spaces gives the number of positions on the line which are used to print. If we want these words to be centered, then there has to be an equal number of spaces on each side, and we find this by dividing the number of unused places, 50 in this example, by two.

Try printing your own name centered on the screen.

Notice, by the way, that we're still entering each one-line program as line 10, deleting the previous program in line 10. Later on, we'll look at other ways of removing old programs, getting rid of unwanted lines or running only the lines we want.

TAB is one of the BASIC commands or functions that not all small computers have, but it's only the start of the options which are available to the TRS-80 owner. Type the next line very carefully.

```
10 PRINT "SPACE", "THE", "WORDS", "OUT"
```

See the commas? Each comma lies outside the quote marks that set off the words, because we don't want the commas printed. If we type:

```
10 PRINT "SPACE, THE, WORDS, OUT"
```

hit ENTER, type RUN and hit ENTER again, the printout on the video screen would be:

```
SPACE, THE, WORDS, OUT
```

As it is written, the effect is quite different, as you will see. The commas have commanded the computer to space the four words across the width of the screen. Each comma instructs the computer to start printing the next word at the next print zone. There are four print zones, each of which can take up to sixteen characters (letters, numbers or spaces). If you PRINT more than sixteen characters in a zone, the comma causes a skip to the next zone. Try the following program.

```
10 PRINT"SEE THE EFFECT";
20 PRINT"OF A SEMICOLON";
30 PRINT"ON THE PRINTOUT"
```

Example 1

```
10 PRINT "SPACE"TAB(POS(0)+5)"THE"TAB
  (POS(0)+5)"WORDS";TAB(POS(0)+5)
  "EVENLY"
```

Example 2

```
10 CLS
20 PRINT TAB(28)"HEADING"
30 PRINT
40 PRINT TAB(5)"THIS IS A
   NEATER WAY OF PRINTING"
```

Example 3

```
10 PRINT "THE ZONES WILL TAKE", "SIXTEEN
CHARACTERS EACH"
```

The comma is used as a print delimiter. We can also use the semicolon as a delimiter, but with a very different action. Try entering the program in Example 1. With more than one line of program, you must remember to push ENTER at the end of each typed program line. When you run the program (type RUN and hit ENTER), what happens?

The semicolon is another signal to the computer when it's used in this way. Typed after a printed quantity, the semicolon means: use the same line and keep printing, so that the words we typed in three separate instruction lines end up on one single line of type. This would happen, incidentally, even if we had several other lines of instructions between these PRINT instructions, so watch carefully for these semicolons if you are entering a program which has been written in BASIC by someone else. All small computers use this form of instruction.

What happens if you leave out the semicolons? Try it! Each PRINT command causes the video to start on a new line. This is one way you can space out your printing vertically.

Meanwhile, look at Example 2 for a very powerful command which few small computers have in their BASIC. It's the POS function. POS means position, and it means the position of the cursor mark, that short line which shows where the next letter will appear on the screen when you type a program line.

As a result of the POS instruction, the computer makes a note of how far along a line you have printed. Add five to that place number, as we've done in Example 2 and the result is five spaces between each word.

When you type line 10, by the way, don't hit ENTER until the end of the instruction after the last set of quote marks. If you hit ENTER before then, you're indicating that you want to start another numbered instruction line. The other point is that you should type DELETE 20-30, and hit ENTER before running the program. If you don't, you'll print the words in lines 20 and 30 of the previous program, unless, of course, you have switched off between working Examples 1 and 2.

By this time, your video screen must be looking a bit cluttered, so let's look at another useful function that helps clear things up. Example 3 shows a four-line program (we're getting more adventurous) which makes your video printouts look better. The new instruction, CLS, means, simply, clear the screen.

When the program is run, the screen clears, removing the program lines. After the CLS instruction has been used, the next PRINT instruction will place the words in quote marks on the top line.

In line 30, I've used the word PRINT by itself. What does that do? Try leaving it out by typing DELETE 30 and hitting ENTER. Then type RUN and hit ENTER again. See the difference? The PRINT command in line 30 causes a one-line space between HEADING and the words in line 40. Want a two-line space? Then type 30 PRINT, hit ENTER, type 40 PRINT and hit ENTER again. Now run this one and watch the larger gap appear between the heading and the line of words.

Time to take a look at the program. Type LIST and hit ENTER. Your program appears under the last lot of printing, with the lines in correct order. That's just one example of why the BASIC language uses these line numbers. We can place any line number against a line, and the computer sorts them into order. If we use the same number for two different lines, then the last one typed and entered wins, the older one is deleted. We can also delete lines by typing DELETE, then the line number and hitting ENTER.

Can we delete more than one line at a time? Sure we can. Just type DELETE 10-40, and every line of the program in Example 3 will be rubbed out. The other way we can remove a whole program, but this time without needing to know how many lines it has or what their numbers are, is to type NEW and then hit ENTER.

More Spaces

Suppose we want to print a word at the center of the screen. There are sixteen lines of print on a full screen, so we could print on line eight. We could type seven lines with PRINT, but no words, to make the print position move down one line at a time. We could then use the commands PRINT TAB( ) to space the word to the center of line eight.

There's a much easier way of doing this sort of thing with a TRS-80, by using the PRINT@ command. It must be entered correctly, with no space between the T of PRINT and the @ sign, and a comma immediately after the @. If you put an unwanted space in, you'll get the SN error message when you try to run it. A much less obvious error is typing @ with the SHIFT key depressed. If you hit SHIFT and @ at the same time, the @ appears as usual on the video screen, but the code number which is fed into the computer is NOT the correct one. You'll get the SN error report when you try to run it, but the line will look good on the screen.

With these warnings in mind, try the program in Example 4. Remove the previous programs by typing NEW and entering. Now type in the three lines of the new program and run it. Interesting? The word GREETINGS appears around the center of the screen, and the next line of print is four lines under that.

Take a look at page E1 at the back of your Level II manual. Turn the page so that the numbers are all correct way up, and you can read off the PRINT@ numbers and the TAB numbers. The numbers in heavy type at the top are the TAB numbers for each line, and also the PRINT@ numbers for the first line, the top line. For the second line of PRINT@ the numbers start at 64 (see the columns down the left and right-hand sides), for the third line, 128, and so on.

To find a PRINT@ starting number for any position on the screen, pick your spot, locate the TAB number at the top of the page and the PRINT@ number for the start (at the left or right-hand side), then add the two. For example, if you want to start around the center, try TAB(31) on the line starting at 448. That gives us a PRINT@ on the number 448 + 31 = 479, so we type PRINT@479, then add the quote marks and the message. Remember the syntax: PRINT, no space, @, no space, comma, then quote marks for the message, and keep your fingers off that shift key when you are typing the @.

Using More

The TRS-80's big, big BASIC allows us yet another way of printing which isn't available to people with other types of machines. The new instruction this time is PRINT USING, and it instructs the computer to arrange the printing to suit some definite pattern which must be specified in the PRINT USING command. PRINT USING is most useful when you have to print out a number in standard form, such as a price or a sum on a check. It's also useful when you want to round off a fraction.

Since the number of times you are likely to want to use this command in your own programs is limited, compared to the everyday ones such as TAB and PRINT@, we'll look at only a few of the PRINT USING commands.

Example 5 shows PRINT USING applied to rounding off a fraction. Enter the program and run it to see how the number is printed. This is a smart way of making sure that your printout doesn't contain lots of figures after the decimal point. After all, you wouldn't like to think that you had just printed a check for $56.2357.

Another useful feature of the PRINT USING command is that it can insert a floating dollar sign. Now, if you thought that the dollar was sinking rather than floating these days, let me explain that phrase. You might want to print out something like amounts of $1.50, $26.40, $147.50 and so on. What the floating dollar sign does is position itself ahead of the first figure of the number, so that you don't print 1$27.50 and $02.40. Try it out with the program in Example 6.

The Level II manual has a large number of examples of PRINT USING, so we'll leave this one, which is a more specialized command than most of the ones we shall be using in this series.

BBug BBreak

Depending on the age of your 80, you may already have met the dreaded kkeybbounce. You type PRINT, and it appears on the video screen as PPRINT or PRIINT or some other

```
10 CLS
20 PRINT@475,"GREETINGS"
30 PRINT@704,"THIS USES THE PRINT@
   COMMAND TO SPACE LINES"
```

Example 4

```
10 CLS
20 PRINT USING "$$##.##";21.471
```

Example 5

```
10 CLS
20 PRINT USING"##.##";2.736
```

Example 6

```
1 POKE 16553,255:CLEAR:FOR N = 16480 TO 16492:READ K:POKE N,K:NEXT:FOR N = 16435TO16
437:READ K:POKE N,K:NEXT:POKE 16405,0:DELETE 1-2
2 DATA 205,227,3,183,200,14,40,16,254,13,32,251,201,195,96,64
```

Listing 1

weird combination of repeated letters.

When you try to run a program with an unwanted double letter in a command word, you'll get a SN error message, meaning that the internal circuits of your computer simply don't recognize the word. Of course, if you have an unwanted double letter in a message which is enclosed in quote marks, then it will simply be printed out that way.

Keybounce is a problem that plagues any mechanical switch, like keyboard switches. You press a key, and the electrical contacts close. But, because they're made from springy material, they bounce open again before finally closing and staying closed. Each time the switch closes, it completes an electrical connection, and if that happens to be the electrical connection which prints the letter P on the video screen, then you get two of them.

Every manufacturer of computers gets around this by using a time delay each time a key is pressed. The computer takes no notice of the key until the time delay is over. Only a small time delay of about a thousandth of one second (one millisecond) is needed.

Radio Shack seems to have given short measure to this problem on the older TRS-80s. On some models, keybounce can be fixed by pulling off the keycaps and cleaning the contacts. That's what Radio Shack says, anyway, but my own TRS-80 has fixed keycaps that don't come off easily, and it bounces very badly. The keybounce is so bad, in fact, that if there were no cure for it, I would have scrapped the whole thing months ago.

Yes, there is a cure, and it works. Radio Shack supplies a program on a machine code tape entitled KBFIX. Enter this one, and your keybounce troubles are over until the next time you switch on. Incidentally, more recent TRS-80s have no trace of keybounce.

Sometimes keybounce is just a nuisance. Other times it can cause a complete hangup of your computer, and one of these other times is when you type LIST and get LLIST. LLIST is a command word which was unfortunately chosen for printing a list on the Radio Shack line printer. Run LLIST and the result is—nothing. No keys have any effect, nothing appears on the screen, and the computer appears to have died on you, with just that accusing word LLIST staring at you from the screen.

What's happened is that you have commanded the computer to print a list on the printer, and because there's no printer connected, it's waiting for you to connect one. Don't rush out with a fistful of dollars, because you can recover from this stall in two ways.

One is to switch off completely, but that way you'll lose any program you had in the computer. The easier way is to push the RESET button at the back of the computer, at the opposite end from the ON/OFF switch. On my 80, this is under a small flap which also houses the connector for expanding memory. Pushing and releasing this switch removes the hangup, and the computer is ready to run again. Whatever your manual says, you don't lose your BASIC program when you use RESET unless the Radio Shack expansion interface is connected.

The keybounce problem can also be avoided by using a short BASIC program, shown in Listing 1. It's a sight longer than the others we've used in this part, and you have to be sure that you've typed each character correctly. Run it, and it sorts out the keybounce and then deletes itself!

At this stage, it's not easy to explain how it works, because it makes use of parts of memory we don't normally use, the reserved RAM. The purpose of these memory parts is kept a close secret, and it's only when someone with a bit of time to spare investigates them that we even get to know about them.

Keybounce is just one example of what folks in the computing business call a bug—a flaw in a system. A bug may be in the operating system, or it may be in a program. Wherever it is, these articles will show you how to stamp out some of the most active bugs. ■

SIMPLE AS BASIC BUT VERY FAST—*BALCODE*

You can now program your Model I TRS-80 using Op codes based on IBM 370 BAL. Why devote your time to BASIC or to Z-80 assembly language when the BALCODE editor/assembler is now available for only $79? We mail you a monthly Newsletter covering all the fine points of assembly language programming. This package comes with a Programming's Manual and a Demo program to walk you through all the BALCODE commands.

Our EDITOR is tops. Includes COPY, MOVE, FIND and CHANGE Commands with stop at CRT fill-up.

Our BALCODE MACRO Assembler assembles floating point numbers.

Our System all resides in memory at the same time. You edit, assemble and run without reloading. Origin is at 7000H and object code is 12,000 bytes.

BALCODE 80A $79. CHECK, MO, VISA OR MASTER CHARGE.

Write for free listings or send $10 for Programmer's Manual. Applies towards purchase.

BALCODE SOFTWARE, INC.

12 LEEDS LANE, N. BABYLON, N.Y. 11703 (516)-242-1665 EVES. ✓391

......60 Cycle Sine Wave U.P.S.

(Uninterruptibe Power Supply)

••••for those systems that need 60 cycle sine wave keeps computer & disk systems on when the power goes out **For less than $500.00***

••••Rated 250 Watt continuous operation

•••• will provide 400 Watts for 5 minutes (Model II TRS-80 with 4 disk drives)

* Standard MAYDAYS available starting at $195.00 for 150 Watt

Mayday™

✓151

from••• *Sun Research, Inc.*

Box 210 New Durham, NH 03855
(603) 859-7110 TWX 510-297-4444

Blow by blow accounts.

A Bout with the I.R.S.

Fred Blechman
7217 Bernadine Ave.
Canoga Park, CA 91307

In one corner, weighing 29 pounds, in black with gray trim, we have the $750 Radio Shack TRS-80 Level II 16K Microcomputer. In the other corner, weighing *tons*, in red, white and blue, we have the multi-million dollar IRS worldwide computer network. Come out fighting, and may the better machine win!

Round 1: Getting it together

It wasn't a referee who introduced me to the IRS. It was a plain brown envelope advising me of an office audit appointment and detailing the items I should bring. Needless to say, that bit of mail destroyed my day—and most of the next few weeks!

I have had a small business with no employees for 15 years and always wondered when the day of reckoning would come. My tax preparer told me many times that my records were complete and accurate and that I would coast through an audit.

Still, there were always gnawing doubts about whether I could provide satisfactory verification for various expenses. Like everyone, I had heard IRS audit horror stories, and my impression was that no one comes through unscathed. Furthermore, I assumed the auditor would be nasty, intimidating and threatening.

The audit was for 1977. In the three-year interim, we had moved and still had not unpacked cardboard boxes with business records going back 15 years! To further complicate the picture, my wife, Ev, had her own retail business before we married in 1977, so we had filed a joint return, with separate Schedule C's for each business.

The auditor's letter specified that we furnish, to my surprise, not only our business records and checking account statements, but all personal checking account statements and checks and *all* savings account passbooks!

Yes, *all*! That meant *every* statement and savings passbook from December 1976 to January 1978!

For many people this might only amount to a few accounts, but for Ev and me this meant eight checking accounts and four savings accounts that showed activity in that time period. It took considerable time and effort to get everything together, especially since Ev had closed her business at a considerable loss (that's what triggered the audit!) and delegated, in disgust, a lot of records to the round file!

Finally it looked like we were ready to face the almost infinite resources behind the IRS paper tiger.

Round 2: The $45,000 Punch

The auditor, Edward, was very congenial, as he led us to his cubical on the third floor of the imposing local Federal Building. Edward spent the first half-hour or so interviewing us about our individual businesses and establishing background information.

"Any inheritances received in 1977? Gifts? How much cash do you keep around the house? Why so many checking and savings accounts? Any loans?"

I suppose if I were cheating, or more paranoid than I had already become, I would have objected to some of the questions as being too personal (practically the only thing he didn't cover was our sex life!)

Instead, I was fascinated by his thoroughness. I did not realize he was slowly but surely peeling away my defenses, preparing for a blow to the solar plexus!

During this time, he was furiously making notes of our replies to his questions. Reading upside down, I saw "T/P claims" in various places. "T/P" means taxpayer.

Asking for our bank statements, he proceeded to list every monthly deposit total on a sheet of lined paper! He did this with *every* bank statement for *every* month in 1977! Then he did the same thing with the savings passbooks, listing *every single deposit* by date!

He used eight sheets of paper and took almost three hours—into his lunch hour—and came up with a grand total of $141,531.86 deposited in all our accounts in 1977.

Then he checked our 1977 Federal Income Tax Return and added the gross income: $96,091.

"Mr. Blechman," Edward said, "We have a difference of $45,440.86 between your deposits and your reported income. Some of this difference may be transfers between accounts or non-reportable income we may not have covered in our initial discussion—loans, social security, gifts, unemployment insurance and so forth. Any of this $45,000 plus you can't identify and verify will have to be considered non-reported taxable income. Our next appointment is......"

The fact that it rained that day, that my raincoat had a big hole in it, and that I lost my rainhat, only added to the feeling that I was down for the count!

Round 3: The Pussycat's Claws

Wondering what I'd look like in stripes and how the food was in Leavenworth, I carefully assessed my situation. I knew we moved money around to pay bills, kept balances equalized and made major purchases through the checking accounts. I had a policy of keeping small balances in the checking accounts, since they earn no interest. Sometimes cash-flow needs required transfers from my savings accounts. But $45,000 worth of transfers? That much? It didn't seem possible.

I looked for errors in the auditor's figures and couldn't find any. Still over $45,000 unaccounted for. I felt helpless and overwhelmed, so I picked up my February copy of *80 Microcomputing* to escape and let my subconscious

To add string function to "Household Accountant," *change* the following program lines:

```
540    Change  C to C$
550    Change  -1 to END
950    Change  C to C$
960    Change  -1 to END
1000   Change  C to C$
20000  Change  -1 to END
```

Example 2.

mind look for a solution.

Fate smiled on me, as I turned to page 114 and saw David C. Andreasen's "Household Accountant" with the kicker "Keep track of how much, to whom and when—up to 32,767 accounts."

Well, I didn't have *that* many accounts, so I looked closer. I got excited—this could do it. This could sort out and add transfers between accounts!

I sprang into action. I keyed the program on page 116 into my 16K Level II TRS-80 pussycat. She almost purred as I ran and tested the program, using dummy DATA. It worked!

Then, I realized, I was limited to check number entries only, so I made a few program changes (Example 1) to accommodate strings (loan, cash, etc.).

I started with one checking account and entered each check that transferred money to another account. I assigned a category number to each "path" (Bank one to Bank two was category one, Bank one to Savings one was category two, and so on). I ended up with 18 different paths, or categories, between accounts!

I made some additional changes to the program (Example 2) to accommodate my printer. The program added and printed the totals for all categories, for each month and for the year, listed all transfers individually, printed and added each category—fourteen pages on my printer!

With the help of this program, I was able to account for almost $43,000 of the $45,000 plus "unexplained income"!

Round 4: The Technical K.O.

Armed with my computer printouts, plus six cardboard boxes full of records, wheeled in on my dolly, we faced Edward a second time. Once he understood what I was showing him, he was delighted. He said it saved him hours of work!

However, just to make sure I was correct (after all, how can a $750 computer do anything but play games?), he spent two hours going over the figures, asking me for checks, deposit books, etc., to verify the printout. It was not one penny off! Personal computing had scored a technical knockout!

You may wonder about the other $2,000 plus that was not accounted for in my analysis. We explained that it was probably garage sale cash and checks deposited in checking accounts, since Ev sold a lot of her furniture and housewares before we got married that year. No problem. You usually sell things at garage sales for less than you paid, so this is usually not taxable income.

Round 5: The Next Match

Forewarned is forearmed! Keep good records. The burden of proof is on the taxpayer, not the IRS. Despite the fantastic help the "Household Accountant" provided, it is useless without correct data.

Fortunately for me, I have a habit of jotting remarks next to every deposit or withdrawal in my savings passbooks, so I can identify the purpose of each transaction. Without these remarks I doubt if I would have been able to discover a lot of transfers, and that would have been more unreported income on which tax would be due, plus penalty and 12 percent interest for two years!

An auditor told me that the majority of taxpayers end up paying extra taxes as the result of an audit simply because they can't identify, document or verify deductible or non-taxable income.

While an IRS audit is about as pleasant as swimming with a bleeding leg in shark-infested waters, it can be a real challenge and a learning experience. ■

To add printer output for "Household Accountant" *add* the following program lines:

```
105   CLS:PRINT:INPUT"PRINTER(YES=1,
      NO=0)";PR
790   IF PR=1 GOSUB 2000
1025  IF PR=1 GOSUB 2000
2000  FOR T=0 TO 15
2010  S$(T)=" "
2020  POKE VARPTR(S$(T)),64
2030  POKE VARPTR(S$(T))+1,(T*64+15360)
      AND 255
2040  POKE VARPTR(S$(T))+2,(T*64+15360)/
      256
2050  NEXT T
2100  FOR T=0 TO 15:LPRINTS$(T):NEXT
2110  RETURN
```

In line 470, *change* -1 to END

Example 1.

✓Reader Service—see page 226

Manipulate your world.

Pulling Strings Together

John D. Adams
13126 Tripoli Ave.
Sylmar, CA 91342

Computers differ from calculators in a number of ways. One of the most significant differences is in the computer's ability to process alphanumeric data (data containing letters, signs and symbols as well as numbers) and to handle "strings."

In computerese, a string is a group of alphanumeric data which is regarded as a single unit. A "prompt line," such as ENTER YOUR NAME, in a program is entered and stored as a string. In a program which must stop to allow manual input of data, the line PRESS ENTER TO CONTINUE might be added to continue execution, after needed data has been typed. Again the line is treated as a string.

A Giant Leap

As with arrays, the distance from Level I to Level II is a giant leap. Level I allows two strings, A$ and B$, each of which can be up to 16 characters long. Now we suddenly have hundreds of string locations available, each of which may contain up to 255 characters.

An extensive set of string handling instructions has also become available, allowing us to make some pretty classy moves. We will deal with those instructions in detail in part two. As a beginning we will discuss some basic facts about strings and how computers "recognize" them.

String and non-string data are handled differently by the TRS-80. Because of this difference we must tell the computer how we want information stored. One way to do this is to use a declaration sign in the variable name. In many dialects of BASIC, the dollar sign is used for this purpose.

With this sign we can use the names A$, A(1)$, A1$ and AA$, and these variables will all be designated as different string variable names. This is a simple way to indicate string input, if you have only one or two strings, and they will not be referred to very often in the program. An example of this might read:

```
10 D$ = "DISTRICT # 1"
```

A second and more efficient method involves the DEFSTR instruction. Inserting the line 10 DEFSTR A – D at the beginning of a program causes the computer to automatically regard any variable starting with A, B, C or D as a string variable, and the dollar sign may be omitted. Combinations such as B(1), DW, A7 or single letters such as C are now valid string locations.

Using these methods makes getting strings into and out of the computer quite easy. We can simply put the string into a program line, such as 10 A$ = "JANUARY". Note that the data must be enclosed in quotation marks. If you omit the quotes, when the RUN starts the computer responds with a ?TM (type mismatch) error message. Even though the dollar sign is there, the computer does not "see" the word JANUARY as proper input. We must remember that a computer uses complex procedures to convert those letters into numbers, after which it proceeds with operations.

Strings may also be entered using the INPUT instruction. Very often this is done with a prompt line, such as: 10 INPUT"ENTER MONTH";A$. Inputting strings this way does not require quotation marks, *unless* the string contains commas, colons or leading spaces. If any of these conditions exist, quotes must be used.

When you enter a lot of string information, perhaps the best method is to utilize the READ and DATA pair. The line, 10 READ A1$: DATA JANUARY" loads the name of the month into string location A1$. Once again, quotes must be used if commas, colons or leading spaces are part of the string.

The following lines illustrate three ways of loading strings:

```
10 A$ = "JANUARY"
20 INPUT"ENTER SECOND MONTH";B$
30 READ C$
40 DATA MARCH
50 PRINT,A$,B$,C$
```

Line 10 loads the string manually through insertion in a program line, line 20 by using the INPUT instruction, and lines 30 and 40 by using the READ/DATA pair. Output is shown in line 50. The comma after the PRINT instruction places the three strings in zones two, three and four on the monitor screen.

Loading large numbers of strings may be done quickly and efficiently by using the READ/DATA instructions along with a FOR-NEXT loop. The following lines load the days of the week into an array:

```
10 DEFSTR N
20 FOR A=1 TO 7
30 READ N(A)
40 NEXT
50 DATA SUNDAY,MONDAY,TUESDAY,
   WEDNESDAY,THURSDAY,FRIDAY,
   SATURDAY
```

The commas in line 50 separate the data into individual "portions" which are picked up in order as the loop cycles. As each portion is stored, the computer marks it off so that it will not be used again without a RESTORE statement. When the loop finishes, the word SUNDAY will be stored in N(1), MONDAY in N(2), etc. through SATURDAY in N(7). Outputting these strings is also accomplished with a loop. Add the following lines and RUN them:

```
60 CLS:FOR A=1 TO 7
70 PRINT N(A)
80 NEXT
```

String Handling

Now that we have had a look at inputting and outputting strings, we can stop to discuss some idiosyncracies of string handling. Earlier, I mentioned that string and non-string data are handled differently. What is this all about? As a visual aid to our discussion of "string literals," enter and RUN the following lines:

```
10 A = 5
20 A$ = "5"
30 PRINT A + A
40 PRINT A$ + A$
```

Hmmmmm. I'm pretty certain that 5 + 5 = 10, but look what line 40 has produced! What happened? When a number is introduced into a string it is thereafter regarded as a *symbol* and not as a *value*. No computation may be done with that number as it exists in memory. You may recall the figure 5, but not the value 5. Level II does give us a way to get around this problem and it will be dealt with later.

Line 40 illustrates an operation called "concatenation," a fancy word for joining or linking strings together. In line 30, the computer took the value 5, added it to the value 5 and returned the value 10. Line 40, however, took the symbol 5 and linked it with the symbol 5 to produce a new string which reads 55. We could change line 40 to read: B$ = A$ + A$ and the new string (55) would then be stored in B$.

This brings us back to the string/non-string matter. You will recall that we could not load A$ = "JANUARY" without using quote marks. Trying to load A = "5" results in the same mismatch error message. The dollar sign and quote marks *must* be used in conjunction with each other.

In the JANUARY case we had specified a string location by using a dollar sign with A. The quotes said, in effect, "We contain the string to be stored, *not the values*." The quotes around the 5 say the same thing, but using A without the dollar sign indicates that the location is set aside for a value and not a string, so the line is rejected. The computer *does not* know the difference; you must tell it. We must be constantly aware that string data cannot be loaded into a non-string location and vice versa.

String operation is also quirky about spaces. Enter and RUN the following lines:

```
10 READ A$,B$,C$
20 DATA JANUARY, FEBRUARY, MARCH
30 PRINT A$;B$;C$
```

Hey wait a minute! I put semicolons in line 30 like I was supposed to for spaces between the words. Why are they all run together? When using strings, the computer neither assumes nor furnishes spaces, unless you put them in. There are good reasons for this. As an example, enter and RUN these lines:

```
10 A$ = "GROSS RECEIPTS FOR 1979 ARE $"
20 INPUT"ENTER GROSS RECEIPTS FOR 1979";R
30 PRINT A$;R
```

Why is there a space between the dollar sign and the amount? I didn't want it there. How can I get rid of it? When the TRS-80 prints a number, it allows a space for the sign of the number. Numbers without signs are assumed to be positive, and we use a sign only if the number is negative.

The computer follows the same rules. Our unwanted space was provided for a sign that we didn't need. To get rid of it, edit lines 20 and 30 to read R$ instead of R. Now we get the first digit snuggled up to the dollar sign like we wanted. Putting the number in a string eliminates the space for the sign, because the number is now regarded as a *symbol* and not a *value*.

Of course, now we cannot do any computation with the number, but we'll get around to that. By not furnishing spaces, we are allowed more flexibility in building strings. Here is a simple example of the potential acquired by using concatenation and space allotment:

```
10 READ A$,B$,C$,D$,E$,F$
20 DATA TRS,-,80,COM,PUT,ING
30 PRINT A$+B$+C$+" "+D$+E$+F$
```

Notice the " " in line 30. That space is the only one I wanted in the string.

We are ready to get our JANUARY, FEBRUARY, MARCH problem unraveled. Remember how they were all run together? There are two ways to fix this. The first is to include the spaces as part of the string so that line 20 reads:

```
20 DATA JANUARY," FEBRUARY"," MARCH"
```

February and March need quotes and not January, because of the leading spaces. An alternate method to handle this, instead of quote marks, is with concatenation. Change line 30 to read:

```
30 PRINT A$+" "+B$+" "+C$
```

We have accomplished the same goal. This short example merely scratches the surface of what can be done with the various string handling instructions and routines furnished by Level II.

No Arguments

There is one last thing that should be cleared up before moving on. The CLEAR instruction has two uses. Used without an argument (with no number following it) it zaps all data out of variable locations and sets all strings to null (erases them).

Used with an argument, as in CLEAR 1000, it still zeros all memory locations containing data, but in addition it sets aside 1000 bytes of space for strings in memory. When the TRS-80 is powered up, it automatically reserves 50 bytes for strings, and if you attempt to enter more than that you will get an ?OS (out of string space) error message.

When you need more than 50 bytes you must reserve such space with the CLEAR(n) instruction, preferably at the very beginning of the program. Should the computer encounter a CLEAR after data has been entered, all such data will be lost. And that can get irritating—fast.

Strings may be tested or compared, just as numbers can in Level II. This makes possible such operations as alphabetizing or searching files for a particular name. This process normally involves using the IF-THEN instruction and what are called the "relational operators." There are six of these operators: equal to, not equal to, greater than, less than, greater than or equal to and less than or equal to.

In string comparisons, these operators work exactly as they do for numerical operations. This brings up an interesting question. We know that the TRS-80 operates on numerical data and that before numbers can be operated on, they must be converted to base 2 or binary numbers. So, how can it tell the difference between A and B or know that the name Brown should come before the name Williams? For an answer we must digress for a moment and take a quick look at what are called the ASCII codes.

The letters ASCII stand for American Standard Code for Information Interchange. Looking at pages C/1 and C/2 of your Level II manual, you see that certain control operations (page C/1) and all signs, symbols, letters and digits (page C/2) are assigned a code number. The dollar sign, for instance, is assigned code # 36, uppercase letters numbers 65 through 90, lowercase letters numbers 96 through 127, digits 0 through 9 numbers 48 through 57, and the graphics characters numbers 129 through 191.

These code numbers were assigned to assist in transferring information between various pieces of equipment, but they also carry some fringe benefits. The computer cannot tell the difference between G and M, but it can differentiate between 71 and 77. The ASCII code numbers are compared when working with alphanumeric information.

To compare the letters G and M for alphabetical precedence, the BASIC language looks at the ASCII codes and not the letters themselves. Since the letters' codes are in ascending numerical order it simply finds the lower number. The computer then sends, on the basis of some test, execution to a specified location or address.

If the strings to be compared contain more than one character each, the computer begins by comparing the first character in the strings, then the second, third and so on, until a match is either found or not. By using multiple IF-THEN tests, very intricate directions can be given; for example, if the first letter precedes the second, go to location A, if the letters are the same, go to location B, and if the second precedes the first, go to location C. Let me stress that strings must be *identical* to be

FROM PROGRAMMA
HI-RESOLUTION GRAPHICS FOR THE TRS-80®

LOWER CASE
The 80-GRAFIX board includes two sets of lower case characters at no additional cost.

INVERSE VIDEO
The 80-GRAFIX board allows you to do inverse video to high-light your screen displays.

FINALLY, AT LAST...

HI- RESOLUTION GRAPHICS is available for your TRS-80 computer system. The 80-GRAFIX board from PROGRAMMA International, Inc. gives your TRS-80 high resolution capability that is greater than the Commodore CBM/PET or even the revered APPLE II.

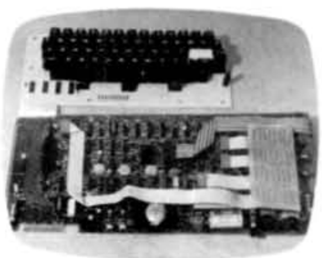
80-GRAFIX gives the TRS-80 an effective screen of 384X192 pixels, versus the normal 127X192 for the TRS-80, 80X50 for the CBM/PET, or the 280X192 of an APPLE II. As an added feature, 80-GRAFIX offers you lower case characters at no additional cost. Of course, you can also create your own set of up to 64 original characters using the supplied Character Generator software.

The 80-GRAFIX board is simple to install (note that this voids your Radio Shack warranty), and programming is done through BASIC. 80-GRAFIX opens up a whole new realm of software development and excitement never dreamed of for the TRS-80!

DEMONSTRATION PROGRAMS
The 80-GRAFIX board is supplied with a Character Generator software and several demonstration programs.

CHARACTER GENERATOR
The supplied character generator software allows you to create your own character set of up to 64 original characters.

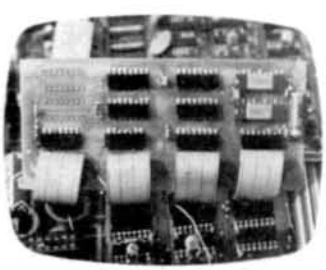
REAL-TIME GRAPHIC GAMES
With the 80-GRAFIX board you can write exciting real-time games using BASIC.



EASY INSTALLATION
The 80-GRAFIX board is simple to install and fits inside the TRS-80 case.

GRAPHICS GALORE
The 80-GRAFIX board and the supplied Character Generator allow you to become an artist.

ELECTRONIC DESIGN
The 80-GRAFIX board has unlimited application in Electronic design and Education.



80-GRAFIX HI-RESOLUTION
Finally, the only means to protect your computer investment is to order an 80-GRAFIX board TODAY!

EXCITEMENT & FUN
Open up a new realm of software development with the 80-GRAFIX board.

Available exclusively through PROGRAMMA at the cost of $149.95
Please check with us for availability prior to ordering
VISA and MASTERCHARGE accepted
TRS-80 is a registered trademark of the Tandy Corp.

✓21

PROGRAMMA INTERNATIONAL, INC.
3400 Wilshire Blvd.
Los Angeles, CA 90010
(213) 384-0579 · 384-1116 · 384-1117

```
10 DEFSTR A,B:DEFINT I,J
20 FOR J=1 TO 7:READ A(J):NEXT
30 DATA B,E,A,D,G,C,F
40 F=0
50 FOR I=1 TO 6
60 IF A(I) <= A(I+1) THEN 90
70 B=A(I):A(I)=A(I+1):A(I+1)=B
80 F=1
90 NEXT
100 IF F=1 THEN 40
110 CLS:PRINT"HERE ARE THE LETTERS
     IN ORDER:"
120 FOR I=1 TO 7:PRINT A(I)+" ";:NE
     XT
```

Listing 1

matched, down to the last space or period. In a long string one character can prevent a match-up.

Applications

Program Listing 1 illustrates a very simple sort routine that takes seven letters and alphabetizes them. In computer jargon a sort is a program which can put data into some specified order. There are numerous sorting routines, including Shell sorts, bubble sorts, ripple sorts, quicksorts and binary search sorts. All depend on string comparisons and branch instructions.

Line 10 of the routine designates A and B as string locations and I and J as integer locations. Sorts run a little faster if the loop counters are defined as integers, as this eliminates some of the computation necessary for single precision numbers. Lines 20 and 30 read data into the computer using an array loop.

Lines 40 and 80 have to do with setting and resetting a flag in the variable location F. This flag defines an escape route from the loop when all of the data is in alphabetical order.

The heart of the program is the loop in lines 50 through 90. After the flag is set to zero in line 40, line 50 establishes the loop and sets I to 1. Line 60 compares the letter in A(1) with the letter in A(2). Remember, the letters are not being compared, but rather their ASCII numbers. If the letters are in the right order, execution branches to line 90, then to line 50, which increments I by one and then tests to see if the letters in A(2) and A(3) are in the right order.

If all of the letters had been in the right order originally, this process would continue until I had been incremented to seven, dropping execution to line 100. Since the flag is still set at zero, the test fails, and lines 110 and 120 are executed to print the results.

If, however, line 60 uncovers two letters which are not in order, execution proceeds to line 70. Let's say that the letters in A(3) and A(4) were not in proper order. The computer then puts the letter in A(3) into a temporary location named B, transfers the letter in A(4) to A(3), and then puts the original letter which is stored in B into A(4). The two letters are now reversed in position. The letter which was in A(3) is still in B, but will either be written over or ignored.

After line 70 has completed the exchange, line 80 sets the flag to 1, and line 90 returns execution to line 50, which increments I and starts the loop cycling again. The 1 in location F told the computer that a change had been made and that it was not finished. This continues until all letters are in order, at which time line 60 sends execution to line 90, until the loop completes and line 100 sends execution to line 40, resetting the flag.

On the next pass the computer escapes to lines 110 and 120 for printout. One of the disadvantages of this routine is operating time. With only seven letters it loads, sorts, and prints out rather quickly. It takes more time relative to the elements increasing in number and/or complexity.

The ability to compare strings is very handy in locating a particular name or file stored in an array. Program Listing 2 shows an elementary example. Line 10 defines variables by type. Lines 20 to 40 load five names into an array named N. Line 60 allows you to enter the name to be found and stores that name into location A($). Lines 70 through 90 set up a loop and compare what is in A to each member of the arrays. If it cannot find the name, line 100 sets up a default instruction and returns you to line 60 for another try. If the name is found in line 80, execution branches to line 110 to return the file number. Any other information that was in that file could, of course, be retrieved at this point.

Try adapting or changing these routines to accomplish a specific task. A good exercise is to combine Listing 1 with Listing 2 to allow you to enter names in any order. Then have the computer alphabetize them and search for a particular name. As you gain fluency in handling strings, you will be amazed at what this little machine can do. ■

```
10 DEFINT I:DEFSTR A,N
20 FOR I=1 TO 5
30 READ N(I)
40 DATA BROWN T.,CARVER J.,FRANK J.,
    MILLER J.,WILLIAMS B.
50 NEXT
60 INPUT"ENTER NAME YOU WANT FOUND"
    ;A
70 FOR I=1 TO 5
80 IF A=N(I) THEN 110
90 NEXT
100 CLS:PRINT"THE NAME YOU SPECIFIE
     D IS NOT LISTED":GOTO 60
110 CLS:PRINT"THE NAME YOU SPECIFIE
     D IS IN FILE #";I
```

Listing 2

Use your input buffer to beat the SYSTEM.

Free Space

David Cornell
335 Parkside Rd.
Harrington Park, NJ 07640

Sooner or later you'll get an out of memory message from your computer. Then what?

The simplest way to increase space is to use a utility program that deletes spaces, line feeds and remarks. There is, however, an interesting catch: When the utility is most needed, there is no place to put it. Having it permanently reside in memory only contributes to the problem and may be the cause of the OM ERROR in the first place.

The solution is to load this utility into the input buffer, as this part of RAM is always available. The program can be loaded and run after an out of memory message is received without disturbing the BASIC program.

This also means that this utility program must be executed immediately after it has been loaded and can be used only once per load.

As a practical matter, this presents no problem. A utility of this type is not needed continually. It can be loaded and run when needed and will take up no memory when it is not.

Clever Idea

Using part of the buffer in this manner is such a clever idea that the SYSTEM command uses it for its stack area during system loads—so much for originality. The utility, therefore, must straddle the SYSTEM stack to avoid being wiped out by it.

When the tape has loaded, its execution follows directly. There is no prompt for a / or entry point.

The program offers one user option: Remarks may be retained or deleted. Before the program executes, a prompt is displayed and the option is selected. For purposes of debugging or documentation, it may be desirable to retain remarks; in some programs a remark may be the target of a GOTO or a GOSUB.

Upon completion, the number of bytes deleted from the BASIC program will be displayed at the upper left of the screen.

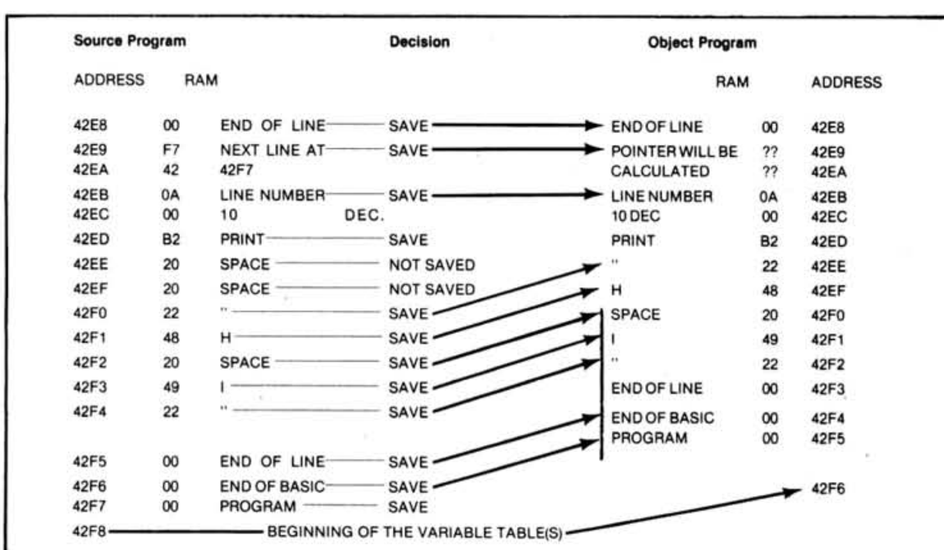
The discussion has been directed toward a Level II system. But it applies as well to a disk-based system, with the exception that there are other areas of RAM available in disk systems.

BASIC

Basic Commands are stored in RAM as one-byte tokens. When a BASIC program line is keyed, it is first written in the input buffer. Then, routines in ROM rewrite the line, substituting one-byte tokens for each reserved word found. Numbers 80H and above are used. These will be greater than the ASCII character set, and when the BASIC interpreter encounters a number in this range, it knows that it has found a command.

The end of line marker is 0. Zeros never appear in the text. When the interpreter encounters 0, it knows it has reached the end of the line. (A 0 may, however, be part of a line number or line pointer.)

While you can find the end of the current line by advancing until 0 is encountered, you cannot back up to find the end of the previous line as the 0 encountered could be part of a line num-



Example 1.

ber or pointer.

The first two bytes of a line point to the address of the next line. The next two bytes of the line are the number of the line. A line pointer of 0000 indicates the end of the BASIC program.

The Z-80 stores two-byte numbers, least significant byte first. 40A4 would appear in RAM, or in the assembler program listing, as A4 40. All bytes which have no program function will be deleted. The BASIC program is altered as described in the following text.

Formatting Commands

Any byte in the range 1 to 32 decimal is deleted. These are usually a space (ASCII 32) or a line feed (ASCII 10, entered from the keyboard or by the down arrow). The tab (right arrow), entered when the line is initially keyed, also writes spaces (ASCII 32s) in the line which will be deleted.

Formatting bytes preceded by a DATA token or a comma will be deleted, not saved. All other bytes will be unaffected, or protected.

Remarks can be deleted or saved depending upon user option. The remark begins with the REM or ' command and preceding bytes may be deleted.

There are some special cases. A legal BASIC line with no text will be deleted. For example: 100 :. This would be accepted by BASIC, but does nothing. Superfluous colons will also be deleted, as in the example below:

```
100 FOR I = 1 TO 6::
```

How it Works

The original BASIC program is treated as a source program and is used to write an object program. Each source byte is examined and is either saved or not saved in the object program.

HL is the source program pointer for the byte of the source program which is being examined. DE is the object program pointer that points to the location where the next byte of the object program will be written.

A byte is saved by writing it in the address indicated by the object program pointer and incrementing both the source and object program pointers. A byte is deleted by incrementing the source pointer without writing the byte to the object program.

The BASIC source program, 10 PRINT "H I", produces the object program. Example 1 illustrates the pointers.

The main program is controlled by TEXT. TEXT sets itself as the return address by pushing its address into the stack. On RET, the top of the stack is popped into the Z-80 program counter. This allows the program to be arranged into small callable blocks and it always returns control to the same point after whatever branching takes place.

TEXT then checks for the cases which will be handled by special processing routines: the end of the source program line

BASIC PROGRAM	APPEARS IN RAM AS
REM	93
'	3C 93 FB
10 :'	0A 00 3C 3C 93 FB
10 '	0A 00 3C 93 FB
15360 REM	00 3C 93
15360 '	00 3C 3C 93 FB
15360 :'	00 3C 3C 3C 93 FB
15538 REM	B2 3C 93
PRINT:REM	B2 3C 93
PRINT:'	B2 3C 3C 93 FB

Table 1.

AUTO	Auto execute
START	Start the program
DISPR	Display prompt
DISSTR	Display string (pointed to by HL)
SCAN	Scan the keyboard
REMOPT	Remark option chosen (save remarks)
NOREM	No remarks to be saved
TEXT	Text of BASIC line
SAVE	Save byte in Object Program
NOSAVE	Do not save byte in Object Program
ENDLINE	End of the line (in Source Program)
NOTEND	Not the end of the BASIC program
FIRST	First byte of the text
BEGSMT	Beginning of the BASIC statement
EXITSP	Exit the SPACE program (sorry!)
QUOTE	Quote found in Source Program
REMDEL	Remarks to be deleted
REMSAV	Remarks to be saved
DATA	DATA statement found in BASIC program
PROMPT	Prompt

Table 2. Assembler Mnemonics and Derivations.

```
 1 "

       THIS IS A TEST  FOR THE UTILITY PROGRAM 'SPACE'

"
10 PRINT    "H E L L O": H E L L O
20 PRINT    "HELLO : GOOD-BYE
30 DATA 1, 2, 3:DATA 1 , 2 , 3
40   DATA 1, 2, 3, ":":DATA "T E S T": T E S T
50 REM TEST
60 'TEST
70  'TEST
80 A:'TEST
90 B: 'TEST
100 AA:'TEST
110 BB: 'TEST
120 :
130  :
140 A:REM TEST
150 B: REM TEST
160 AA:REM TEST
170 BB: REM TEST
180 DATA MONDAY,TUESDAY, WEDNESDAY,
       THURSDAY, FRIDAY, SATURDAY,
       SUNDAY
190     X$="
       MONDAY
       TUESDAY
       WEDNESDAY
       THURSDAY
       FRIDAY
       SATURDAY
       SUNDAY"

15360 :'TEST
20000 END
```

Source program.

```
1 "

       THIS IS A TEST  FOR THE UTILITY PROGRAM 'SPACE'

"
10 PRINT"H E L L O":HELLO
20 PRINT"HELLO : GOOD-BYE
30 DATA1,2,3:DATA1 ,2 ,3
40 DATA1,2,3,":":DATA"T E S T":TEST
80 A
90 B
100 AA
110 BB
140 A
150 B
160 AA
170 BB
180 DATAMONDAY,TUESDAY,WEDNESDAY,THURSDAY,FRIDAY,SAT
    URDAY,SUNDAY
190 X$="
       MONDAY
       TUESDAY
       WEDNESDAY
       THURSDAY
       FRIDAY
       SATURDAY
       SUNDAY"
20000 END
```

Object program.

or a quote. If neither is encountered, a check is made for a formatting command.

If the byte is a formatting command, there is a jump to NOSAVE in the object program. If the byte is not, it is considered necessary to the BASIC program and is saved.

RET returns to TEXT to begin the process all over again for the next byte. Note that SAVE and NOSAVE also update the pointers, return the source byte in the accumulator and set the Z flag if the end of the source line has been reached.

Endline

This module is entered if the end of the source line has been reached. On entry A = 0; HL points to the end of the source line; and DE points to the address where the end of the object program line will be written.

A check is also made to determine if this is also the end of the BASIC program. If this is the case, a jump is made to the exit routine, EXITSP.

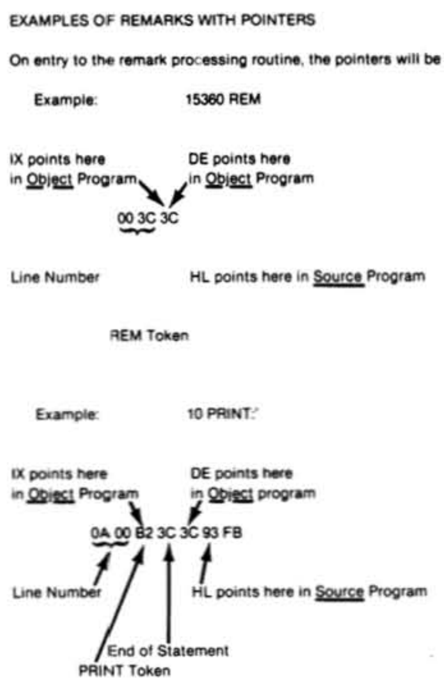
In the first byte of a statement a number of unique situations may occur. These are treated as special cases and handled by BEGSMT. A check is made for DATA and REMark commands, which occur only at the beginning of a statement. Also, a statement may have no text, as mentioned earlier. In this case, it is treated as a remark and is deleted by the remark delete routine. Extra colons may have been entered through user error or keyboard bounce and are not saved. A RET is used again to return to TEXT.

Remarks present a problem. Processing remarks is made complicated by the fact that there are two different remark formats. REM is stored in RAM as 93H (REM token). " ' " is stored in RAM as 3C 93 FB (: REM FB). The colon may also indicate the end of a statement which in turn may precede a REM token.

Things are further complicated by the need to know whether the remark is at the beginning of a line or at the beginning of a statement in a multiple statement line. Once all this has been figured out, the source and object program pointers are adjusted accordingly. Table 1 shows several example remarks as they appear in RAM.

Table 1 also shows the problems of backing up. One solution is to save the address of the first byte of the object text in register pair IX. If, on entry to the remark processing routine, DE = IX, then the remark processing routine, DE = IX, then the remark is at the beginning of the line. If DE is not equal to IX, then the remark is in a statement in a multiple statement line. Note that the colons as the first character of the text will not be saved. Example 2 shows remarks with pointers.

In addition to containing the display message SAVE REMARKS Y/N ?, the prompt string contains a number of ASCII control codes. When the prompt is



Example 2. Remarks with pointers

```
42D1 53 S    15600     DEFM    'SAVE REMARKS Y/N ?'
42D2 41 A
42D3 56 V
42D4 45 E
42D5 20
42D6 52 R
42D7 45 E
42D8 4D M
42D9 41 A
42DA 52 R
42DB 4B K
42DC 53 S
42DD 20
42DE 20
42DF 59 Y
42E0 2F /
42E1 4E N
42E1 20
42E3 3F ?
42E4 00      15700     DEFB    0       ;END OF MESSAGE MARKER

00000 TOTAL ERRORS
```

Example 3. Hexadecimal Representation of ASCII Characters.

Program Listing

```
40A4               BPRPTR   EQU    40A4H          ;BASIC PROGRAM POINTER
0093               REMTOK   EQU    93H            ;REMARK TOKEN
0088               DATTOK   EQU    88H            ;DATA TOKEN
40F9               SCLRPT   EQU    40F9H          ;POINTER TO VARIABLE TABLE (SCALARS)
0021               FORMAT   EQU    ' '+1          ;ASCII CONTROL CODE & SPACE, 1 TO 32 DEC.
01C9               CLS      EQU    01C9H          ;CLS ROUTINE IN ROM
28A7               DISSTR   EQU    28A7H          ;DISPLAYS STRING AT HL UNTIL BYTE=0
0FAF               DISNUM   EQU    0FAFH          ;DISPLAYS DECIMAL REPRESENTATION OF NUMBER

41E2                        ORG    41E2H
41E2 C3E641                 JP     AUTO

41E6                        ORG    41E6H

41E6 3EC9          AUTO     LD     A,0C9H         ;'RETURN' COMMAND
41E8 32E241                 LD     (41E2H),A      ;   TO RAM
41EB 21CB42        DISPR    LD     HL,PROMPT      ;POINT TO MESSAGE
41EE CDA728                 CALL   DISSTR         ;DISPLAY MESSAGE
41G1 CD4900        SCAN     CALL   0049H          ;SCAN UNTIL KEYBOARD INPUT
41F4 FE4E                   CP     'N'            ;="N" ?
41F6 2809                   JR     Z,NOREM        ;YES, DELETE REMARKS
41F8 FE59                   CP     'Y'            ;="Y" ?
41FA 20F5                   JR     NZ,SCAN        ;NO, UNACCEPTABLE INPUT
41FC 3E52                   LD     A,REMSAV-REMJP-2      ;JR OFFSET
41FE 324742        REMOPT   LD     (REMJP+1),A    ;MODIFY PROGRAM
4201 CDC901        NOREM    CALL   CLS            ;CLEAR SCREEN
4204 2AA440                 LD     HL,(BPRPTR)    ;POINT TO BASIC PROGRAM
4207 2B                     DEC    HL             ;
4208 E5                     PUSH   HL             ;
4209 D1                     POP    DE             ;HL AND DE POINT BEG. PROG. -1
420A AF                     XOR    A              ;SET AS FOR END OF LINE

420B E5            TEXT     PUSH   HL             ;SET 'TEXT'
420C 210B42                 LD     HL,TEXT        ;  AS
420F E3                     EX     (SP),HL        ;      RETURN ADDRESS
4210 B7                     OR     A              ;A = 0  ?
4211 2817                   JR     Z,ENDLNE       ;YES, END OF LINE
4213 FE3A                   CP     ':'            ;= ':' ?
4215 2005                   JR     NZ,TEXT1       ;NO, NOT END OF STATEMENT
4217 CD2442                 CALL   SAVE           ;SAVE END OF STATEMENT MARKER
421A 1822                   JR     BEGSMT         ;CHECK FIRST BYTE OF TEXT
421C FE21          TEXT1    CP     FORMAT         ;SPACE OR LINE FEED ?
421E 3806                   JR     C,NOSAVE       ;YES, DO NOT SAVE
4220 FE22                   CP     '"'            ;QUOTE ?
4222 283F                   JR     Z,QUOTE        ;YES, JUMP TO QUOTE ROUTINE

4224 12            SAVE     LD     (DE),A         ;BYTE TO OBJECT PROGRAM
4225 13                     INC    DE             ;INC. OBJ. PROG. POINTER
4226 23            NOSAVE   INC    HL             ;INC. SOURCE PROG. POINTER
4227 7E                     LD     A,(HL)         ;SOURCE BYTE TO A
4228 B7                     OR     A              ;RETURNS Z FLAG IF END OF LINE
4229 C9                     RET                   ;RETURN

422A E5            ENDLNE   PUSH   HL             ;END
422B 23                     INC    HL             ;  OF
422C 7E                     LD     A,(HL)         ;     BASIC
422D 23                     INC    HL             ;        LINE
422E B6                     OR     (HL)           ;          ?
422F E1            NOTEND   POP    HL             ;.       FIVE BYTES
4230 010500                 LD     BC,05          ;.         TO
4233 EDB0                   LDIR                  ;.       OBJECT PROGRAM
4235 281A                   JR     Z,EXITSP       ;END OF PROGRAM, EXIT
4237 D5                     PUSH   DE             ;SAVE ADDRESS OF
4238 DDE1                   POP    IX             ;  OBJECT PROG. TEXT
423A 2B                     DEC    HL             ;FUDGE FACTOR
423B CD2642        FIRST    CALL   NOSAVE         ;INC SOURCE PROGRAM POINTER
423E 2849          BEGSMT   JR     Z,REMDEL       ;NO TEXT, DELETE LINE
4240 FE3A                   CP     ':'            ;= ':' ?
4242 28F7                   JR     Z,FIRST        ;YES, DO NOT SAVE
4244 FE93                   CP     REMTOK         ;IS IT A REMARK ?
4246 2841          REMJP    JR     Z,REMDEL       ;YES, JUMP TO REMARK PROCESSING
                                                  ROUTINE
4248 FE21                   CP     FORMAT         ;SPACE OR LINE FEED ?
424A 38EF                   JR     C,FIRST        ;YES, DO NOT SAVE IN OBJ.PROGRAM
424C FE88                   CP     DATTOK         ;DATA STATEMENT ?
424E 2859                   JR     Z,DATA         ;YES, JUMP TO PROCESSING ROUTINE
4250 C9                     RET                   ;RETURN

4251 1B            EXITSP   DEC    DE             ;POINT DE TO NEW
4252 1B                     DEC    DE             ;       VARIABLE (SCALAR) POINTER
4253 D5                     PUSH   DE             ;SAVE ADDRESS OF NEW SCLRPTR
4254 2AF940                 LD     HL,(SCLRPT)    ;OLD POINTER TO HL
4257 ED52                   SBC    HL,DE          ;COMPUTE DIFFERENCE
4259 CDAF0F                 CALL   DISNUM         ;DISPLAY NUMBER
425C CDFE20                 CALL   20FEH          ;CURSOR TO NEXT LINE
425F E1                     POP    HL             ;NEW POINTER TO HL
4260 C3772C                 JP     2C77H          ;INITIALIZE FOR BASIC

4263 CD2442        QUOTE    CALL   SAVE           ;SAVE (OPENING '"' ON ENTRY)
4266 C8                     RET    Z              ;RETURN IF END OF LINE
4267 FE22                   CP     '"'            ;CLOSING '"'?
4269 20F8                   JR     NZ,QUOTE       ;NO, SAVE BYTE
426B C32442                 JP     SAVE           ;SAVE CLOSING '"'
```

Program continues

printed by routines in BASIC ROM, these are recognized as control codes and the appropriate function is performed.

The first six bytes of the prompt would be duplicated in BASIC by:

```
PRINT CHR$(28);CHR$(31);:
PRINT CHR$(13);CHR$(13);CHR$(13);:
  (or PRINT:PRINT:PRINT)
PRINT TAB(16);
```

The ASCII control codes are described on page C/1 of the *TRS-80 Level II Manual*.

Auto Execute

After the tape has been loaded, control goes directly to the program. The system command calls 42E2. BASIC initializes this as a RET. Using a new ORG, the system tape changes this to a jump to assembler routine AUTO. AUTO restores the return command so that system will not attempt a disasterous jump to this program the next time it is invoked.

For those interested in understanding the program in more depth, the assembler listing is commented and, in addition to the usual flow charts, an "organization" chart has been provided.

The program uses a number of routines in BASIC ROM. For the purposes of this article, they are treated as "black boxes." However, by reading the comments and seeing how they have been used their function and application may be understood.

T-BUG and Space

Although the assembler listing uses multiple origins and the

```
# M 41E2 C9 C3
41E3 00 E6
41E4 00 41
41E5 3A 3A
41E6 B3 3E
41E7 48 C9
41E8 00 32
41E9 19 E2
41EA 80 41
41EB 44 21
41EC C3 CB
41ED 63 42
41EE 0B ETC.....
```

Example 4.

```
                ;NEW ORG IF LOADED INTO BUFFER AREA
                ;SYSTEM COMMAND USES 4287 TO 426E FOR STACK
                ;ON ENTRY TO 'SYSTEM', SP SET TO 4288
4289                            ORG     4289H
4289 DDE5       REMDEL  PUSH    IX              ;IX TO HL
428B E3                 EX      (SP),HL         ;  HL TO STACK
428C DF                 RST     18H             ;HL = DE ?
428D E1                 POP     HL              ;RESTORE HL
428E 1B                 DEC     DE              ;DEC OBJ PROG POINTER
428F 2004               JR      NZ,REM1         ;REM IS NOT AT BEGINNING OF LINE
4291 1B                 DEC     DE              ;REM IS AT BEGINNING OF LINE
4292 1B                 DEC     DE              ;  DEC OBJECT PROG. POINTER
4293 1B                 DEC     DE              ;    TO END OF
4294 1B                 DEC     DE              ;         PREVIOUS LINE
4295 AF         REM1    XOR     A               ;COMPARISON BYTE FOR SEARCH
4296 EDB1               CPIR                    ;SEARCH
4298 2B                 DEC     HL              ;POINT TO END OF LINE
4299 C9                 RET                     ;RETURN
429A 23         REMSAV  INC     HL              ;WHICH
429B 7E                 LD      A,(HL)          ;  FORM
429C 2B                 DEC     HL              ;    OF
429D FEFB               CP      0FBH            ;       REMARK ?
429F 2001               JR      NZ,REMSA0       ;SIMPLE  'REM', NOT " ' "
42A1 2B                 DEC     HL              ;POINT TO PRECEEDING ':'
42A2 7E         REMSA0  LD      A,(HL)          ;BYTE TO A
42A3 CD2442             CALL    SAVE            ;SAVE UNTIL
42A6 20FA               JR      NZ,REMSA0       ;  END OF LINE
42A8 C9                 RET                     ;RETURN
42A9 062C       DATA    LD      B,','           ;SET FOR "OK TO DELETE FORMAT"
42AB CD2442     DATA1   CALL    SAVE            ;SAVE DATA TOKEN
42AE C8         DATA2   RET     Z               ;END OF LINE
42AF FE22               CP      '"'             ; = '"'
42B1 CC6342             CALL    Z,QUOTE         ;YES, PROCESS QUOTE
42B4 B7                 OR      A               ;END OF LINE ?
42B5 C8                 RET     Z               ;YES, RETURN
42B6 FE3A               CP      ':'             ;END OF STATEMENT ?
42B8 C8                 RET     Z               ;YES, RETURN
42B9 FE21               CP      FORMAT          ;SPACE OR LINE FEED ?
42BB 300B               JR      NC,DATA4        ;NO, SAVE BYTE
42BD 78                 LD      A,B             ;PREVIOUS OBJECT
42BE FE2C               CP      ','             ;  BYTE A ',' ?
42C0 2005               JR      NZ,DATA3        ;NO, SAVE BYTE
42C2 CC2642             CALL    Z,NOSAVE        ;YES, OK TO DELETE
42C5 18E7               JR      DATA2           ;NEXT DATA BYTE
42C7 7E         DATA3   LD      A,(HL)          ;RESTORE CURRENT BYTE
42C8 47         DATA4   LD      B,A             ;SAVE FOR COMPARISON
42C9 18E0               JR      DATA1           ;NEXT BYTE
42CB 1C         PROMPT  DEFB    28              ;HOME CURSOR
42CC 1F                 DEFB    31              ;CLEAR TO END OF FRAME
42CD 0D                 DEFB    13              ;CURSOR TO NEXT LINE
42CE 0D                 DEFB    13
42CF 0D                 DEFB    13
42D0 D0                 DEFB    208             ;TAB(16)
42D1 53                 DEFM    'SAVE REMARKS  Y/N ?'
42E4 00                 DEFB    0               ;END OF MESSAGE MARKER
0000                    END
```

```
AUTO    41E6 01500   00200
BEGSMT  423D 06600   04100
BPRPTR  40A4 00600   02700
CLS     01C9 01100   02600
DATA    42A9 12500   07400
DATA1   42AB 12600   14300
DATA2   42AE 12700   14000
DATA3   42C7 14100   13800
DATA4   42C8 14200   13500
DATTOK  0088 00800   07300
DISNUM  0FAF 01300   08200
DISPR   41EB 01700
DISSTR  28A7 01200   01800
ENDLNE  422A 05400   03700
EXITSP  4250 07700   06100
FIRST   423A 06500   06800 07200
FORMAT  0021 01000   04200 07100 13400
NOREM   4201 02600   02100
NOSAVE  4226 04900   04300 06500 13900
NOTEND  422E 05800
PROMPT  42CB 14500   01700
QUOTE   4262 08700   04500 09000 12900
REM1    4295 10900   10400
REMDEL  4289 09900   06600 07000
REMJP   4245 07000   02400 02500
REMOPT  41FE 02500
REMSA0  42A2 12000   11800 12200
REMSAV  429A 11400   02400
REMTOK  0093 00700   06900
SAVE    4224 04700   04000 08700 09100 12100 12600
SCAN    41F1 01900   02300
SCLRPT  40F9 00900   08000
TEXT    420B 03300   03400
TEXT1   421C 04200   03900
```

program lies in noncontiguous memory, it may be entered by T-BUG or another monitor program.

At the time when the byte read from the tape is written into RAM, the stack is at its highest position, and therefore no damage will be done. Bytes will be loaded into the stack area, but will immediately be overwritten by the stack operations of the system load.

The assembler does not print out the ASCII values for DEFM commands. These values are given in Example 3.

After the program has been entered, punch a tape with P 41E2 42E4 0000 SPACE (enter). Any value may be entered for the entry point as it is never used.

If all of this is still a little new to you, compare Example 4 with the assembler listing. This will show how the two relate and how to enter the listing with T-BUG.

Be careful that 41E5 is not changed. The column on the right is the program from the assembler listing. The column in the middle is whatever was in the input buffer.

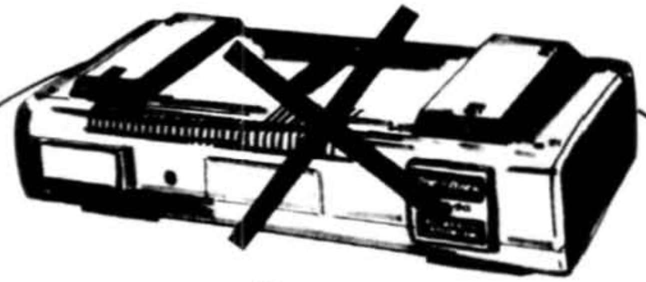
Disk BASIC and Space

This program is intended primarily for the Level II user. It is, however, currently being used with a recently acquired disk operating system with no noticeable ill effects.

However, 41E6 to 42E7 is no longer the input buffer area. I have no idea what it becomes. The buffer may be found by looking in 40A7 and 40A8. These two bytes hold the address of the input buffer.

Three bytes of RAM will be changed. In Level II, 41E2 is a RET (c9). In NEWDOS, it is a jump to another address which is a RET. Changing 41E2 to RET, obviously, performs the same function from a different address. This change has yet to cause trouble.

With an input buffer, an overlay area and a number of disk input buffers to play with, other areas of RAM could be called into use for this utility; and, the auto execute feature could be removed. ■



TRS-80* interfacing alternatives...

Featuring the Quality Engineering and Manufacturing of THE MICRO MINT

COMM-80

- FULL 8-BIT PARALLEL PORT
- RS-232-C SERIAL PORT (up to 19,200 baud)
- UP TO 16 UNITS CAN BE CHAINED TOGETHER
- TRS-BUS CONNECTOR FOR FUTURE EXPANSION
- CONNECTS TO KEYBOARD OR E.I.
- INCLUDES TERMINAL SOFTWARE

ONLY $179.95

CHATTER BOX

- FULL 8-BIT PARALLEL PORT
- RS-232-C SERIAL PORT (up to 19,200 baud)
- ACOUSTIC MODEM — 300 BAUD
- TRS-BUS CONNECTOR FOR FUTURE EXPANSION
- CONNECTS TO KEYBOARD OR E.I.
- CAN BE CHAINED TOGETHER WITH COMM-80
- INCLUDES TERMINAL SOFTWARE

ONLY $259.95

DISK-80

- DISK CONTROLLER (up to 4 drives)
- DATA SEPARATOR INCLUDED
- INCLUDES 16K OF RAM
- PROVISION FOR AN ADDITIONAL 16K RAM
- TRS-BUS CONNECTOR FOR FUTURE EXPANSION

ONLY $299.95

BUSY BOX

- ALLOWS TRS-80 TO CONTROL AC APPLIANCES
- CABLE AND CONNECTOR FOR TRS-80 KEYBOARD OR E.I.
- WORKS WITH THE POPULAR BSR X-10 SYSTEM
- INCLUDES BASIC LISTING FOR CONTROL ROUTINE
- LEVEL II MINIMUM SYSTEM

ONLY $109.95

ALL INTERFACES ARE RADIO SHACK HARDWARE AND SOFTWARE COMPATIBLE.

MICRO MINT

↙310

MAIL ORDER OR INFORMATION
WRITE OR CALL: (516) 374-6793
The MicroMint Inc.
917 Midway
Woodmere, NY 11598
Dealer inquiries invited.

CALL OUR QUICK TOLL FREE NUMBER:
1-800-258-1790

*TRS-80 is a trademark of Tandy Corp.

Single stroke entry of BASIC keywords.

Uni-key

Rowland Archer Jr.
Flint Ridge Apt. 59
Hillsborough, NC 27278

One night while I was trying to massage some life into my tired fingers after a couple of hours at the keyboard, it occurred to me that typing programs is the sort of drudgery a computer is supposed to take out of life, not put into it. Typing a BASIC program repeats many of the same BASIC keywords over and over again.

To pass some of this work off on the TRS-80, I needed a way to let it know which keyword I wanted with a single-key abbreviation. Using computerized keywords would also cut down on syntax errors and tedious editing.

Lowercase ASCII Code

Even though BASIC is uppercase only on the TRS-80, the keyboard will generate lowercase ASCII character codes. Try this short BASIC program and see what happens:

```
10 CLS:
20 A$ = ""
30 A$ = INKEY$: IF A$ = "" THEN GOTO 30
40 PRINT@0,ASC(A$);: GOTO 20
```

Run the program and press any alphabetic key, say "A." The ASCII code for uppercase A, 65, should appear on the screen. Now press SHIFT A; the code printed should be 97, which is lowercase a.

To get lowercase letters on a TRS-80, you press the SHIFT key. Although it may seem backwards to shift for lowercase, would you rather have to shift for uppercase? You would have to hold down the SHIFT key to enter every BASIC keyword. A SHIFT-lock key would get around this problem, but apparently Radio Shack didn't feel the need for one. This article, however, will relate a software fix.

A routine, which examines every pressed key before the character value is returned to the BASIC interpreter program, is necessary. When the TRS-80 appears to be doing nothing, it is actually reading the keyboard over and over, waiting for a key to be pressed.

Assuming it is possible to intercept characters from the keyboard and look them over before they reach BASIC; then it is possible to decide whether to send the character on to BASIC as is, send back some other character instead, or even send BASIC a stream of two or more characters in its place.

To accomplish this, it is necessary to install a filter between the keyboard and the BASIC interpreter. This would be a device whose action filters input data streams to produce output data.

We need a filter which translates some input characters to BASIC keywords, and leaves others alone. If the character we intercept from the keyboard is an uppercase letter, a number or a special symbol (@, <, +, etc.), our program should pass the character on to the caller unchanged.

However, if it is a lowercase letter (a-z), which is transmitted when the user hits the SHIFT key and a letter, the filter should

Input	ASCII Value	Output
Special characters	0-25	Same as input
SHIFT/Down Arrow	26	Substitute user-defined string (up to 64 chars)
Special characters	27-30	Same as input
SHIFT/CLEAR	31 & Shift*	None (Start/end user string definition)
Special characters and numbers	32-64	Same as input
Uppercase letters	65-90	Same as input
Special characters	91-96	Same as input
Lowercase letters	97-122	Substitute a BASIC keyword
Special characters	123-127	Same as input

*Since the keyboard driver returns 31 for both SHIFT/CLEAR and CLEAR, we must test for the Shift key separately.

Table 1. Function of keyboard filter routine.

Input	ASCII Value	Output
Special characters	0-25	Same as input
Shift-Down Arrow	26	Substitute user-defined string (up to 64 chars)
Special characters	27-30	Same as input
Shift-Clear	31 & Shift*	None (Start/end user string definition)
Special characters	32-64	Same as input

RAM Size	ORG of Program	Answer MEMORY SIZE?
16K	7E58H	32361
32K	BE58H	48745
48K	FE58H	65129

Table 2. Origin of program and MEMORY SIZE for different RAM sizes.

SHIFT for Keywords:

A	PRINT@	N	NEXT
B	ELSE	O	POKE
C	CHR$(	P	PEEK(
D	DATA	Q	LEFT$(
E	RIGHT$(	R	RETURN
F	FOR	S	GOSUB
G	GOTO	T	TAB(
H	RND(	U	USING
I	INPUT"	V	STRING$(
J	READ	W	MID$(
K	INKEY$	X	SET(
L	LEN(	Y	THEN
M	ASC(	Z	RESET(

CLEAR—START/END Definition
Down Arrow—User String

Table 3. Reference chart to tape on your monitor.

Program Listing

```
                 ; ONESTR - ONE STROKE KEYWORD ENTRY PROGRAM.  INTERCEPTS
                 ;         LOWER-CASE CHARACTERS AND REPLACES THEM WITH
                 ;         KEYWORD STRINGS.  ALSO ALLOWS THE USER TO ASSIGN A
                 ;         STRING OF UP TO 64 BYTES TO 'SHIFT-DOWN ARROW'.
                 ;         DEFINITION OF THIS STRING IS INITIATED AND
                 ;         TERMINATED BY 'SHIFT-CLEAR'.
                 ;
                 ;         REV 2.2          2/9/79
                 ;
                 ;         BY ROWLAND ARCHER
                 ;            FLINT RIDGE 59
                 ;            HILLSBOROUGH, NC  27278
                 ;
4016             KEYDRV  EQU     4016H           ;ADDRESS IN KEYBOARD DCB
                                                 ; OF DRIVER ROUTINE
3880             SHIFT   EQU     3880H           ;(3880H) IS 1 IF 'SHIFT'
                                                 ; KEY IS PRESSED
033A             PUTC    EQU     033AH           ;PUT CHAR IN A ON SCREEN
00CD             CALL    EQU     0CDH            ;'CALL' OPCODE VALUE
001F             DEFKEY  EQU     31              ;'SHIFT-CLEAR' KEY
001A             UDSKEY  EQU     26              ;'RETURN USER-DEF STRING'
                                                 ; KEY = 'SHIFT-DOWN ARROW'
0040             USTLEN  EQU     64              ;USER-DEFINED STRING LENGTH
                 ;
                 ;THE FOLLOWING INITIALIZATION CODE IS PERFORMED ONLY
                 ;WHEN THIS ROUTINE IS LOADED AND RUN THE FIRST TIME.
                 ;
7E58                     ORG     07E58H          ;ORG FOR 16K SYSTEM
7E58 2A1640      INIT    LD      HL,(KEYDRV)     ;GET ADDR OF KEYBRD DRIVER
7E5B 224E7F              LD      (KEYDR1),HL     ; ROUTINE AND BUILD TWO
7E5E 22B07F              LD      (KEYDR2),HL     ; CALL INSTRUCTIONS IN
                                                 ; THIS CODE WITH IT
7E61 21477F              LD      HL,ONESTR       ;NOW PUT THE ADDRESS OF THE
7E64 221640              LD      (KEYDRV),HL     ; ENTRY PT TO THIS ROUTINE
                                                 ; INTO THE KEYBOARD DCB
                 ;*******************************************************
                 ;YOU MUST CHOOSE ONLY ONE OF THE FOLLOWING TWO JUMP
                 ;INSTRUCTIONS TO EXIT THIS INITIALIZATION CODE.
                 ;IF YOU ARE GOING TO LOAD THIS PROGRAM FROM DISK WHILE
                 ;IN DOS, USE
                 ;       JP      402DH           ; TO RETURN TO DOS
                 ;IF YOU ARE GOING TO LOAD FROM TAPE WHILE IN BASIC, USE
7E67 C3191A              JP      1A19H           ; TO RETURN TO BASIC
                 ;*******************************************************
                 ;
                 ; ABOVE CODE IS ONLY USED ONCE AND CAN BE OVERWRITTEN
                 ; AFTER IT RUNS - SO 'MEMORY SIZE?' PROTECTIONS STARTS
                 ; WITH THE FOLLOWING DATA STRUCTURES:
                 ;
7E6A 52          USTR    DEFM    'RUN'           ;USER STRING; INITIALLY
7E6D 0D                  DEFB    0DH             ; 'RUN <ENTER>'
7E6E 00                  DEFB    0               ;END OF STRING
0005             USED    EQU     $-USTR          ;SIZE OF PREDEFINED STRING
003B             REST    DEFS    USTLEN-USED     ;ALLOCATE SPACE FOR REST
                                                 ; OF USER-DEFINED STRING
7EAA 00                  DEFB    0               ;FORCE END OF STRING
7EAB 00          OSFLAG  DEFB    0               ;ONE-STROKE FLAG: = 1 WHILE
                                                 ; WE ARE SUBSTITUTING FOR A
                                                 ; LOWER-CASE CHARACTER
7EAC 0000        OSPTR   DEFW    0               ;ADDRESS OF CURRENT CHAR
                                                 ; IN SUBSTITUTE STRING
                 ;
                 ;TABLE OF STRINGS TO SUBSTITUTE FOR LOWER-CASE CHARS.
                 ;STRINGS ARE TERMINATED BY NULL (0) BYTES.  STRING
                 ;LABELLED 'LA' IS SUBSTITUTED FOR 'SHIFT-A', 'LB'
                 ;FOR 'SHIFT-B', ETC.  EXCEPT FOR 'LA', LABELS
                 ;ARE NOT REQUIRED, AND ARE ONLY INCLUDED FOR EASE IN
                 ;DETERMINING THE STRING TO BE SUBSTITUTED FOR EACH LETTER.
                 ;
7EAE 50          LA      DEFM    'PRINT@'
7EB4 00                  DEFB    0               ;ZERO BYTE END OF STRING
7EB5 45          LB      DEFM    'ELSE'
7EB9 00                  DEFB    0
7EBA 43          LC      DEFM    'CHR$('
7EBF 00                  DEFB    0
7EC0 44          LD      DEFM    'DATA'
7EC4 00                  DEFB    0
7EC5 52          LE      DEFM    'RIGHT$('
7ECC 00                  DEFB    0
7ECD 46          LF      DEFM    'FOR'
7ED0 00                  DEFB    0
7ED1 47          LG      DEFM    'GOTO'
7ED5 00                  DEFB    0
7ED6 52          LH      DEFM    'RND('
7EDA 00                  DEFB    0
7EDB 49          LI      DEFM    'INPUT"'
7EE1 00                  DEFB    0
7EE2 52          LJ      DEFM    'READ'
7EE6 00                  DEFB    0
```

Program continues

replace that letter in the input stream with a BASIC keyword.

To do this, when a lowercase letter is read from the keyboard set a substitution flag in the routine, indicating that keyword substitution has begun. Use the ASCII value of the letter as an index to a table of BASIC keywords. Pass BASIC the first letter of the indexed keyword instead of the lowercase letter read from the keyboard.

When BASIC calls for input from the keyboard again, the routine will note that the substitution flag is set and will *send back the next character of* the keyword, without bothering to look at the keyboard to see if any keys are pressed. This continues until the entire keyword is sent. Then the flag is reset to normal operation until the next lowercase letter code is received. All this happens so quickly that the keyword seems to appear on the screen the instant the key is pressed.

Note that only 26 keywords can be handled with this method. However, it is possible to select a group of keywords which are either frequently used, difficult to type or both. You can experiment by including different keywords until you find the best subset for your needs. I have found the set of 26 chosen here to be very useful.

Intercepting Input

Since the routine will be used while editing BASIC programs, the easiest approach is to write an assembly language routine to look at characters before the BASIC editor scans them.

Take a look at the memory map in the back of your Level II BASIC manual. At location 4015H (hexadecimal) in the BASIC Reserved RAM area, there is a device control block for the keyboard.

Location 4016H is initialized by BASIC with the address of the keyboard driver routine. A call to this routine returns with 0 in the A register if no key is pressed, or the ASCII value of the key if one is pressed (like an assembly language INKEY$ routine).

Load the routine into high

```
7EE7 49          LK      DEFM    'INKEY$'
7EED 00                  DEFB    0
7EEE 4C          LL      DEFM    'LEN('
7EF2 00                  DEFB    0
7EF3 41          LM      DEFM    'ASC('
7EF7 00                  DEFB    0
7EF8 4E          LN      DEFM    'NEXT'
7EFC 00                  DEFB    0
7EFD 50          LO      DEFM    'POKE'
7F01 00                  DEFB    0
7F02 50          LP      DEFM    'PEEK('
7F07 00                  DEFB    0
7F08 4C          LQ      DEFM    'LEFT$('
7F0E 00                  DEFB    0
7F0F 52          LR      DEFM    'RETURN'
7F15 00                  DEFB    0
7F16 47          LS      DEFM    'GOSUB'
7F1B 00                  DEFB    0
7F1C 54          LT      DEFM    'TAB('
7F20 00                  DEFB    0
7F21 55          LU      DEFM    'USING'
7F26 00                  DEFB    0
7F27 53          LV      DEFM    'STRING$('
7F2F 00                  DEFB    0
7F30 4D          LW      DEFM    'MID$('
7F35 00                  DEFB    0
7F36 53          LX      DEFM    'SET('
7F3A 00                  DEFB    0
7F3B 54          LY      DEFM    'THEN'
7F3F 00                  DEFB    0
7F40 52          LZ      DEFM    'RESET('
7F46 00                  DEFB    0
                 ;
                 ;MAIN ROUTINE ENTRY POINT:
                 ;
7F47 3AAB7E      ONESTR  LD      A,(OSFLAG)      ;IF FLAG<>0 WE ARE IN THE
7F4A B7                  OR      A               ;MIDDLE OF A SUBSTITUTION
7F4B 2038                JR      NZ,SUBST        ;CONTINUE SUBSTITUTION
                 ;
                 ;CALL NORMAL ROUTINE TO GET CHARACTER FROM KEYBOARD
                 ;
7F4D CD                  DEFB    CALL            ;BECOMES 'CALL GET-CHAR'
7F4E 0000        KEYDR1  DEFW    0               ; WHEN INITIALIZATION CODE
                                                 ; PUTS ADDRESS OF KEYBOARD
                                                 ; DRIVER ROUTINE HERE
7F50 B7                  OR      A               ;CHARACTER RETURNED IN A
7F51 C8                  RET     Z               ;0 MEANS NO KEY PRESSED,
                                                 ;SO JUST RETURN TO CALLER
                 ;
                 ;A KEY HAS BEEN PRESSED, HANDLE IT IF IT IS 'DEFKEY',
                 ;'UDSKEY' OR LOWER-CASE LETTER; ELSE JUST RETURN IT.
                 ;
7F52 FE1F                CP      DEFKEY          ;DEFINE USER STRING?
7F54 2840                JR      Z,DEFINE        ;YES, GO DO IT
7F56 FE1A                CP      UDSKEY          ;REQUESTING USER STRING?
7F58 2808                JR      Z,SUBMOD        ;YES, START SUBSTITUTION
7F5A FE61                CP      97              ;KEY < LOWER-CASE A?
7F5C D8                  RET     C               ;YES, RETURN UNCHANGED
7F5D FE7B                CP      123             ;KEY < LOWER-CASE Z + 1?
7F5F 3801                JR      C,SUBMOD        ;YES, SUBSTITUTE
7F61 C9                  RET                     ;NO, RETURN UNCHANGED
                 ;
                 ;START NEW SUBSTITUTION - SET OSFLAG = 1,
                 ;SET POINTER TO STRING TO SUBSTITUTE
                 ;
7F62 E5          SUBMOD  PUSH    HL
7F63 FE1A                CP      UDSKEY          ;USER-DEFINED STRING?
7F65 2005                JR      NZ,KEYWRD       ;NO, IT'S KEYWORD
7F67 216A7E              LD      HL,USTR         ;YES, GO SAVE POINTER TO
7F6A 1810                JR      SRCHDN          ; STRING AND SET MODE FLAG
                 ;
                 ;KEY PRESSED WAS A LOWER-CASE LETTER.  SET POINTER TO FIRST
                 ;CHARACTER OF KEYWORD TO SUBSTITUTE FOR IT.
                 ;
7F6C 21AE7E      KEYWRD  LD      HL,LA           ;BASE OF SUBST-STRING TABLE
7F6F D661                SUB     97              ;SUBTRACT ASCII(LOWER-CASE
                                                 ; A) FROM KEY PRESSED
7F71 B7                  OR      A               ;ZERO => LOWER-CASE A
7F72 2808                JR      Z,SRCHDN        ; PRESSED, END SEARCH
7F74 47                  LD      B,A             ;ELSE A HOLDS NUMBER OF
7F75 7E          NXTC    LD      A,(HL)          ; KEYWORDS TO SKIP OVER TO
7F76 23                  INC     HL              ; FIND STRING TO SUBSTITUTE
7F77 B7                  OR      A
7F78 20FB                JR      NZ,NXTC         ;INNER LOOP FINDS NULL
                                                 ; END-OF-STRING BYTES
7F7A 10F9                DJNZ    NXTC            ;OUTER LOOP COUNTS KEYWORDS
                 ;
                 ;END SEARCH - HL HAS POINTER TO DESIRED STRING
                 ;
7F7C 22AC7E      SRCHDN  LD      (OSPTR),HL      ;SAVE POINTER TO STRING
7F7F 3E01        SETMD   LD      A,1             ;SUBSTITUTION MODE STARTS
```

Program continues

memory, where it can be protected from BASIC by answering the MEMORY SIZE? prompt appropriately on startup. When it is first run, it will grab the address of the keyboard driver routine from location 4016H and save it for later use. It then stores the address of its own entry point at 4016H, so that every routine (in ROM or elsewhere) which used to call the keyboard driver directly will now call this routine instead.

The normal keyboard driver is called as a subroutine to read from the keyboard. If no key is pressed, the subroutine will return a zero to the accumulator (register A); in this case we return to the caller without changing a thing.

If a key has been pressed, it is examined as discussed above, and if it is a lowercase alphabetic code, the keyword substitution routine begins.

A User Defined Key

The above technique allows entry of 26 BASIC keywords with a single keystroke, and by itself will save a lot of typing. But many BASIC programs use the same expression over and over again; it would be convenient to enter a phrase once and then recall it with a single keystroke.

To do this, a key must be defined whose substitution value can be changed dynamically, without having to reassemble and reload the assembly language routine. This feature is easily added by declaring one key as the "define user string" key. This routine uses SHIFT/CLEAR. The "substitute user defined string" key is the shifted down arrow.

These keys can be located in the input stream in the same way shifted alphabetic characters are intercepted. When SHIFT/CLEAR is pressed, a START DEFINITION prompt is printed on the screen. Each character typed, up to 64 characters, is saved in memory until SHIFT/CLEAR is hit again. This terminates the definition of the string. END DEFINITION is written on the screen.

Now when SHIFT plus the down arrow are pressed, the defined string is returned. Table 1 gives a summary of the actions the filter routines will perform for the range of possible keyboard inputs.

The origin shown in the Program Listing of ONESTR is for a 16K machine. Table 2 gives the ORG value to substitute, as well as the appropriate answers to the MEMORY SIZE? prompt, for 32K and 48K machines.

At label INIT you will find the initialization code, which must be executed once when ONESTR is loaded. This code retrieves the address of the current keyboard driver routine from the keyboard device control block and stores it after the CALL opcodes at labels KEYDR1 and KEYDR2.

```
7F81 32AB7E    01810          LD      (OSFLAG),A
7F84 E1        01820          POP     HL              ;RESTORE HL
               01830 ;
               01840 ;BRANCH HERE WHEN WE ARE DOING A SUBSTITUTION
               01850 ;
7F85 E5        01860 SUBST    PUSH    HL              ;SAVE HL
7F86 2AAC7E    01870          LD      HL,(OSPTR)      ;GET CURRENT CHARACTER
7F89 7E        01880          LD      A,(HL)          ; OF SUBSTITUTION STRING
7F8A B7        01890          OR      A               ;NULL END-OF-STRING?
7F8B 2003      01900          JR      NZ,NOTEND       ;NO, MORE TO GO
7F8D 32AB7E    01910          LD      (OSFLAG),A      ;YES, END SUBSTITUTION
7F90 23        01920 NOTEND   INC     HL              ;BUMP POINTER TO NEXT
7F91 22AC7E    01930          LD      (OSPTR),HL      ; CHARACTER AND SAVE IT
7F94 E1        01940          POP     HL              ;RESTORE HL
7F95 C9        01950          RET                     ;RETURN CHARACTER IN A
               01960 ;
               01970 ;DEFINITION OF USER STRING
               01980 ;
7F96 3A8038    01990 DEFINE   LD      A,(SHIFT)       ;SHIFT DEPRESSED?
7F99 B7        02000          OR      A               ; (DEFINE ON 'SHIFT-CLEAR')
7F9A 2004      02010          JR      NZ,DEF          ;YES, DEFINE IT
7F9C 3E1F      02020          LD      A,DEFKEY        ;NO, RETURN 'CLEAR'
7F9E B7        02030          OR      A               ;SET FLAGS FOR CALLER
7F9F C9        02040          RET                     ;RETURN CHAR IN A
               02050 ;
7FA0 C5        02060 DEF      PUSH    BC              ;SAVE CALLER'S BC
7FA1 E5        02070          PUSH    HL              ;SAVE CALLER'S HL
7FA2 21DE7F    02080          LD      HL,STRTDF       ;PUT PROMPT FOR START OF
7FA5 CDD57F    02090          CALL    PUTSTR          ; USER STRING DEFINITION
7FA8 216A7E    02100          LD      HL,USTR         ;POINTER TO USER STRING AREA
7FAB 0640      02110          LD      B,USTLEN        ;MAX SIZE OF USER STRING
7FAD E5        02120 GETC     PUSH    HL              ;SAVE OUR HL
7FAE C5        02130          PUSH    BC              ;SAVE OUR BC
7FAF CD        02140          DEFB    CALL            ;BECOMES 'CALL GET-CHAR'
7FB0 0000      02150 KEYDR2   DEFW    0               ;ADDR OF KEY DRIVER HERE
7FB2 C1        02160          POP     BC              ;RESTORE OUR BC
7FB3 E1        02170          POP     HL              ;RESTORE OUR HL
7FB4 B7        02180          OR      A               ;IS A NON-ZERO?
7FB5 28F6      02190          JR      Z,GETC          ;LOOP UNTIL KEY PRESSED
7FB7 FE1F      02200          CP      DEFKEY          ;END DEFINITION?
7FB9 2008      02210          JR      NZ,NTENDF       ;NOT END DEFINITION CHAR
7FBB 3A8038    02220          LD      A,(SHIFT)       ;SHIFT KEY PRESSED?
7FBE B7        02230          OR      A               ;NOT ZERO IF IT IS
7FBF 2009      02240          JR      NZ,ENDDEF       ;YES, END DEFINITION
7FC1 3E1F      02250          LD      A,DEFKEY        ;RESTORE A
7FC3 77        02260 NTENDF   LD      (HL),A          ;ADD CHAR TO USER STRING
7FC4 23        02270          INC     HL              ;BUMP POINTER TO USER STRING
7FC5 CD3A03    02280          CALL    PUTC            ;ECHO KEY PRESSED
7FC8 10E3      02290          DJNZ    GETC            ;REPEAT IF MAX STRING
               02300                                  ; LENGTH NOT YET EXCEEDED
7FCA AF        02310 ENDDEF   XOR     A               ;PUT NULL END-OF-STRING
7FCB 77        02320          LD      (HL),A          ; MARKER AFTER USER STRING
7FCC 21EF7F    02330          LD      HL,ENDMSG       ;PUT OUT 'END OF DEF' MSG
7FCF CDD57F    02340          CALL    PUTSTR          ;ALSO LEAVES 0 IN A
7FD2 E1        02350          POP     HL              ;RESTORE CALLER'S HL
7FD3 C1        02360          POP     BC              ;RESTORE CALLER'S BC
7FD4 C9        02370          RET                     ;AND RETURN TO CALLER
               02380 ;
               02390 ;PUTSTR: PUT STRING AT (HL) ON SCREEN.  STRING IS
               02400 ;         TERMINATED BY A 0 BYTE.
               02410 ;
7FD5 7E        02420 PUTSTR   LD      A,(HL)          ;GET CHARACTER FROM STRING
7FD6 B7        02430          OR      A               ;IF CHARACTER IS NULL (0)
7FD7 C8        02440          RET     Z               ;THEN FINISHED
7FD8 CD3A03    02450          CALL    PUTC            ;ELSE PUT CHAR ON SCREEN
7FDB 23        02460          INC     HL              ;POINT AT NEXT CHARACTER
7FDC 18F7      02470          JR      PUTSTR          ;AND GO GET IT
               02480 ;
               02490 ;PROMPT MESSAGES:
7FDE 0D        02500 STRTDF   DEFB    0DH             ;CARRIAGE RETURN
7FDF 44        02510          DEFM    'DEFINE STRING:'
7FED 0D        02520          DEFB    0DH
7FEE 00        02530          DEFB    0               ;END OF STRTDF
7FEF 0D        02540 ENDMSG   DEFB    0DH             ;CARRIAGE RETURN
7FF0 45        02550          DEFM    'END DEFINITION'
7FFE 0D        02560          DEFB    0DH
7FFF 00        02570          DEFB    0               ;END OF ENDMSG
7E58           02580          END     INIT
00000 TOTAL ERRORS

CALL    00CD 00190    01310 02140
DEF     7FA0 02060    02010
DEFINE  7F96 01990    01430
DEFKEY  001F 00200    01420 02020 02200 02250
ENDDEF  7FCA 02310    02240
ENDMSG  7FEF 02540    02330
GETC    7FAD 02120    02190 02290
INIT    7E58 00290    02580
KEYDR1  7F4E 01320    00300
KEYDR2  7FB0 02150    00310
```

Program continues

```
KEYDRV  4016 00140   00290 00340
KEYWRD  7F6C 01640   01570
LA      7EAE 00700   01640
LB      7EB5 00720
LC      7EBA 00740
LD      7EC0 00760
LE      7EC5 00780
LF      7ECD 00800
LG      7ED1 00820
LH      7ED6 00840
LI      7EDB 00860
LJ      7EE2 00880
LK      7EE7 00900
LL      7EEE 00920
LM      7EF3 00940
LN      7EF8 00960
LO      7EFD 00980
LP      7F02 01000
LQ      7F08 01020
LR      7F0F 01040
LS      7F16 01060
LT      7F1C 01080
LU      7F21 01100
LV      7F27 01120
LW      7F30 01140
LX      7F36 01160
LY      7F3B 01180
LZ      7F40 01200
NOTEND  7F90 01920   01900
NTENDF  7FC3 02260   02210
NXTC    7F75 01700   01730 01750
ONESTR  7F47 01250   00330
OSFLAG  7EAB 00570   01250 01810 01910
OSPTR   7EAC 00600   01790 01870 01930
PUTC    033A 00180   02280 02450
PUTSTR  7FD5 02420   02090 02340 02470
REST    7E6F 00540
SETMD   7F7F 01800
SHIFT   3880 00160   01990 02220
SRCHDN  7F7C 01790   01590 01680
STRTDF  7FDE 02500   02080
SUBMOD  7F62 01550   01450 01490
SUBST   7F85 01860   01270
UDSKEY  001A 00210   01440 01560
USED    0005 00530   00540
USTLEN  0040 00230   00540 02110
USTR    7E6A 00500   00530 01580 02100
```

Level II BASIC, TRSDOS 2.1, 2.2 and NEWDOS all use different keyboard drivers. By picking up the address they have already put in the device control block, it is possible to reap the benefits (debounce, etc.) of these drivers and the benefits of ONESTR at the same time. The address of label ONESTR is installed in the device control block so that it is called in the future for keyboard input.

Choose an Instruction

Choose one of two instructions, depending on whether you will be loading ONESTR from tape or disk. If you will load ONESTR from disk, you must use JP 402DH to return to TRSDOS or NEWDOS. If you will be loading from tape with the SYSTEM command in BASIC, use a JP 1A19H to return to BASIC.

Note that the space taken up by this initialization code can be reused after it is run, since the code is no longer needed. The answers to the MEMORY SIZE? prompt are given in Table 2 and reflect this reuse of space.

At label USTR allocate 64 bytes for the user-defined string and give it an initial value of RUN. Until redefined, this will run a BASIC program by typing SHIFT/down arrow. This may be changed to any other initial value you like, and the code at label REST will allocate the remainder of the 64-byte buffer. A zero byte after the string serves as the string terminator.

Beginning at label LA the table of strings to be substituted for the lowercase letters are listed. Except for LA, the labels are included strictly for convenience in determining which keyword gets substituted for which letter; they are not needed by the code.

This is where you would substitute assembly language mnemonics, Pascal keywords or anything else you would like to type with one key stroke. If you do make substitutions which cause the length of the program to change, be sure to change the program origin so it will fit in your machine! Adjust your answer to the MEMORY SIZE? prompt in this case to protect the new size of the program. Again, a zero byte at the end of each string in the table serves as a terminator.

Selecting Keywords

I chose not to include an entry for PRINT, which is certainly a commonly used BASIC keyword. The Level II BASIC handbook explains that a question mark is a built-in abbreviation for PRINT.

I did choose to include PRINT@ as a keyword, even though typing ?@ works as well and is almost as easy. The reasoning behind this is that @ and SHIFT @ appear the same on the screen, but SHIFT @ doesn't work as a PRINT qualifier. It's a nasty bug to catch since a listing appears normal. Including PRINT@ as a one-stroke entry avoids the problem.

ONESTR is the main entry point to the program, and this is where control is transferred whenever keyboard entry is requested. The OS FLAG is tested to see if the routine is in the middle of a keyword substitution. If so, it branches to SUBST and continues with the substitution. Otherwise, the routine whose address was in the keyboard device control block before ONESTR was loaded is called for keyboard input. If no key has been pressed, it returns to the caller. If a key has been pressed, a decision is made about the next action to take, based on the key's ASCII value (as shown in Table 1).

If the keyboard input is a lowercase letter, the OS flag is set on. Search the keyword table sequentially to find the start of the keyword to substitute.

The number of keywords to skip is calculated from the ASCII value of the lowercase letter read from the keyboard, minus the ASCII value of lowercase a. Since each keyword is terminated by a zero byte, start at the head of the table and check each character. Continue until you pass as many zero bytes as the number of keywords you are supposed to skip.

Don't moan and groan about the inefficiency of a sequential search—it's easy to build an index for the address table and to put it into the routine so that the appropriate address may be computed.

To do this, use a DEFW pseudo-op instruction for each label whose address is included in the following table:

```
ADTABL  DEFW  LA
        DEFW  LB
        ...
        DEFW  LZ
```

The offset into the table for a lowercase letter, say b, is 2 • (ASCII(b)-ASCII(a)). The factor of 2 is used because of the two bytes each entry takes. Load the HL register pair with the contents of this location and you've got the address of the first letter of the keyword.

Why not use this technique, then, instead of the sequential search? Because the address table takes up space. And although it does give faster results, the difference isn't noticeable at the keyboard.

Test the Shift Key

The code at the label DEFINE is executed when the CLEAR key is pressed. Since the keyboard driver routine returns the same value whether CLEAR is pressed alone or shifted, a test must be done to verify that the SHIFT key is indicated. Location 3880H contains a 1 if SHIFT is pressed, and a 0 otherwise. Checking 3880H determines whether to start user string definition, which is triggered by SHIFT/CLEAR.

Assuming SHIFT/CLEAR was pressed, the program types DEFINE STRING: on the screen and waits for input. Each key pressed is tested to see if it is SHIFT/CLEAR, which ends the definition. If not, it is added to the user string buffer at USTR.

If 64 characters are typed without a SHIFT/CLEAR, then definition mode is automatically terminated. User string definition ends by returning a zero to the caller, indicating that no key was pressed. Thus the entire process is invisible to the calls. This process can be used with any program (BASIC, etc.) requesting keyboard input.

Note that most of the Level II BASIC string input editing is not implemented. The back arrow will delete a character, but SHIFT plus the back arrow will not delete the whole line.

Another design trade-off is reflected here. The ROM routines for string input editing could have been used, but they terminate input when BREAK or ENTER are pressed. The approach taken allows entire commands to be typed with one keystroke, including ENTER at the end of the command.

Running ONESTR

If you are loading from disk, run ONESTR from the DOS READY prompt. You can test it at this point by typing shifted letters; keywords should appear.

With either disk or tape, bring BASIC up, answering MEMORY SIZE? as shown in Table 2. Disk users should be in business at this point.

Tape users should enter the SYSTEM command, and load the object tape. Once loaded, run the program by hitting ENTER.

Table 3 is a handy listing of the keys to press for each keyword. Cut it out and tape it to your monitor for quick reference while you are getting used to the system.

Now it's time to find those back issues of *80 Microcomputing* and start enjoying all the programs that you were too lazy to type.■

✓Reader Service—see page 226

More versatile than a line printer.

Teletype Interface

Jake Commander
Shadow Lane, Apt. 2
Peterborough, NH 03458

This article describes how to build and use a simple teletype (TTY) interface to work with a TRS-80 (Level II and up). By interrupting the normal TRS-80 input/output driver software, we can: output line print commands (including LLISTS) to the TTY, have a keyed option to direct screen print commands simultaneously to the video and TTY, input characters to the computer from the TTY.

The following is a 110 baud version, but a few simple adjustments will allow I/O at other speeds.

Cheap Hard Copy

This design provides cheap, high quality hard copy.

In this interface, hardware and software work in conjunction; however, the bulk of the work rests with the software, allowing a greater degree of flexibility.

The hardware consists of a 556 dual timer I.C. and one medium gain NPN transistor, which draw their minimal power from the TRS-80 video socket, pin 1. The circuit is shown in Fig. 1, and a suitable layout on 0.1-inch matrix board in Fig. 2.

To avoid opening up the TRS-80, all signals are carried to the unit via DIN plugs from the video and cassette sockets. These signals are tapped off in the interface and made available again through DIN sockets at the rear. Fig. 3 shows this arrangement.

The only difficulty is that the TRS-80 cassette DIN plug is a non-standard plastic type and will not mate with the standard DIN socket in the interface. The simplest expedient is to remove this plug and replace it with a normal DIN type. The I/O to the TTY is via only three contacts of a 25-way socket. This provides a tidy, compact unit which may be left hooked-up to the computer, even when the TTY is not in use.

Input to the TTY driver section comes from the cassette output port. This part of the circuit receives pulses determined by the software, then stretches them to the required pulse width, which in a 110 baud system is 9.09 ms.

A description of an ASCII character output shows how things happen. Suppose we wish to output to the TTY the letter *J*, which is 0100 1010 in binary. This is odd parity, so to make it even, the software adds a bit, and it now appears as 1100 1010. Before this is output, we need a zero start bit which indicates to the TTY that a byte follows. Now bits 0-7 of the byte are output, followed by two stop bits. The final output is a string of bits as follows:

0	01010011	11
start bit	*J*	stop bits

All the timing, data bits and start/stop bits are manipulated by the software (Program Listing), which receives characters by interrupting the normal flow of I/O. This I/O in the TRS-80 is handled by ROM driver routines which are pointed to by vectors in RAM. By placing our own routine addresses in these vectors we can handle I/O as we like. The vectors are at these locations:

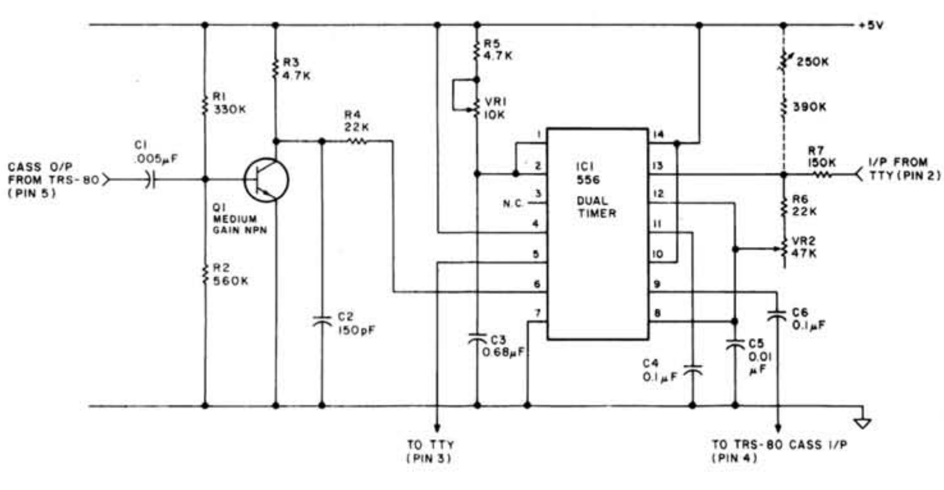


Fig. 1.

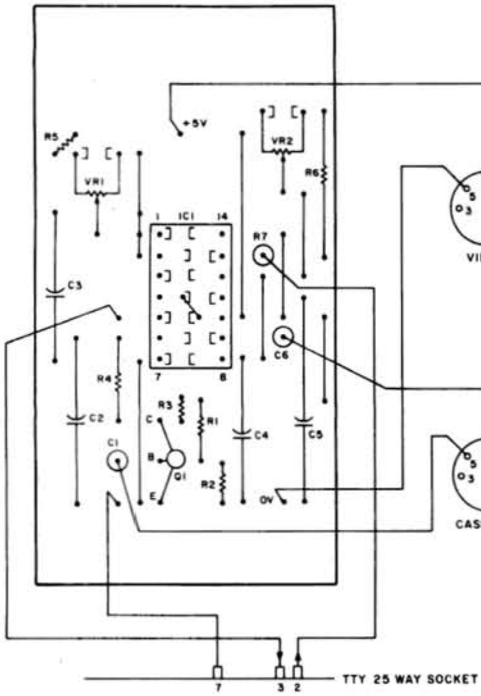


Fig. 2.

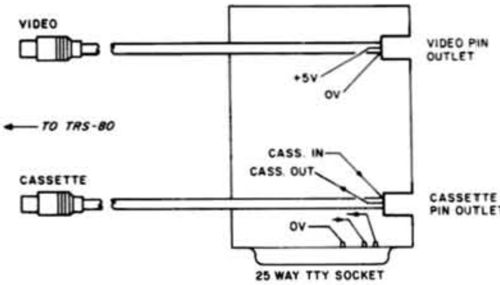


Fig. 3.

16406 + 7 – keyboard driver
16414 + 5 – video driver
16422 + 3 – line print driver

The heart of the TTY output software is the routine called OUT, which is called every time a zero bit is to be output. This issues a positive-going pulse of about 0.1 ms. to transistor Q1, inverting the pulse to trigger the pulse-stretcher half of the 556.

This, in turn, switches high for 9.09 ms. (determined by time-constant R5/VR1/C3) satisfying the requirement of the TTY. To output a one, the routine is merely ignored for 9.09 ms., as the 556 reverts from a zero after its time-constant has expired. The rest of the routine shifts the character and tests each bit to see if a zero or one is required.

Adjusting VR1

When the software is loaded and the TTY hooked up, the simplest way to adjust preset VR1 is to line print an unbroken string of U*. These two characters have bit patterns of 01010101 and 10101010, which are ideal for testing.

VR1 should then be adjusted for a mid position that prints the string with no errors. To increase the baud rate to 300, the time constants at NBIT and WT are altered to 218 and 652, respectively. VR1 is then adjusted as above to give a 3.33 ms. pulse. Other baud rates can be accommodated in the same way.

To perform LPRINTS and LLISTS, the line print vector at 16422 is replaced with the address of the TTY driver routine described above. To get the option of screen print to TTY, the vector at 16414 is then pointed at the routine SPRINT. This routine always prints on the screen first, and, before outputting to the TTY, checks to see if the character is either an up arrow or a shift @. If it is a shift @, the routine disables itself from any further screen-to-TTY output; if it is an up arrow, the output is enabled. Keying or printing these characters allows this option to be switched on or off.

The keyboard input from the TTY is handled by the other half of the 556, which triggers an unstable pulse. When the TTY outputs zero bits, the 556 chops the voltage into pulses that can be detected when input through the cassette port.

If the voltage wasn't chopped, then there would be no way of knowing whether consecutive bits were zeros or ones. The cas-

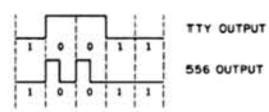


Fig. 4.

Program Listing.

```
                ORG     7E00H
       SETU     LD      HL,SPRINT
                LD      (16414),HL;SCRN PRNT DR ADDR
                LD      HL,LPRINT
                LD      (16422),HL;LINE PRNT DR ADDR
                LD      HL,START
                LD      (16406),HL;KBD DR ADDR
                JP      6CCH     ;TO TRSDOS DO A 'RET'

       SPRINT   CALL    1112     ;PRINT ON SCREEN
                PUSH    AF
                CALL    CHKIT    ;CHECK TTY PRNT
                POP     AF
                RET
       ;***ENTERS WITH CHAR. IN C***
       CHKIT    LD      A,C
                LD      HL,LP
                CP      96       ;SHIFT @ - DISABLE
                JR      NZ,CHK27
                LD      (HL),201         ;RET
       CHK27    CP      91       ;UP-ARROW - ENABLE
                JR      NZ,LP
                LD      (HL),183         ;OR A
       LP       RET              ;INITIALLY DISENGAGED
                LD      HL,CHAR
                CP      (HL)
                LD      (HL),0   ;TURN OFF TTY FLAG
                RET     Z        ;DON'T PRNT IF TTY CHR
                CP      14
                JR      C,LPRINT ;C/R L/F ETC
                CP      16
                RET     C        ;CSRSR ON/OFF
                CP      28
                JR      C,LPRINT ;CRSR MOVES
                CP      32
                JR      C,CRLF   ;SCREEN CNTROLS
       LPRINT   LD      A,C      ;LINE PRINT ENTRY
                CP      10
                JR      C,NOLF
                CP      14
                JR      C,CRLF
       NOLF     LD      A,C
                AND     0C0H     ;TAB?
                CP      0C0H
                LD      A,C
                JR      NZ,TBC   ;NO
                PUSH    BC
                LD      A,3FH    ;MASK
                AND     C        ;= # TAB SPACES
                LD      B,A
                LD      C,20H
       TAB      CALL    LPRINT
                DJNZ    TAB
                POP     BC
```

Program continues

PROGRAM STORE (202) 337-4691

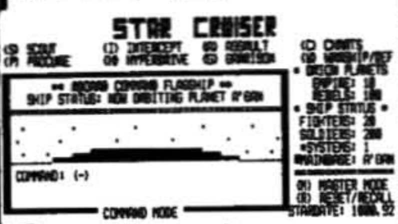
4200 Wisconsin Ave NW P.O.Box 9609 Washington D.C. 20016

All programs for TRS-80 16k, Level II computer.

SUPER NOVA

by Bill Hogue from Big Five

This arcade game of shooting asteroids and alien ships is written in machine language and is the best we have seen on any computer. There are five different types of alien ships including the very deadly Flagship. You shoot from your ship, rotate it, use your thrust key to move, and in emergencies you go into hyperspace. Level 1 or 2 - $14.95



Star Cruiser

from Computer Simulations

You are on the side of the bad guys, the Empire. Program includes fighter combat, ground action, hyperdrive, garrisoning, retaking the 10 system capitals and avoiding the Rebel Star Fighter groups. The Star Cruiser is your last hope. $14.95

Checker King

by Michael Marks from Personal

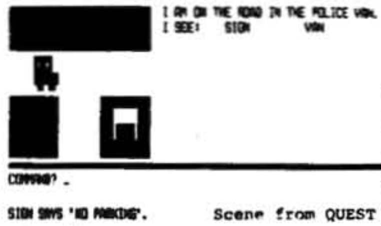
Play checkers against the computer in one of eight levels of skill. Features include saving board positions, switching sides, going back a move and letting it play against itself. Also includes three checker puzzles. Level 1 or 2 - $19.95

Pigskin

by John Laurence, Rick Sothen and Walter Gavenda from Acorn

In this football game you call the plays, watch the thirty-second clock, and get called for penalties if you aren't careful. There are eleven offensive and seven defensive plays. Features graphic display of field, the ball, and statistics on the scoreboard. You can play against the program, against a friend, or watch the play in the spectator mode.

Protected cassette $14.95
Protected diskette $20.95



Scene from QUEST

Adventures

with GRAPHICS & SOUND

by Robert Nicholas from Mad Hatter

These two adventures are similar to most others where you use two word sentences to explore. But these also show you the 'rooms' and have sound effects!

In QUEST you travel in search of fame, fortune, treasures, monsters and more. Three levels of play with a random set up of each game.

As the detective in SLEUTH you try to solve a murder. The victim, weapon and murderer will be different each game.

On cassette $14.95 each.
On disk $24.95 for both.

COMPUTER Bismark

from Strategic Simulations

The historic wargame that accurately simulates the epic battle between the German battleship and the British Home Fleet. Features a computer controlled mapboard of the North Atlantic, hidden movement, ship vrs. ship combat and shadowing, firepower and damage, historic setup of the ships, and both two-player and solitaire scenarios.

Cassette $49.95 Disk $59.95

Coming soon Computer Ambush

MicrocosmI&II

from Basics & Beyond

Combined these two packages contain fifty solid programs.

MICROCOSM I contains 16k-Memory Test, Atlantis, Biorhythms, Calories Food, Calories Ingredients, Challenge Match-em, Checkbook Balancing, Country Guess, Day Of The Week, Divisor Game, Driver, Eliminate, Factorials, Fakeout, Flowering Houseplants, Home Insurance, Instruction Test, Magic Squares, Match-em, Math Tables Drill, Metric Conversion, Morse Code, Mortgage, Music Transposition, News Photographer, Remainder, South Pole, Spelling Drill, Spelling Review, Wipe-out.

MICROCOSM II contains Algebraic Factoring, Base Conversion, Blank Slate, Decipher, Doodle Art 1, Electrical Use, Explore, Fish, Foliage Houseplants, Isolation, Perspective Racer, Planet Finder, Prefix Study and Quiz, Roman Numerals, Shooting Gallery, Spelling Bee, Stains, State Guess, Suffix Study and Quiz, Trivia Quiz.

Each available on two cassettes for $19.95

DISK SCOPE

from Instant

This diskette contains three programs to check out your disks. You can find the tracks and sectors where a file is stored with FILELOCK. With CDISK you can read the ASCII and hex representation of any sector. PASSWORD will give you the password for any file or any disk. $19.95

DISK EDITOR

from Instant

Access to any byte on a diskette is possible with the machine language utility. You just give either the track and sector or the filename in order to alter, add, delete, or view information.

You can also search a diskette for string of characters. And the data can be dumped to a printer. Both the 35 and 40 track versions are on the same diskette. $39.95

SuperSTEP

by Allen Gelder

Single-step/TRACE/Disassembler for TBUG. Variable speed TRACE mode lets you run any Z80 machine language program under total control. Includes intelligent RAM window, foreground background breakpointing, and more. TBUG required. $19.95

FORTRANslator

by Peter Charlton

You can literally translate BASIC programs into Fortran with this program which assumes that you know Disk BASIC and Fortran. It will not do a complete translation because of the differences between the languages. For example, subroutines, 'THEN,' string handling and disk operations are different. But for a programmer this utility is a good start. Requires Microsoft FORTRAN. $24.95 on disk.

FORTRAN available for $95.00

THE PROGRAM STORE
4200 Wisconsin Ave NW
P.O. Box 9609 Dept. K 3
Washington DC 20007

VISA AND MASTERCARD

TO ORDER TOLL FREE 800-424-2738

For program information call (202) 337-4691

Mail orders: Send check or money order for total purchase price, plus $1.00 for shipping & handling. Charge card customers, please include card number, expiration date, and your name as it appears on the card.

```
        RET
TBC     CALL    PRNT
        LD      A,0         ;WILL BE CHR CNT
CRCNT   EQU     $-1
        DEC     A
        LD      (CRCNT),A
        RET     NZ
CRLF    LD      A,13        ;C/R
        CALL    PRNT
        LD      HL,5000 ;TIME FOR
        CALL    WAIT    ;CARRIAGE RETURN
        LD      A,10    ;LINE FEED
        CALL    PRNT
LLEN    LD      A,64    ;CHRS PER LINE
        LD      (CRCNT),A
        RET
PRNT    DI
        OR      A       ;CHK PARITY
        JP      PE,P1   ;NO PRTY BIT NEEDED
        OR      128     ;IF ODD PARITY
P1      LD      D,A
        CALL    OUT     ;0 START BIT
        LD      E,8
NBIT    LD      HL,625  ;110 BAUD DLY
        CALL    WAIT
        LD      A,D
        AND     1       ;0 BIT TO BE O/P?
        JR      NZ,SHIFT;IF NOT
        CALL    OUT     ;O/P 0 BIT
SHIFT   SRL     D
        DEC     E
        JR      NZ,NBIT
        EI
WT      LD      HL,2300 ;TIME FOR STOP BITS
WAIT    DEC     HL      ;*
        LD      A,L     ;* DELAY
        OR      H       ;* LOOP
        JR      NZ,WAIT ;*
        RET             ;*
OUT     LD      A,1
        OUT     (255),A ;PULSE LOW
        LD      A,14    ;PULSE
PLSE    DEC     A       ;WIDTH
        JR      NZ,PLSE ;DELAY
        LD      A,2
        OUT     (255),A ;PULSE HIGH
        RET

START   DI
        IN      A,(255)
        AND     80H     ;TEST, AND..
        JR      NZ,GBIT ;GOT A START BIT
        EI
        JP      995     ;TRY TRS-80 KBD
GBIT    LD      B,8
NXBIT   CALL    INBT    ;GET I/P BIT
        LD      HL,CHAR ;ADDR OF CHR
        SRL     (HL)    ;SHFT RDY FOR
        OR      (HL)    ;NEXT BIT
        LD      (HL),A  ;RE-STORE
        DJNZ    NXBIT   ;IF MORE BITS
        PUSH    AF
        CALL    INBT    ;1ST STOP BIT
        CALL    INBT    ;2ND STOP BIT
        EI
        JR      NZ,STOP ;MUST BE NZ
        POP     AF      ;ASSUME BREAK
        LD      A,1     ;LIKE ROM
        PUSH    AF
        JR      LF
STOP    POP     AF
        INC     A       ;TEST CHR
        JR      Z,START ;IGNORE IF DEL
        DEC     A       ;TEST CHR
        JR      Z,START ;IGNORE IF NUL
        AND     7FH     ;CHOP PARITY BIT
        LD      (CHAR),A ;STORE WITH NO PARITY
        PUSH    AF
        CP      13      ;C/R?
        JR      NZ,FINI
LF      CALL    CRLF
FINI    POP     AF
        RET
INBT    PUSH    HL
        LD      HL,15AH ;WAIT FOR RESET
        CALL    WAIT
        OUT     (255),A ;RESET LATCH
        LD      HL,12DH ;WAIT FOR I/P
        CALL    WAIT
        IN      A,(255) ;GET LATCH STATE
        CPL             ;GET RIGHT WAY RND
        AND     80H     ;ELIM OTHER BITS
        POP     HL
        RET
CHAR    DEFB    0       ;TTY CHR STORE
        END     SETU
```

sette input latch is only set when a pulse is input to it. If a steady voltage caused by a sequence of zeros or ones is input, then it just remains in its last latched state. Fig. 4 shows how this is overcome.

Fig. 4 shows that the cassette input latch will be set/reset during a zero and reset during a one. This is detected by the software and merged into a byte.

If the software does not detect a start byte during its scan, then the TRS-80 keyboard is strobed, allowing either keyboard to be used. The timing of this part of the circuit (VR2) must be set up with care. If the frequency from the 556 is too high, an extra transition is possible, which gives a false zero; if it is too low, then the software is likely to detect a pulse hanging over from the previous one.

Break Detection

A simple break detection may be employed by connecting a 390K resistor in series with a 250K preset between the positive rail and the junction of R6 and R7. This causes the 556 to free-run if the TTY is disconnected, and the frequency can be set to represent a train of zeros, which can then be detected by software.

One useful bonus from using cassette pulses to drive the TTY is an off-line print facility. By recording the driver pulses on cassette and replaying them through the record jack, a printout can be stored on cassette and output from cassette player to TTY as often as required without re-running the original program.

This gives you a unit which is even more versatile than a normal printer and can be used with or without an expansion interface.■

Parts List

C1 .005 μF	R1 330K
C2 150 pF	R2 560K
C3 .68 μF	R3, R5 4K7
C4, C6 .1 μF	R4, R6 22K
C5 .01 μF	R7 150K
VR1 10K	Q1 medium gain NPN
VR2 47K	IC1 556 dual timer

A hardworking utility that saves time and space.

Document Those Variables

William Noel
15 Kingswood Ct.
Columbus, GA 31907

You've just received someone's advanced Star Trek game and want to soup it up a little or maybe update it to the movie version. Now comes the tedious part. You must sift through the program trying to figure out how and what variable names are used.

One long night I sat peering at the screen with bloodshot eyes trying to shift @ the program list, while trying to remember what had disappeared ten lines ago. Either inspiration or desperation hit me: Why not get my TRS-80 to help with this lengthy effort?

Program Text Area

First, I had to figure how the information was coded in the BASIC text area. Program Listing 1 helps examine this area of memory. It begins at location 17129, where the memory contents are PEEKed and printed. If the memory byte contains a normal screen character, this symbol is also shown.

After much trial and error in analyzing memory displays I have discovered that the first two positions of each line point to the memory location beginning the next line. The BASIC interpreter can use this, for example, to avoid the delay of trying to decode remark lines during a program run.

The third and fourth locations represent the line number itself. Table 1 shows this coding for the first two lines of our memory display program.

```
239          66   B        1             0
132          0             250           66   B
5            0             73   I        32
213          73   I        205           49   1
0            14            67   C        10
0            72   H        32            213
32           229           40   (        49   1
55   7       49   1        50   2        57   9
205          71   G        71   G        41   )
0            26            67   C        20
0            71   G        71   G        213
71   G       71   G        205           49   1
0            40   (        67   C        30
0            178           32            72   H
59   ;       34   "        32            32
?
34   "       59   ;        0             76   L
67   C       40   (        0             143
32           72   H        212           51   3
49   1       32            210           32
72   H       214           57   9        49   1
32           202           32            178
32           247           40   (        72   H
41   )       44   ,        32            58   :
149          32            178           34   "
34   "       44   ,        0             96
67   C       50   2        0             143
32           73   I        213           54   6
48   0       32            137           32
81   Q       58   :        32            73   I
213          48   0        0             104
?
67   C       60   <        0             141
32           53   5        0             0
0            4             0             73   I
```

Fig. 1. Out of Memory Display.

```
65000 A$ L M N A$ C S
65005 D T D T D E T E T E J T S F E E
65010 D U Z S S U U
65012 D U U U Z S S T
65015 F Z Z
65020 F Z U D
65030 W Z W
65040 W Z W
65050 W
65060 Z J
65080 Z J
65090 J
65100 Z Z
65110 Y$ Z S S S S S
65120 Y$ Y$ S T Y$ Y$ T
65140 Y$ F A
65150 A$ A Y$
65160 A$ A Y$
65170 B C A
65180 A$ B A$ B L B L B M B M B N B N B B
65190 C C C
65200 A$ A Y$ L A E M A E N A
65210 M A E M A E N A N A
65220 A
65230 A
65240 A C
65250 A$ A L A M A N A K K K R K
65260 A
65270
65280 T U U U
```

Fig. 2. Variable Hunt.

```
1 CLS
5 I =I+1
10 H = PEEK(17129+GG)
20 GG=GG+1
30 PRINT H;"  ";
40 IF H>31 AND H<91 THEN PRINT CHR$(H), ELSE PRINT"",
50 IF I=60 INPUT Q: I=0
60 GOTO 5
```

Listing 1.

Listing 2.

```
65000 'RESET BEGINNING OF TEXT & DEFINE TABLE'
        POKE 16548,233:POKE 16549,66:CLEAR 200:CLS:DIM A$(
           100), L(100), M(100), N(100):A$(1)="[":C=1:S=1

65005 'GET NEXT LINE, THIS LINE, PRINT, AND CHECK IF THROUGH'
        GOSUB 65280:D=T:GOSUB 65280:D=T*256+D:GOSUB 65280:
         E=T:GOSUB 65280:E=T*256+E:PRINT"":PRINT E;:J=0:
         T=1:S=1:F=0:IF E=65000 GOTO 65240

65010 'FINISH LAST CHARACTER OF TEXT LINE'
        IF D=17129+U THEN Z=S:S=1:U=U+1:GOTO 65015

65012 'CHECK IF LINE HAS ENDED; OTHERWISE GET NEXT LOCATION'
        IF D=17128+U THEN U=U-1:GOTO 65005:ELSE Z=S:S=T:
         GOSUB65280

65015 'IF LONG VARIABLE NAME, SKIP REST'
        IF F=1 AND Z>64 AND Z<91 GOTO 65010

65020 'WHEN A REMARK COMMAND, JUMP TO NEXT LINE'
        F=0:IF Z=147 THEN U=D-17129:GOTO 65005

65030 'BEGIN SKIP OF CHARACTERS FOLLOWING A " '
        IF W=0 AND Z=34 THEN W=1:GOTO 65010

65040 'END SKIP WHEN THE 2ND " IS REACHED'
        IF W=1 AND Z=34 THEN W=0:GOTO 65010

65050 'SKIP CHARACTERS BETWEEN'
        IF W=1 GOTO 65010

65060 'WHEN : IS REACHED, RESET THE DATA COMMAND SKIP'
        IF Z=58 THEN J=0

65080 'SET TO SKIP IF A DATA COMMAND'
        IF Z=136 THEN J=1

65090 'DATA COMMAND SKIP'
        IF J=1 GOTO 65010

65100 'ALLOW ONLY LETTERS IN 1ST CHARACTER'
        IF Z<65 OR Z>90 GOTO 65010

65110 'BUILD 1ST LETTER, CHECK IF VALID 2ND CHARACTER'
        Y$=CHR$(Z):IF(S<48 OR S>57) AND (S<65 OR S>90) AND
         S<>36 GOTO 65140

65120 'ADD 2ND CHARACTER, ADD 3RD CHARACTER IF $ '
        Y$=Y$+CHR$(S):IF T=36 THEN Y$=Y$+CHR$(T)

65140 'PRINT VARIABLE AND BEGIN TABLE SEARCH'
        PRINT Y$;" ";:F=1:FOR A=1 TO 100

65150 'TRY AGAIN IF VARIABLE NOT YET REACHED'
        IF A$(A)<Y$ GOTO 65230

65160 'FIND THE MATCHING VARIABLE'
        IF A$(A)=Y$ GOTO 65210

65170 'BEGIN SHIFT TO MAKE ROOM FOR NEW VARIABLE'
        FOR B=C TO A STEP-1

65180 'SHIFT OVER 1 WORKING RIGHT TO LEFT'
        A$(B+1)=A$(B): L(B+1)=L(B): M(B+1)=M(B): N(B+1)=N(B)
         :NEXT B

65190 'INCREASE TABLE ENTRY COUNT AND CHECK IF OUT OF ROOM'
        C=C+1:IF C=100 STOP

65200 'INSERT NEW VARIABLE'
         A$(A)=Y$: L(A)=E: M(A)=E: N(A)=1: GOTO 65220

65210 'WHEN THE SAME VARIABLE, UPDATE FOR THIS LINE'
        IF M(A)<>E THEN M(A)=E: N(A)=N(A)+1

65220 'CAUSE AN EARLY END TO THE TABLE SEARCH'
        A=100

65230 'CONTINUE TABLE SEARCH OR GET NEXT TEXT POSITION'
        NEXT A:GOTO 65010

65240 'BEGIN DISPLAY OF VARIABLE SUMMARY'
        PRINT"":FOR A=1 TO C-1

65250 'PRINT SUMMARY AND WAIT IF A FULL SCREEN'
        PRINT A$(A),L(A),M(A),N(A),: K=K+1:IF K=15 THEN
         INPUT R:CLS:K=0

65260 'PRINT THE NEXT LINE'
        NEXT A

65270 END

65280 'GET THE CURRENT POSITION OF TEXT & MOVE COUNTER TO NEXT'
        T=PEEK(17129+U): U=U+1: RETURN
```

Note: Up arrow appears as left bracket in listings.

Note: Line 65005 allows the program to run independently.

As you can see, the line number code has the least significant value first. This seemingly backwards way of coding is the manner the TRS-80's microprocessor handles two-byte numbers.

Look a little closer at the next line pointer which also has this reverse sequence.

A byte, of course, has eight bits that can be turned either off or on representing either zero or some numeric value. A code of two bytes gives a total of 16 bits to represent a memory address in the next line pointer's case.

Each of these 16 bits are numbered 0-15 and represent, when on, a value based upon their location number. Beginning with bit 0 having a value of $1(2^0 = 1)$, two is multiplied times the prior bit value. These values are shown in Table 2.

Looking at byte 1, you can create total values from 0 to 255 by adding different bit combinations. This matches the range of values generated by PEEKs into memory. After looking at both bytes together in this manner, it is easy to see why the first byte should represent the low order value.

Adding all 16 of these bit values together results roughly in the largest memory value and line number that can be represented in two bytes. This figure totals to 65535.

When I say it is a rough maximum, I am referring to the line number. For some reason that I have yet to find, we are restricted to an upper line number limit of 65529, although the coding possibilities will allow six more. But who's going to need them with over 65000 available?

Of course, in BASIC we must deal with one byte at a time while using the PEEK statement. It is possible to convert the second byte to its proper value by multiplying it by 256.

Convert line one's next line pointer:

$$239 + (66 \cdot 256) = 17135$$

This result matches the first memory byte of line five.

Text Coding

Fig. 1 shows the complete text area for the memory display program. The variables I, H, GG, and Q are kept in normal screen and keyboard coding.

But the statements and functions CLS, PEEK, PRINT, IF, AND, ELSE, INPUT, GOTO, =, +, >, <, CHR$ have all been converted to a one-character special code.

This conversion in addition to saving memory also makes it easier to identify program variable names. PEEK won't be confused with the variable name PE. The same is true of other statements and variables.

Listing 2 shows a program that picks out variables. At the same time the program saves the first line number, last line number, and number of total line occurrences for each variable.

When a program line contains a remark, everything following it will be ignored by Level II machines. When you enter these program lines, either leave out the remarks or enter them at different line numbers.

Figs. 2 and 3 show the screen results of this program analyzing itself. Fig. 2 shows the program as it progresses through the text identifying each line and the variables found in that line. Fig. 3 shows the screen summary displayed when that run is finished.

Table 3 defines each variable used in the variable documentation program.

Now that we have a program that documents variables, another bothersome detail crops up. This program must be in the same text area as the program it will analyze.

The July, 1979, issue of Radio Shack's *Microcomputer Newsletter*, includes a method for adding a BASIC program by keying it in at the end of existing text contents. The flowchart in Fig. 4 shows the steps involved.

After trying it a few times, I looked for an easier way. What was this PEEKing and POKEing really accomplishing?

With our knowledge of the coding technique for current line number and next line location, let's convert the last two

A	65140	65260	11
A$	65000	65250	6
B	65170	65180	2
C	65000	65240	4
D	65005	65020	4
E	65005	65210	3
F	65005	65140	4
J	65005	65090	4
K	65250	65250	1
L	65000	65250	4
M	65000	65250	5
N	65000	65250	5
R	65250	65250	1
S	65000	65120	6
T	65005	65280	4
?			
U	65010	65280	4
W	65030	65050	3
Y$	65110	65200	6
Z	65010	65110	10

Fig. 3. Variable Summary.

Line Number		Next Line Pointer	Current Line Number	Line Program Statements	Trailing Character
1	Memory				
	Location:	17129 30	31 32	33	34
	Value:	239 66	1 0	132	0
	Character:	B			
5	Memory				
	Location:	17135 36	37 38	39 40 41 42 43 44	45
	Value:	250 66	5 0	73 32 213 73 205 49	0
	Character:	B		I Space I 1	

Table 1. First two lines of Memory Display Program.

Byte									
1:	Bit No.	0	1	2	3	4	5	6	7
	Value	1	2	4	8	16	32	64	128
2:	Bit No.	8	9	10	11	12	13	14	15
	Value	256	512	1,024	2,048	4,096	8,192	16,384	32,768

Table 2. Two Byte Numbers.

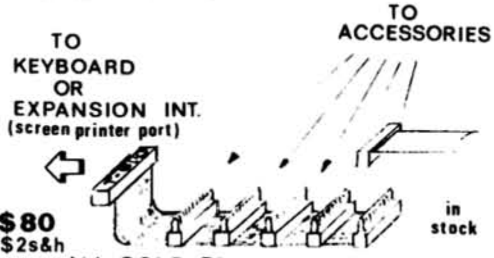
A	Counter for table search
A$	Table for variable name
B	Present location during table shift
C	Pointer for last table entry
D	Next line beginning memory address
E	Current line number
F	Switch to skip rest of long variable name
J	Switch to skip after DATA statement
K	Line count for screen summary
L	Table for first line occurrence
M	Table for last line occurrence
N	Table for number of total line occurrences
R	Fake input variable
S	Next to oldest text character (1 ago)
T	Current text character
U	Counter for current position in text
W	Switch to skip from first "to 2nd"
Y$	Current variable name
Z	Oldest text character (2 ago)

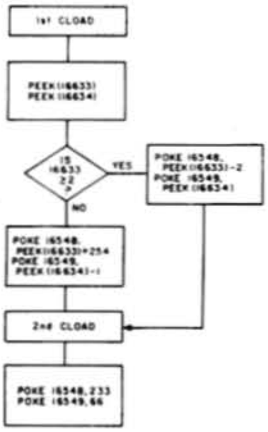
Table 3. Variable List.

Line number	Change
65000	delete the two POKEs
65010	change 17129 to 26302
65012	change 17128 to 26301
65020	change 17129 to 26302
65280	change 17129 to 26302

Then type SAVE "VARDOC,"A

Table 4. Program Changes for Disk BASIC.





Flow Chart.

POKEs to

233 + (66•256) = 17129

Voila! That is the beginning text memory location we have been using. Therefore, locations 16548 and 16549 must be where Level II BASIC keeps this pointer.

The comparison of location 16633 to 2 and the two POKEs on either route afterwards, are going to be a little more difficult to decipher.

If you have the two-byte number 1 67 and want to subtract 2 from it, you run into trouble right away. One minus 2 gives −1, which isn't within the 0-255 code range. So borrow 1 from the second byte making it 66 and add its multiplied value 1•256 to the first byte. This manipulation gives you the number 257 66. Now subtract 2, giving 255 66.

If the first byte were 2 or larger, you wouldn't have to borrow from the second. Two minus 2 gives 0 which is a valid result.

Suppose, somewhere in memory, Level II BASIC keeps track of the memory location that follows the last program line. It could then point two bytes beyond the final program line.

Double *VOILA!!* The beginning of text pointer (locations 16548 and 9) is replaced with the position new text should begin (locations 16633 and 4 minus 2). When this is done, Level II BASIC is tricked into loading a new program after the current text program. Finally, the beginning text pointer must be reset back to the correct value of 17129.

Assembler Help

Program Listing 4 shows an assembler program which replaces the beginning text pointer. The decision-element (DE) register pair is loaded with the value 2 while the HL register pair is loaded with the hexadecimal address of the end of text pointer.

The DE pair is subtracted from this address, and any borrowing is automatic. It then stores this adjusted value in the hexadecimal address of the beginning text pointer and jumps to the Level II READY message.

This program loads just below Radio Shack's debounce routine on a 16K system.

The beginning text pointer is then reset to its correct value by the two POKEs in line 65000 as shown in Program Listing 2.

The steps to run variable documentation are:

1. CLOAD the program to be analyzed.

2. Enter SYSTEM, ADDTXT and / (or follow the steps in Fig. 4).

3. CLOAD the variable documentation program in Program Listing 2.

4. Enter RUN 65000.

If the screen becomes completely full during the summary display, hit ENTER to continue.

To run variable documentation with Disk BASIC, make the program changes shown in Table 4.

The execution steps are 1) LOAD the program to be analyzed; 2) enter MERGE "VARDOC"; and 3) enter RUN 65000.

There are several things that could cause complications in running the variable documentation program. Remember that the program to be analyzed should not have lines numbered 65000 or greater.

Program tables allow only 99 entries. They must be increased to record more variable names. Also the CLEAR must be increased correspondingly. These changes will naturally use more memory.

If there is not enough memory to do the run, delete the following from the program to be analyzed:

- REM lines
- Lines that don't have variables
- DATA statements
- Characters between double quotes

If you want a printout of the summary, insert the following program line:

```
65255 LPRINT A$(A), L(A), M(A), N(A)
```

Saving Memory

Understanding how program lines are coded in the text area makes it easier to see what to avoid for programs tight on memory. Remember that spaces in the text line use one byte each; each program line adds five bytes (VS one byte for a :); and REM lines use a full byte for each comment character. It is better to choose single-letter variable names than two character names.

Now that you have a program and an understanding of how to document variables, how about a challenge? Who out there can develop a program to effectively document GOTOs, GOSUBs, and RETURNs?■

```
65000 POKE16548,233:POKE16549,66:CLEAR200:CLS:DIMA$(100
      ),L(100),M(100),N(100):A$(1)="[":C=1:S=1
65005 GOSUB65280:D=T:GOSUB65280:D=T*256+D:GOSUB65280:E=
      T:GOSUB65280:E=T*256+E:LPRINT"":J=0:T=1:S=1:F=0:IF
      E=65300GOTO65240ELSELPRINTE;
65010 IFD=17129+UTHENZ=S:S=1:U=U+1:GOTO65015
65012 IFD=17128+UTHENU=U-1:GOTO65005:ELSEZ=S:S=T:GOSUB6
      5280
65015 IFF=1ANDZ>64ANDZ<91GOTO65010
65020 F=0:IFZ=147THENU=D-17129:GOTO65005
65030 IFW=0ANDZ=34THENW=1:GOTO65010
65040 IFW=1ANDZ=34THENW=0:GOTO65010
65050 IFW=1GOTO65010
65060 IFZ=58THENJ=0
65080 IFZ=136THENJ=1
65090 IFJ=1GOTO65010
65100 IFZ<65ORZ>90GOTO65010
65110 Y$=CHR$(Z):IF(S<48ORS>57)AND(S<65ORS>90)ANDS<>36G
      OTO65140
65120 Y$=Y$+CHR$(S):IFT=36THENY$=Y$+CHR$(T)
65140 LPRINTY$;" ";:F=1:FORA=1TO100
65150 IFA$(A)<Y$GOTO65230
65160 IFA$(A)=Y$GOTO65210
65170 FORB=CTOASTEP-1
65180 A$(B+1)=A$(B):L(B+1)=L(B):M(B+1)=M(B):N(B+1)=N(B)
      :NEXTB
65190 C=C+1:IFC=100STOP
65200 A$(A)=Y$:L(A)=E:M(A)=E:N(A)=1:GOTO65220
65210 IFM(A)<>ETHENM(A)=E:N(A)=N(A)+1
65220 A=100
65230 NEXTA:GOTO65010
65240 STOP:LPRINT"":FORA=1TOC-1
65250 LPRINTA$(A),L(A),M(A),N(A):K=K+1:IFK=15THENINPUTR
      :LPRINT"?":LPRINT" ":LPRINT" ":LPRINT" ":LPRINT" "
      :CLS:K=0
65260 NEXTA
65270 END
65280 T=PEEK(17129+U):U=U+1:RETURN
65300 REM
```

Listing 3.

```
7FBA          00100         ORG    7FBAH        ;BEGIN AT 32698
7FBA 110200   00105 BEGIN   LD     DE,2         ;LOAD IN 2
7FBD 2AF940   00110         LD     HL,(40F9H)   ;LOAD CONTENTS OF 16633 & 4
7FC0 ED52     00120         SBC    HL,DE        ;SUBTRACT 2
7FC2 22A440   00130         LD     (40A4H),HL   ;LOAD RESULT IN 16548 & 9
7FC5 C3191A   00140         JP     1A19H        ;JUMP TO READY MESSAGE
7FBA          00150         END    BEGIN
00000 TOTAL ERRORS
```

Listing 4.

Get perfectly aligned printouts every time.

Printer Calibration

L. O. Rexrode
35026-A Turtle Trail
Willoughby, OH 44094

If you use more than one type of paper with your printer, this simple modification makes a lot of sense.

At the office my TRS-80 and Centronics 779 with tractor feed are used primarily for handling advertising mail inquiries and our normal operation uses five different types of forms.

In the interest of saving time and improving efficiency, the need for a method that starts every print routine exactly where it is intended, regardless of who is operating the sytem, was apparent.

Three Variables

With the Centronics 779, three variables are of primary importance in printing a professional looking form: left margin setting, top-of-form setting and the "density" setting. Any, or all of these, may require different settings for various print routines or form sizes. The length of form (number of lines) is not included, since a program is usually written with this factor as a constant.

The simplest variable is the left margin setting. The printer's roller/tear-bar assembly has an engraved scale in one-eighth-inch increments from −15 on the left to 90 on the right. This scale should be used to set the left edge of the form to be printed. The value is determined by a trial run.

To set the top-of-form requires the addition of a scale to the 779. This is done quite easily if you use the hold down clamp on the left pin feed assembly. As shown in Photo 1, use a three and one-half-inch peel-off label and type a column of numbers, say, from 1 to 21. Trim off the excess margins and place the label right along the edge of the hold down clamp. The resultant scale now allows you to set the top-of-form to a specified position.

The third variable is the print density control. This sets the number of characters-per-inch printed. For maximum legibility you usually want this set as wide as the form will allow. The control, located on the rear of the printer, is not only difficult to reach but requires several lines of print, with trial and error adjustments, to get the optimum settings. A calibrated dial knob solves this problem easily.

The dial knob I used is Radio Shack's part no. 274-413. However, on some printers, the shaft may not protrude quite far enough to get a good bite when the set screw is tightened. If so, use a coarse file on the rear face of the knob, removing about 1/32 of an inch to 1/16 of an inch.

Cut a sliver from the adhesive-backed label to make a pointer for the dial and place it approximately as shown (Photo 2). Rotate the adjustment shaft fully counterclockwise (lowest density setting) and push the knob onto the shaft, setting the number one mark to the arrow. Tighten the set screw and you're through.

Find Your Values

Run each of your print routines and determine the optimum value of the three variables: left margin, top-of-form and density. Note these values for use with the routines. The best place to note them is in your program, immediately preceding the first LPRINT statement. If you are new to programming, here is the statement I use to get the results shown in Photo 3 (your line number, title and values will relate to your own program, of course):

```
1000 CLS:PRINT@320,"READY TO PRINT
     MAILING LABELS DIRECTLY FROM
     DISC":PRINT:PRINT"SET TOP-OF-
     FORM TO 20":PRINT"SET LEFT
     EDGE OF FORM TO  - 4":PRINT
     "SET DENSITY CONTROL TO 4":
     INPUT" 'ENTER' TO CONTINUE";X
```

When LLISTing a program in BASIC, it is not possible to use such a prompting message, and I am constantly LLISTing with the density set too low, which results in my losing the ends of long program statements. As a reminder to myself to set the density before starting a LLIST, I

Photo 1

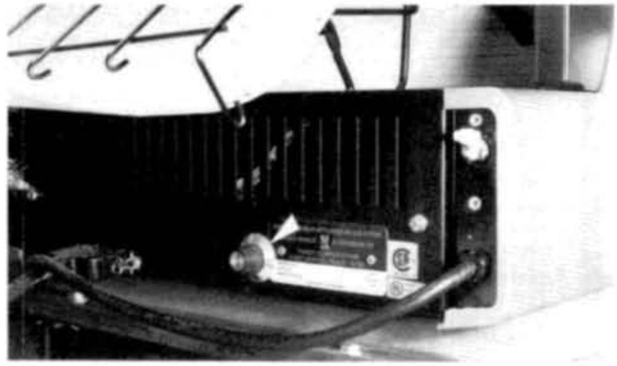


Photo 2

Photo 3

used another label, placed prominently on the top front of the printer that almost shouts, "For Listings Set the Density to 7 inches. It works.

We have been using this method for several months now, and it is quite gratifying to see the results. No matter who is operating the system, and regardless of how often we change forms or paper, the first line of print goes down exactly where we want it to.

A couple of other time savers we have added to our program are worth the time and the slight additional memory required. The first one resulted from our operator having spent over an hour trying to figure out why the computer wouldn't run and finally realizing that the "out of paper switch was off, due to a small tear in the form which wasn't readily apparent.

To prevent this from happening again, I added a GOSUB in front of every LPRINT statement. The subroutine checks the status of both the "out of paper switch and the print switch. If either is off, a prompting message is displayed telling the operator what is wrong and allowing him to correct before continuing. My subroutine reads as follows:

```
750 IFPEEK(14312)<128 RETURN ELSE 751
751 CLS:PRINT@468,"PRINTER NOT
    READY": PRINT@ 583,"'R'—RETURN
    TO MENU 'O'—OK TO CONTINUE"
752 Q$ = INKEY$ :IF Q$ = "R"THEN 511
753 IFQ$ = "O"THEN RETURN
```

This simple solution has saved us hours.

Add Top-of-form

Since the 779 doesn't have front panel controls that allow either a line feed or a top-of-form, I added these as a part of my program. Since each program has a menu, this was the obvious place to access these routines.

The menu uses the INKEY$ function, so it was just a matter of adding the up arrow as the selection to advance one line and adding the letter "T" to go to top-of-form. I did not include these two symbols in the menu table to prevent clutter. They are blind selections. Here are the routines:

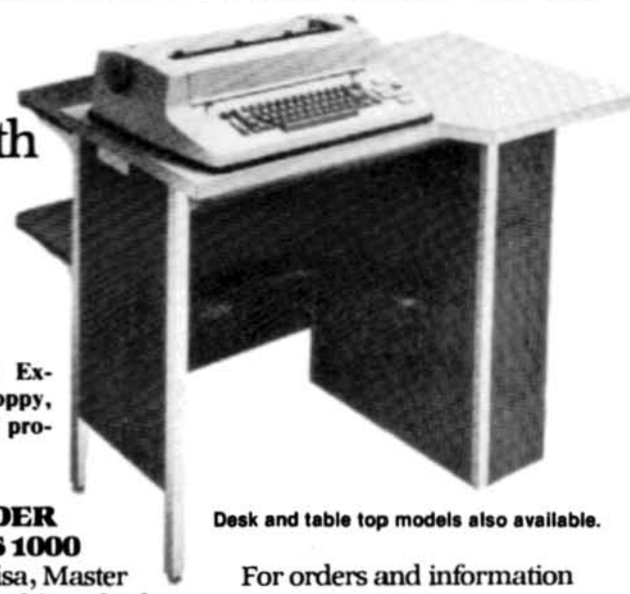
```
620 IFQ$ = "↑"THEN 800
625 IFQ$ = "T"THEN 810

800 IF PEEK(14312)<128 GOTO 805 ELSE
    511
805 POKE 16424,1;POKE 16425,0:LPRINT
    CHR$(11): GOTO 511
810 IF PEEK(14312)<128 GOTO815 ELSE
    511
815 POKE 16424,51:POKE 16425,0:LPRINT
    CHR$(11):GOTO511
```

In either of these routines, if the printer is not ready, nothing happens except returning you to the menu. The ↑ advances the paper one line. The T advances the paper 51 lines, then you are returned to the menu.

In all the examples given, my menu routine begins at line 511. ■

The DATA-TRANS 1000



A completely refurbished **IBM** Selectric Terminal with built-in **ASCII** Interface.

***FOR YOUR TRS-80 WITH OR WITHOUT EXPANSION INTERFACE.**

Features:

- 300 Baud
- 14.9 characters per second printout
- Reliable heavy duty Selectric mechanism
- RS-232C Interface
- Documentation included
- 60 day warranty - parts and labor
- High quality Selectric printing Off-line use as typewriter
- Optional tractor feed available
- 15 inch carriage width

Also works with Exatron's Stringy Floppy, for fast loading of programs.

Desk and table top models also available.

HOW TO ORDER DATA-TRANS 1000

1. We accept Visa, Master Charge. Make cashiers checks or personal check payable to:

DATA-TRANS

2. All orders are shipped F.O.B. San Jose, CA
3. Deliveries are immediate

For orders and information

DATA-TRANS
2154 O'Toole St. ✓274
Unit E
San Jose, CA 95131
Phone: . (408) 448-0800

SMALL SOFTWARE SYSTEM **PRODUCTS FOR THE TRS-80*** SMALL SOFTWARE SYSTEM

NEW!

PENMOD - $19.95. Adapts Disk-Pencil to Radio Shack lower case modification. Also adds single page printing and several other new features.

SCRIPMOD - $14.95. Add TRS232 print driver, or add handshake/linefeed control to RS-232-C driver in Radio Shack's SCRIPSIT (disk version only).

WHISTLER: HOME CONTROLLER INTERFACE - $34.95. New hardware product that controls lights, appliances, computer peripherals, darkroom timers and other 115 volt devices anywhere in your house! Software controlled by cassette cable. Use with Sears or BSR Home Control System with ultrasonic option. Assembled, tested, self-contained, and includes Basic software.

UTILITIES

RSM-2: MACHINE LANGUAGE MONITOR FOR 16K TRS-80'S - $26.95
RSM-2D: THREE VERSIONS OF RSM-2 FOR DISK SYSTEMS - 29.95
RSM-2 RELOCATOR: PUT RSM-2/2D ANYWHERE IN MEMORY - 9.95

Machine Language monitors with Z-80 disassembler! HEX and ASCII memory dumps; EDIT, MOVE, EXCHANGE, VERIFY, FILL, ZERO, TEST, or SEARCH memory, read/write SYSTEM tapes, enter BREAKPOINTS, PRINT with TRS232 or Centronics, read/write disk sectors directly! RSM-2 tape loads at top of 16K LEVEL I or II; RSM-2D disk includes 3 versions for 16K, 32K and 48K.

DCV-1: CONVERT SYSTEM PROGRAMS TO DISK FILES -$9.95. Execute Adventure, Air Raid, RSL-1, ESP-1, T-BUG, etc. from disk, even if they interfere with TRSDOS! New version works with TRSDOS 2.3.

BASIC-1P: LEVEL-1 BASIC WITH PRINTING! - $19.95. Run any LEVEL-I BASIC program on your 16K Level-2. PLUS LPRINT and LLIST with our TRS232 or Centronics. Furnished on tape; can be used from disk.

MACHINE LANGUAGE GAMES

AIR RAID, BARRICADE or **RSL-1: - $10.00** each, all 3 for **$25.00**

AIR RAID: A super shooting gallery; our most popular game. Ground based missile launcher shoots high speed aircraft! Hours of fun!

BARRICADE: "BREAKOUT" for the TRS-80! Break through 5 walls with high-speed ball and keyboard controlled paddle! 96 different options!

RSL-1: Enter patterns with repeating keyboard! Save patterns on tape (4 furnished). Play John Conway's LIFE. FAST - about 1 second per generation!

SMALL SYSTEM SOFTWARE P.O. BOX 366 NEWBURY PARK, CA 91320

PROFESSIONAL SOFTWARE

NEW! ELECTRIC PENCIL-IIB FOR MODEL-II. Super Pencil version runs under TRSDOS or CP/M. Automatic centering, dynamic print formatting, single-page printing,etc. Buffered keyboard eliminates missed characters at line ends! Diablo, NEC, Qume versions include bold face print, variable pitch, & more!
TRSDOS PENCIL: Standard printer - $325; Diablo, NEC, Qume (specify) - $350
CP/M PENCIL: Standard printer - $275; Diablo, NEC, Qume (specify) - $300

ELECTRIC PENCIL FOR MODEL-I: TAPE-$99.95, DISK-$150.00. Popular video editor for creating and saving text files. Prints formatted copy with right justification, page titling & numbering, etc. Upper case only, or lower case with modification. Requires at least 16K.

RSMII: ENHANCED RSM MONITOR FOR THE MODEL-II - $39.95. Relocatable version of RSM-2D plus screen editor for modifying either memory or disk sectors in both Hex and ASCII, split screen scrolling, and formatted serial or parallel printing. Sold on self-booting disk; directions to save as TRSDOS file.

CP/M OPERATING SYSTEM: MODEL-I - $145.00; MODEL-II - $170.00. The 8080/Z80 "Software Bus" for TRS-80's. Model-I includes TRS232 and RS-232-C software. Model-II supports single and double density disks, and reads TRSDOS files. Many unique utilities included in both versions!

PRINTER SUPPORT

TRS232 PRINTER INTERFACE - $59.95 Assembled & tested printer interface for RS232 or 20-mil current loop printers. Expansion interface not required. Print from level-II BASIC, CP/M, BASIC-1P, ELECTRIC PENCIL, etc. Standard cassette software included. Add $2.00 for shipping.

TRS232 "FORMATTER" SOFTWARE PACKAGE - $14.95. Adds page and line length control, printer pause, "smart" line termination, etc. to TRS232.

RSM232: Adds RS-232-C capability to RSM-2/2D monitors - $9.95
PEN232: RS-232-C for cassette version Electric Pencil - 9.95
EDT232: TRS232 and RS-232-C for tape version of EDTASM - 9.95

OTHER PRODUCTS FOR THE TRS-80

ESP-1: $29.95. Assembler, Editor, Monitor (8080 mnemonics)
LST-1: 8.00. Listing of Level-1 BASIC with some comments

CP/M tm Digital Research, Inc. TRS-80 tm Tandy Corp.
See your dealer or order direct. Calif. Residents add 6% tax ✓30

SMALL SYSTEM SOFTWARE P.O. BOX 366 NEWBURY PARK, CA 91320

Puts sophisticated business strategies at the fingertips of all TRS-80 Level II owners! **FINPLAN: A Financial Planning Program for Small Businesses** *is the most complete financial planning program available.*

Here's why . . . **Developed by an expert:** The author, Robert Montgomery, is president of a management consulting firm and a recognized authority on new product planning. **Easy to use:** Data can be entered from normal balance sheet statements. Within seconds, **FINPLAN** will provide financial projections for up to 5 years. **Versatility:** Are you thinking about modernizing equipment, expanding into a new product line, or starting a new business from scratch? Will your plans run you out of business, out of cash, or on to the Fortune 500 list? **FINPLAN** will answer these and other financial questions before you invest a dime. **Extensive documentation:** *You don't have to be a financial wizard to use* **FINPLAN** because it comes with 80 pages of documentation. Presented in a step-by-step format, the documentation is a great learning tool for those who need it or a dynamic applications tool for getting the most out of **FINPLAN.** And, **FINPLAN** is reasonably priced. #05103, TRS-80 Level II cassette, $69.95; #05108, TRS-80 Level II Disk, $74.95.

Available at your local computer store!

or **Call Toll Free,** 24 hours a day, **(1-800-827-3777, ext. 302)*** TO CHARGE YOUR ORDER TO Master-Charge or Visa! Minimum order is $10.00; Customer pays postage and handling.

*From Missouri, call 1-800-892-7655, ext. 302

Hayden Book Company, Inc.
50 Essex Street
Rochelle Park NJ 07662 ✓419

Useful software for easy data input.

Versatile Input

Tim Wilde
Box 8775
Universal City, CA 91608

I don't know about you, but for me the greatest joy from my computer comes during the development of some useful software. Each time I write a new program I try to be more creative, using lessons learned from past programs plus new techniques and subroutines that I've run across in the many magazines and club newsletters that I read.

Advancing from SET (X,Y) graphics to POKE and then string-packing techniques, you also advance in the artful use of BASIC. It's a very satisfying experience.

More Elegant INPUT

So with this noble goal in mind I decided to come up with an alternative to the INPUT statement. What originally prompted me to do this was my desire to have the majority of the screen "painted" white, with only a block of darkness in the center where input questions or prompts would appear.

With the INPUT statement, after typing in the data, you would hit ENTER, at which time the rest of the line (which I wanted to keep white) would be erased. Not only that, but I wanted a blinking cursor along with other automatic features I'll describe later.

Another negative aspect of the INPUT statement is that you can't enter commas and other string delimiters. Disk BASIC gives the LINE INPUT statement which solves that problem, but it's still basically the old inelegant INPUT statement.

The resulting input subroutine was developed and its first application was in a program for recording customer sales data for my own company.

The overall scheme uses an input subroutine to enter data into a temporary array I(EC). During the course of a single pass through the subroutine, I use 14 variables in array I(EC). I'll call these variables INPUT LINES I(EL).

After completing a pass through the input subroutine, the contents of I(EC) will be transferred to a master array for later output to tape or disk. Following this transfer, all the elements of I(EC) will be nulled (I(EC) = "") and the program will return to the beginning of the INPUT subroutine for new customer data.

After initialization, I run a program called Screen that creates and stores various graphic forms that will be recalled in the subroutine as well as other parts of Customer Sales Program, using the GSF Package by Racet Computes. One of GSF's many features is the ability to store a display in protected memory, so that it can be recalled by a simple BASIC statement.

At the end of Screen program I have the statement RUN SALES/BAS, which loads and runs the Customer Sales Program. This is where the INPUT subroutine comes in.

For purposes of this article I've added lines 1000 to 1020 to create a simple version of the graphic display used during data input.

The Subroutine

Line 20 clears 1000 bytes of string space and predefines the variables used as string or integer. Array I(EL) is dimensioned to hold 14 data items, and array S(N,X) is dimensioned for 20 customers.

Line 30 calls or creates a screen print routine. You can paint your screen however you like. For demonstration purposes, I'll use the subroutine at line 1000 for screen painting.

In line 100 PC% stands for the blinking cursor. Initially it's set equal to 32, which when inserted into a PRINT CHR$(PC%) statement prints a blank space. I'll discuss the cursor further at line 130. EL is the subscript for array I(EL) and is initially set for INPUT LINE #1 [EL(1)].

Line 110 is the FOR-NEXT loop that makes the whole thing work. Each I(EL) has a number, 1 to 14. The first time through the program, EL = 1. Therefore, the FOR-NEXT loop reads EP (EP = "print @ location" where I want the cursor to begin) and ES (ES = maximum number of characters allowed for an input line).

As the program progresses EL increments, or decrements. The FOR-NEXT loop rereads DATA lines 270–275 ending up with EP and ES equal to the correct data for I(EL).

RESTORE enables you to start from the beginning of the DATA line for each READ session. EZ counts the number of spaces over from the original position (EP). Initially it's set to zero.

The PRINT@ EP statement prints a string of blanks equal to the maximum number of characters (ES) allowed for that line. This is my way of clearing the line without erasing any of the white boundary.

Each I(EL) is defined below:

I(1) – customer number (3 characters)
I(2) – customer name (20 char.)
I(3) – customer business name (20 char.)
I(4) – address line 1 (20 char.)
I(5) – address line 2 (20 char.)
I(6) – address line 3 (20 char.)

The remaining lines are for customer phone numbers.

I(7) and I(11) – "H" for home or "W" for work (1 char.)
I(8) and I(12) – area code (3 char.)
I(9) and I(13) – 3-digit prefix (3 char.)
I(10) and I(14) – 4-digit number (4 char.)

As you can see, each phone number is made up of four separate variables. The reason for this is to allow for punctuation during data input, and yet end up with a single variable for each phone number containing no punctuation (or at least limited to 11 characters).

Later, in line 290 each set of

four variables is added together to create these two phone number variables [I(7) and I(11)].

In line 120 when EL is equal to, or greater than 7, this statement POKEs the necessary punctuation to the screen for the phone numbers. Since line 120 is within the main FOR-NEXT loop this punctuation will remain on the screen for the duration of the telephone number input.

The INKEY$ Loop

Line 130 is very important. It starts the INKEY$ loop. PC% = 175 − PC% creates a blinking cursor.

You'll recall that in line 100 PC% was set equal to 32. During the first pass of the INKEY$ loop, PC% becomes equal to 143. This is a graphic block when used in a PRINT CHR$ (PC%) statement. The next time line 130 is executed, PC% equals 32 again, resulting in a blank and so on. Voilá—a blinking cursor!

The statement PRINT@ EP + EZ, CHR$(PC%) therefore prints the cursor at (EP + zero) the first time, then EZ increments or decrements as characters are entered.

Lastly there's the statement I = INKEY$ which scans the keyboard setting string variable I equal to whatever the keyboard sees.

In line 140, PEEK (14400) and (14464) check to see if both the shift key and ← are pressed. If they're both down, the program branches to LINE 280.

If both keys are down the cursor will move backward erasing characters from the screen and memory. If you continue to hold the keys down, they act like a repeat key.

This is the only occasion where I PEEK the keyboard memory for control keys. For the remaining control keys I test the ASCII value of I [ASC(I)], for reasons you'll see later.

Line 150, as with any INKEY$ statement, tests for no key being depressed, in which case the INKEY$ loop is repeated. This no-key test must come before you test for the ASCII value of string variable I, because if I is null, you'll get an FC ERROR. On the other hand, it must come after the statement in line 140 in order to get the repeat backspace.

Line 160 checks the cursor character. If it's the graphic block, a blank gets printed in its place. This eliminates a possible stray cursor after hitting ENTER, CLEAR, ←, etc.

Lines 170 to 230 check to see if the key being depressed is a control key. These keys are SHIFT/E, SHIFT/S, CLEAR, ←, ↑ and ENTER. I test for the ASCII value, since it gives a single discrete value for individual keys or any combination of keys.

While writing this program I used a subroutine in line 9000: I$ = INKEY$:IF I$ = ""THEN 9000 ELSE PRINT@ 0, I$,ASC(I$);: GOTO9000. Every time I wanted to find out the ASCII value of a key or combination of keys, I'd run line 9000.

In line 170 if EL is greater than 6, we must be in the telephone number segment of the program. At that point, if SHIFT and the S key are held down, the program won't enter that particular set of four phone number variables. Instead it goes to the next phone number, or, if it happens to be skipping the second and last phone number entry, it branches to line 290, where the phone numbers are assembled.

Starting Over

At line 180 if ASC(I) = 32, you've hit the CLEAR key, indicating that you want to erase the line and start over. A string of blanks is printed at the cursor starting position (EP) for that time. The number of blanks is the same as the maximum number of characters allowed (ES). EZ is set to zero, and the temporary input, I(EL), is nulled. Note that EL stays the same.

At line 190 if ASC(I) = 8, you've hit the ← causing the cursor to backspace one step and

```
20 CLEAR (1000) : DEFINT E,N : DEFSTR I,P,Q,S,T : CLS : DIM I(15), S(20,8)
30 CLS : GOSUB 1000 : REM   *** CREATE OR RECALL SCREEN DISPLAY HERE ***
100 PC% = 32 : EL = 1
110 RESTORE : FOR EC = 1 TO EL : READ EP,ES : NEXT : EZ = 0 : PRINT @ EP, STRING$(ES,32);
120 IF EL > 6 POKE 15833,62 : POKE 15837,45 : POKE 15841,45 : POKE 15897,62 :
    POKE 15901,45 : POKE 15905,45 : PRINT @ 599, "↓HOME OR WORK";
130 PC% = 175 - PC% : I = INKEY$ : PRINT @ EP + EZ, CHR$(PC%);
140 IF PEEK(14400) = 32 AND PEEK(14464) = 1 GOTO 280
    REM     *** SHIFT/LEFT ARROW = REPEAT BACKSPACE ***
150 IF I = "" GOTO 130
160 IF PC% = 143 PRINT @ EP + EZ, CHR$(32);
170 IF (EL > 6) AND (ASC(I) = 115) THEN IF EL =7  THEN EL = EL + 4 : GOTO 110 ELSE GOTO 290:
    REM     *** SHIFT/S TO SKIP PHONE NUMBER ***
180 IF ASC(I) = 31 PRINT @ EP, STRING$(ES,32); : I(EL) = "" : EZ = 0 : GOTO130 :
    REM    ***  CLEAR KEY = ERASE AND START OVER ***
190 IF ASC(I) = 8 GOTO 280 : REM   ***  LEFT ARROW = SINGLE BACKSPACE  ***
200 IF ASC(I) = 91 THEN EL = EL - 1 : I(EL) = "" : PRINT @ EP, STRING$(ES,32); : GOTO 110 :
    REM    *** UP ARROW = REDO LINE ABOVE ***
210 IF ASC(I) = 13 THEN PRINT @ EP, I(EL) + STRING$( ES - LEN(I(EL)) + 1, 32); : EL = EL + 1 :
    IF EL < 15 GOTO 110 ELSE 290 : REM    ***  ENTER KEY = SAVE ENTRY ***
220 IF ASC(I) < 32 GOTO 130
230 REM   *** RESERVED FOR SHIFT/E = EDIT MODE ROUTINE ***
240 IF EZ <ES THEN I(EL) = I(EL) + I : EZ = EZ + 1 : PRINT @ EP, I(EL); : GOTO 130
250 IF EL <7 THEN 130 ELSE IF EL < 14 THEN PRINT @ EP, I(EL); : EL = EL + 1 : GOTO 110
260 GOTO 290
270 DATA 88,3,152,20,216,20,280,20,344,20,408,20,472,1,474,3,478
275 DATA 3,482,4,536,1,538,3,542,3,546,4
280 IF EZ > 0 THEN I(EL) = LEFT$( I(EL), EZ - 1) : PRINT @ EP + EZ - 1, STRING$( ES - EZ + 2, 32);:
    EZ = EZ - 1 : GOTO 130 : ELSE GOTO 130
290 I(7) = I(7) + I(8) + I(9) + I(10) : I(11) = I(11) + I(12) + I(13) + I(14)
300 FOR X = 1 TO 7 : S(N,X) = I(X) : NEXT : S(N,X) = I(11) : CLS :
    FOR X = 1 TO 15 : I(X) = "" : NEXT : N = N + 1 :
    CLS : PRINT @ 320, "TO CONTINUE ENTERING DATA [ENTER C]" :
    INPUT "TO PRINT IT OUT [ENTER P]"; Q : IF Q = "C" GOTO 30
305 FOR X = 0 TO N - 1 : FOR E = 1 TO 8 : PRINT "ENTRY:"; X + 1, "ITEM:"; E, S(X,E) : NEXT E :
    PRINT STRING$(60,42) : NEXT X : END
1000 T1 = STRING$(22,191) : T2 = STRING$(24,128) : T3 = STRING$(18,191)
1010 PRINT STRING$(64,191); : FOR X = 1 TO 9 : PRINT T1 + T2 + T3; : NEXT : FOR X = 1 TO 5 :
     PRINT STRING$(64191 ); : NEXT
1020 RETURN
```

Program Listing.

erase the last character. The erasing is done by a branch to line 280.

At line 200 if ASC(I) = 91 you hit ↑. This means you want to erase the line above. I put this in the program in the event you prematurely hit ENTER.

Line 210 responds to ENTER by processing the data contained in I(EL). First, I(EL) plus some cleanup blanks are printed at the proper location (EP). Subscript EL is incremented by one in preparation for entering the next INPUT line [I(EL + 1)].

If EL is less than 15 then the program branches to line 110 to begin entering the new I(EL). However, if EL is greater than 15, then all I(EL) have been entered and we branch to line 290 to complete data processing.

Line 220 acts as a protection against entering certain TRS-80 control characters. Refer to page C1 of the Level II manual.

Line 230 is reserved for my SHIFT/E, edit/correction subroutine. Normally, it would say "IF ASC(I) = 101 GOTO . . . the editing subroutine."

At line 240 if the number of characters in E(EL) is less than the maximum number allowed (ES), then the new character (I) is concatenated to the tail of I(EL). Since I(EL) is now longer by one character, EZ is incremented by one. Then I(EL) is printed at EP. This completes the INKEY$ loop so the program returns to line 130.

If, in the above line EZ equals ES, then the program drops to line 250. You won't be able to add any more characters to I(EL). However, if you're working on INPUT lines 7 through 13, the second half of line 250 comes into play. Basically, whatever you type after each of the four elements of the phone number will trigger an advance to the next input line.

Line 260 lets you jump over line 280.

Lines 270 and 275 are the DATA statements for reading in line 110.

Testing the Cursor

The first instruction in line 280 tests whether the cursor can move any further to the left. You'll recall that EZ increments every time the cursor moves to the right of the original starting position. As long as EZ is greater than zero, there's room to move backwards or left. If EZ is greater than zero then the backspacing and erasing occurs.

This is accomplished by the statement I(EL) = LEFT$(I(EL), EZ-1) which chops off the last character in string I(EL). The new value of I(EL) is printed, followed by a string of blanks equal to the maximum string length (defined by ES), minus the length of I(EL). The blanks are for erasing the rest of the line.

The new cursor position becomes EZ minus one, then the program returns to line 130.

Line 290 combines the four telephone number variables into two master variables (I(7) and I(11)].

Lines 300, 305 and 1000 through 1020 are included in order to make the program work for demonstration purposes.

Data is processed in line 300. The contents of the eight temporary storage variables [I(1) through I(7) and I(11)] are transferred to permanent storage in two-dimensional array S(N,X).

Line 305 prints out the contents of array S(N,X).

Lines 1000 through 1020 simply create a screen image.

In my Customer Sales Program line 300 is the beginning of the error correction and editing subroutine mentioned in line 200. Following that, I enter the remaining sales data and then store it all on disk.

All of these later program segments draw from the input subroutine. ■

Addendum

Since writing this article, Wilde has written a machine language version of the program. It is available from the author.

A high quality alternative.

Selectric Hard Copy

Michael W. Bickerton, M.D.
2631 Wharton St.
Philadelphia, PA 19146

It didn't take long after buying my TRS-80 for me to realize that I needed hard copy.

As a surgical intern, I didn't have $700 for a quality printer. But I did have $75 for a broken-down, used input-output IBM Selectric.

Although the typewriter was in bad shape, it had the electromagnets and contacts for use with a computer. All I had to do was figure out how to interface the two.

The only information I could find was Emerson Brooks' article, "Taming the I/O Selectric" (*Kilobaud*, June-July 1978). But Brook's method has several drawbacks.

First, he relies on software to perform all decoding and timing functions. Every carriage return uses 500 ms, even from mid-page.

Also, the software must be loaded every time you want to use the printer. I wanted a driver that would allow the Selectric to operate at its maximum speed of 15 characters per second. I also wanted to be able to simply turn on my TRS-80 and go to work.

My driver solves both problems. It will work with BASIC's LLIST and LPRINT commands without any software patches. It will also decode both upper and lowercase characters, and will work well with such programs as the Electric Pencil. You can even write your own word processor in BASIC, since the TRS-80 will output upper and lowercase strings.

If you don't have a TRS-80, an interface circuit will still let you use this driver.

The Selectric Typewriter

The Selectric mechanism has been used in computer terminals for years.

The models 731 (Photo 1) and 735 are available on the surplus market. The only difference is the 735's wider fifteen-inch mechanism.

The basic Selectric mechanism is sturdy and well designed. A type element the size of a golf ball rotates as you hit each key and types the characters. The printing quality is second to none and is suitable for everything from business letters to hardcopy memory dumps. Also, a number of different typefaces are available.

The Selectric uses a series of electromagnets to activate the proper mechanical clutches. A seven-bit Selectric code directly represents units of ball rotation (R1, R2, R2a and R5) and ball tilt (T1 and T2). The seventh bit is a shift that is actually a 180 degree rotate operation.

For example, the character "s" is represented by 0010001 (the format is shift/T2/T1/R5/R2a/R2/R1, zeros being active in all but shift and R5). To print an "s" the ball tilts two units (T2) and rotates four units (R2 + R2a). The character "S" is represented by 1010001, which is shift, tilt two and rotate four.

The standard Selectric code is called correspondence code, but many Selectrics use BCD (Binary Code Decimal) coding. These codes are not interchangeable, and use different balls. For example, the character "S" is represented by 0100110 in BCD. Also, BCD type balls have only upper case letters, numbers and special symbols.

To tell which kind your Selectric is, buy a standard type ball (correspondence). If your type-

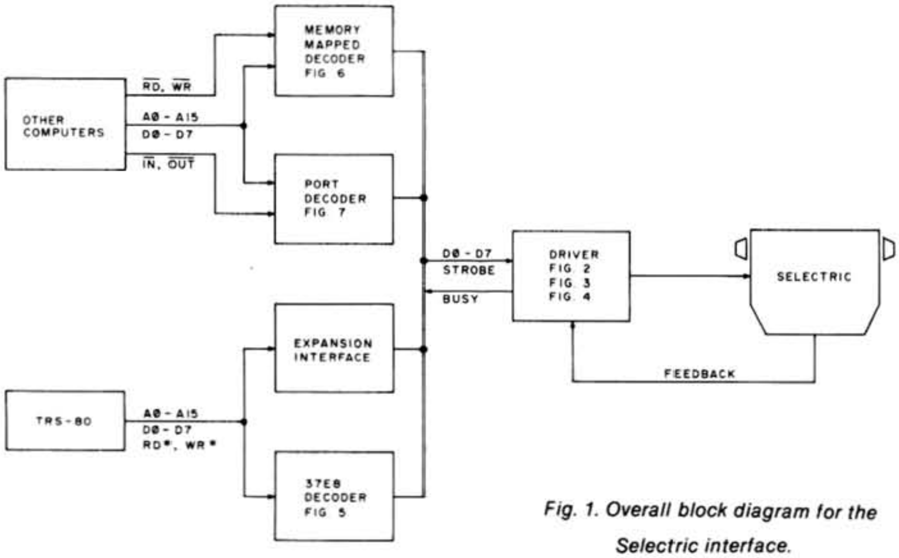


Fig. 1. Overall block diagram for the Selectric interface.

writer prints characters different from those on the keys, you have a BCD machine.

If you use the typewriter as an output device only, a BCD machine works fine. But if you wish to type from the keyboard, you must convert to correspondence. (See Robert M. Weil's "Converting Selectric Keyboards to Correspondence Code," *Kilobaud*, Dec. 1979 and Jan. 1980.)

The Selectric code is shown in Table 1. Note that the characters "<," ">", and "↑" are not available on standard type balls. My interface substitutes "l" for "less than," "g" for "greater than" and "E" for "exponentiation."

Photo 2 shows the undersurface of the Selectric mechanism and Photo 3 shows a close up of the printing (T1, T2, R5, R2a, R2 and R1) electromagnets. Input-output Selectrics also contain a bank of switch contacts, which are directly above the printing electromagnets (Photo 2). These input data from the Selectric to a computer. Since most microcomputers have their own keyboards, you can disconnect the switch bank.

Driver Circuitry

Fig. 2 is a block diagram and Fig. 3 is a schematic of the driver circuit. IC1 and IC2 form the heart of the driver. They are bipolar read-only memories that convert ASCII into correspondence code. A clever arrangement decodes control commands. If a control function is input D07 goes high. This is fed back to the input through R1 and C2. This serves to re-configure the read-only memories for decoding control commands.

When a strobe pulse is received, the decoded data is latched into eight-bit latch IC3 and IC4. Flip-flop IC11a is also set, providing a busy signal. The strobe pulse propagates through a series of monostable multivibrators. If a control command (carriage return, space, backspace, tab and index) is decoded, pin 2 of IC15 is high and the pulse from IC9a triggers IC10a. IC10a in turn strobes the control command to the proper relay through IC7 and half of IC6.

If a printing command is received, the situation is more complex. This time pin 4 of IC15 is high and the pulse from IC9a passes through IC12a to trigger IC9b. (IC10b is also triggered but has no effect.) IC9b activates IC5 and half of IC6 and prints the character. It also provides a strobe out to the strobe electromagnet.

If the typewriter is in the incorrect case the pulse triggers only IC10b and shifts up or down. IC12b detects the shift. IC9b then prints the character.

Any print or control operation activates one of five contacts in the typewriter. This is debounced by IC11b and clears IC11a, resetting the busy signal.

The 14 electromagnet drivers (Fig. 4) are identical, and use MJE6043 darlington transistors. The diodes in parallel with the relays protect the transistor from the high voltage induced when the electromagnet is turned off. Since the duty cycle is low, the transistors do not require heat sinking.

The power supply is a conventional full-wave rectifier with a regulated +5 V supply for the TTL integrated circuits. My Selectric had 48 V electromagnets that measure 475 ohms. If your electromagnets measure 125 ohms, then use the alternate 24 V supply.

Photo 2. Bottom view of the Selectric showing keyboard contacts (left), printing electromagnets (left lower) and control electromagnets (right).

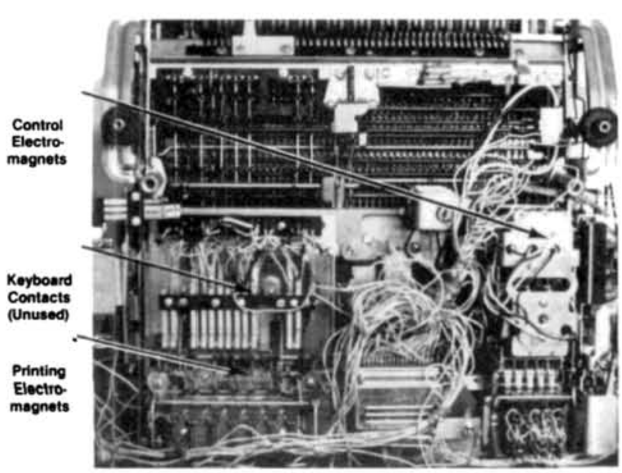
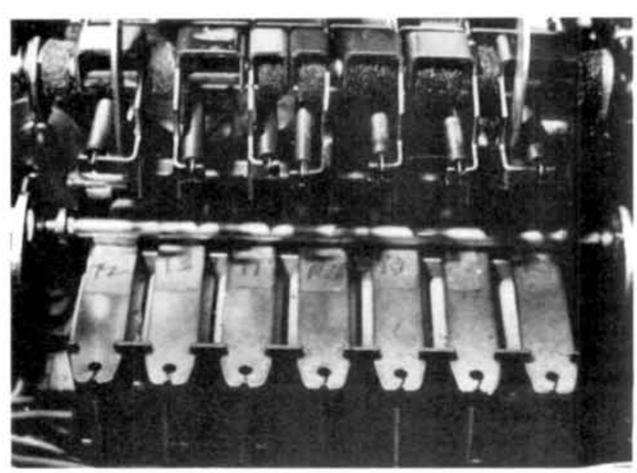


Photo 3. Closeup view of printing electromagnets. From left to right they are T2, Strobe, T1, R2a, R1, R2 and R5.



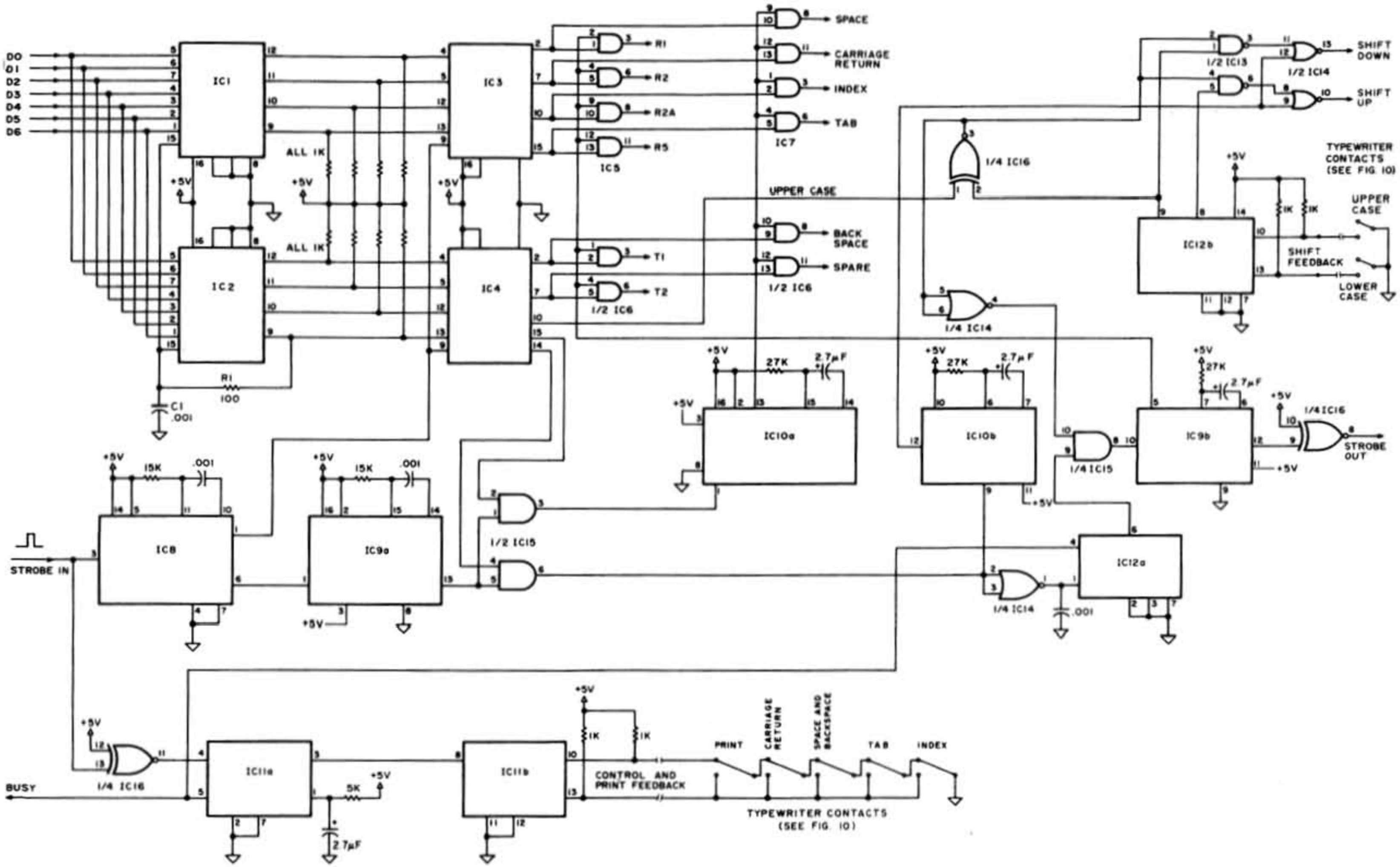
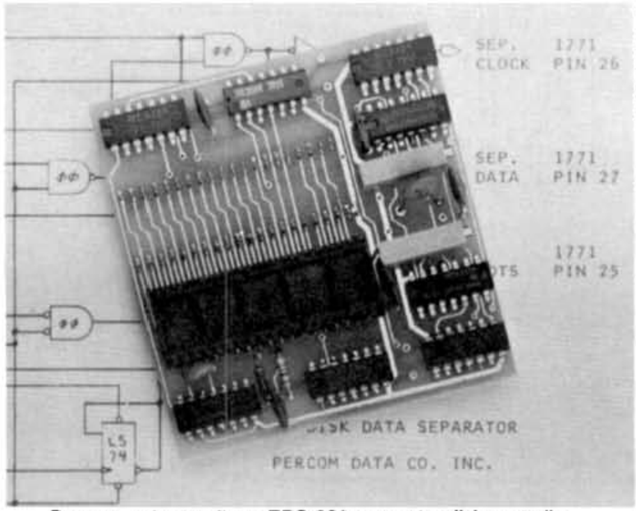


Fig. 3. Schematic diagram of the driver logic. The upper and lowercase contacts and the feedback contacts are located in the typewriter and should be connected as shown.

★★★ A PERCOM BULLETIN ★★★

Adapter for TRS-80* computer eliminates disk read errors



Percom adapter fixes TRS-80 computer disk controller.*

Garland, Texas — Harold Mauch, president of Percom Data Company, announced that the company is marketing a simple plug-in adapter for TRS-80* computers that corrects a design deficiency in the disk controller circuit.

The problem, which causes disk read errors, has been traced to Tandy's reliance on a circuit internal to the FD1771 controller IC to perform the function of separating clock and data pulses.

As explained in the *Backgrounder,* use of the internal chip circuit for reliable data-clock separation is a design shortcut which the manufacturer of the controller IC warns against.

The Percom solution, a PC card adapter called the SEPARATOR™, eliminates the problem by substituting an explicit data separator circuit — one which has been used reliably in Percom disk controllers since 1977 — for the internal IC separator circuit.

The SEPARATOR™ is installed without modifying the host system. The user merely removes the FD1771 IC from the host controller, installs the IC in the DIP socket on the SEPARATOR™ card, and plugs the adapter into the vacated socket of the host controller.

Percom cautions that opening the Expansion Interface of the TRS-80* computer, which is required to install the SEPARATOR™, may void the computer's limited 90-day warranty.

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

Payment for mail orders may be made by certified check, cashier's check or money order, or charged to a Master Card or VISA account. Texas residents must add 5% sales tax.

Percom Mini-Disk Drives Store More, Cost Less.

Percom mini-disk drives store more data, are more reliable, yet a 40-track Percom drive costs **$100.00 less** than a 35-track Tandy drive.

You can store over 102 Kbytes per side on Percom TFD-100™ 40-track drives; 197 Kbytes on one side of a TFD-200™ 77-track drive. A patch — supplied free on minidiskette — upgrades TRSDOS* for operation with the newer 40- and 77-track drives.

Both TFD-100™ and TFD-200™ models are available in one-, two- and three-drive configurations.

Prices start at $399 for a single-drive TFD-100™, $675 for a single-drive TFD-200™. Drives are supplied with heavy-duty power supplies. Metal enclosure is finished in compatible silver enamel.

See your nearby Percom dealer or order direct by calling toll-free 1-800-527-1592.

Five-Inch Disks Store More Than Eight-Inch Disks!

Garland, Texas — June 25, 1980 — Percom Data Company has begun production of a double-density disk controller adapter for TRS-80* Model I computers.

Harold Mauch, president of Percom, made that announcement here today, saying that data storage capacity using the adapter and double-density disk operating system — which is included — can be increased to as much as 294 Kbytes per minidiskette.

By comparison, the maximum storage for larger eight-inch disk systems used with the TRS-80* Model I computer is about 290 Kbytes.

Mauch said the PC card adapter, which plugs into the controller chip socket of the computer Expansion Interface, works equally well for either single-density or double-density storage, and users may continue to run programs under TRSDOS*, OS-80™ and other single-density operating systems with the adapter installed.

Price, for the plug-in adapter, the TRSDOS*-like double-density DOS and a utility for converting files and programs from single- to double-density format is expected to be $219.95.

BACKGROUNDER

CRC ERROR! TRACK LOCKED OUT!

by the Technical Staff
Percom Data Company

This problem started while we were studying an annoying problem with the TRS-80* computer. Disk drives sold by Percom are realigned and tested before shipment. We noticed, however, that some disk drives would pass the Percom inspection but just would not work reliably on the inner tracks with a TRS-80* computer. These drives were within the manufacturer's specifications, and would function perfectly on other disk systems Percom manufactures — "perfectly" here meaning more than 50 million bytes read without error!

The disk read data separation arrangement in the TRS-80* computer Expansion Interface uses an internal data separator of the FD1771 disk formatter/controller IC. Use of the FD1771 internal data separator is not recommended by Western Digital, the IC manufacturer. The following note appears on page 17 of the FD1771 data sheet:

> Internal data separation may work for some applications. However, for applications requiring high data recovery reliability, WDC recommends external data separation be used.

We suspected the data separator because the problem was most severe on disk inner tracks where storage density is highest and data separation is most critical.

To prove our point, a technician breadboarded a standard Percom data separator circuit, and configured it to plug directly into the FD1771 IC socket of the TRS-80* computer controller.

When connected to the TRS-80* computer, a troublesome drive functioned perfectly! We ran a BACKUP utility many times and never got a track lockout. Before we added the external data separator circuit to the computer, this same drive would always lock out tracks, and would have difficulty reading from the inner (higher number) tracks.

The Percom data separator circuit fixes the mini-disk controller of the TRS-80* computer. The type of drives being used is irrelevant; the circuit eliminates disk read errors resulting from the inability of the Tandy controller design to reliably separate clock and data signals when reading high density inner tracks.

PRICES AND SPECIFICATIONS SUBJECT TO CHANGE WITHOUT NOTICE.

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

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

✓Reader Service—see page 226

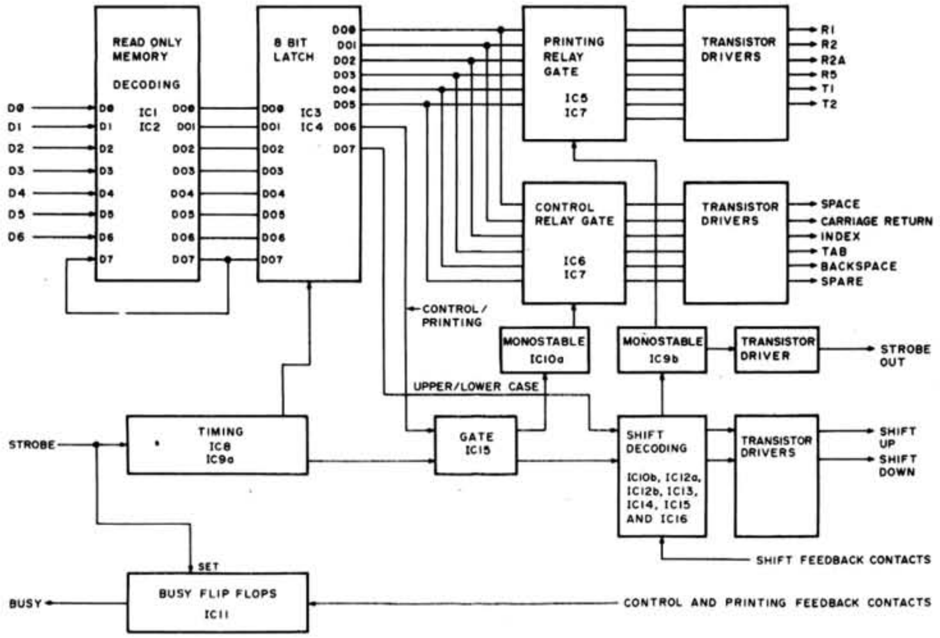


Fig. 2. Block diagram of the driver circuitry. With the exception of the feedback connections, all outputs on the right side of the diagram go to the Selectric electromagnets.

CHAR.	ASCII	CORR.
CR-LF	10–13	128
BS	24	128
INDEX	26	128
SPACE	32	128
!	33	127
"	34	85
#	35	126
$	36	121
%	37	117
&	38	125
'	39	21
(	40	112
)	41	113
*	42	124
+	43	70
,	44	12
-	45	0
.	46	22
/	47	9
0	48	49
1	49	63
2	50	54
3	51	62
4	52	57
5	53	53
6	54	52
7	55	61
8	56	60
9	57	48
:	58	77
;	59	13
< (1)	60	41
=	61	6
> (g)	62	15
?	63	73
@	64	118
A	65	92
B	66	96
C	67	108
D	68	109
E	69	101
F	70	78
G	71	79
H	72	97
I	73	84
J	74	71
K	75	100
L	76	105
M	77	95
N	78	102
O	79	89
P	80	69

Table 1. ASCII and correspondence codes. All values are decimal. When decoded into binary the correspondence code represents (MSB)Control Character/Shift/T2/T1*/R5/R2a*/R2*/R1*(LSB).*

Interface Circuitry

The driver connects directly to the TRS-80 expansion interface. If you don't own the interface, Fig. 5 presents a suitable one.

Radio Shack Level II BASIC reserves memory position 14312 decimal (37E8 hexadecimal) for the line printer. IC20, IC21 and IC22 decode this signal, and pin 6 of IC22 goes low when memory position 14312 is addressed.

Any time the TRS-80 outputs to the line printer, it places the ASCII code on the data bus, outputs 14312 to the address lines and pulses the WR* line low. ("*" denotes a signal that is normally high and goes low when active. Thus, $\overline{WR}$ = WR*.) When this happens pin 1 of IC23 goes high, latching the data into eight-bit buffer IC18 and IC19 and triggering the monostable IC24 to produce a strobe signal.

Then Level II Basic looks at memory position 14312 by pulsing the RD* line low, and waits until D7 (the busy signal) goes low before outputting another character.

To use another computer, you need to know if your language uses memory mapping or port addressing. Either of the circuits in Figs. 6 or 7 should be suitable.

Fig. 6 is a memory-mapped interface that uses four cascaded 74LS85 four-bit magnitude comparators. When the input address lines A0–A15 are equal to the address programmed on the dip switches SW1 and SW2, the cascade output (pin 6 of IC25) goes high. If the write line WR* goes low, then the data lines D0 to D7 are latched into eight-bit latch IC18 and IC19.

At the same time, IC24, a monostable multivibrator, gives a strobe signal. If the read line RD* goes low, the busy signal is input to the computer. On the TRS-80, D7 inputs the busy status, and D6, D5 and D4 input other status information (which I defeated by tying the lines to the appropriate logic levels). If your language uses another arrangement, connect pins 3, 5, 7 and 9 of IC17 to the appropriate data lines.

Fig. 7 shows a port-based interface circuit. 74LS85 magnitude comparators are again used to compare A0 through A7, with the programmed port address on dip SW1. Instead of RD* and WR* signals, the port interface uses IN* and OUT* lines. Restrictions on status information are the same as for the memory-mapped interface.

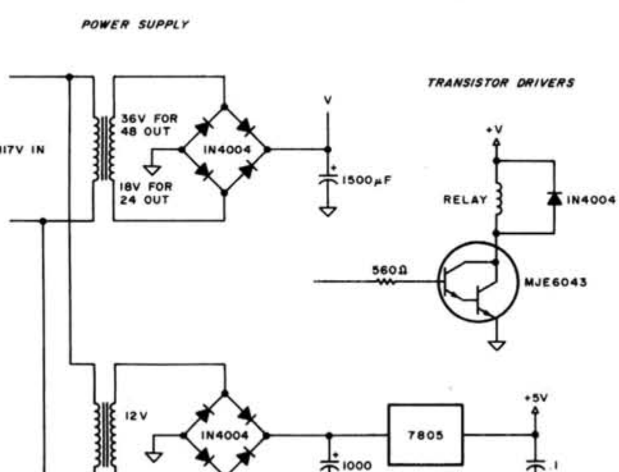


Fig. 4. The transistor drivers are all identical and use MJE6043 transistors. There are 14 transistor drivers; six for T1, T2, R1, R2, R2a and R5, one for the strobe electromagnet, five for the carriage return, space, backspace, tab and index electromagnets and one each for shift up and shift down electromagnets.

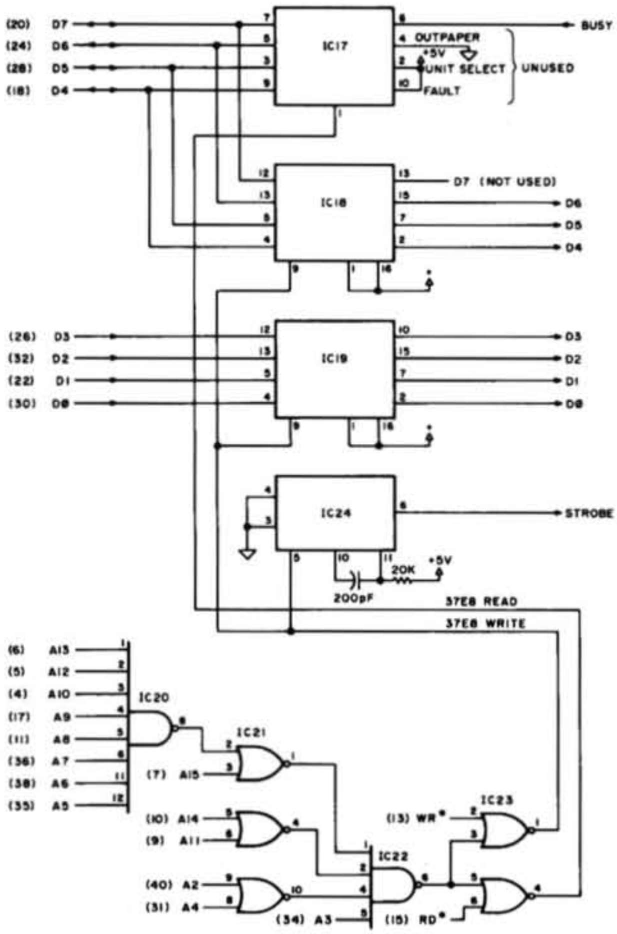
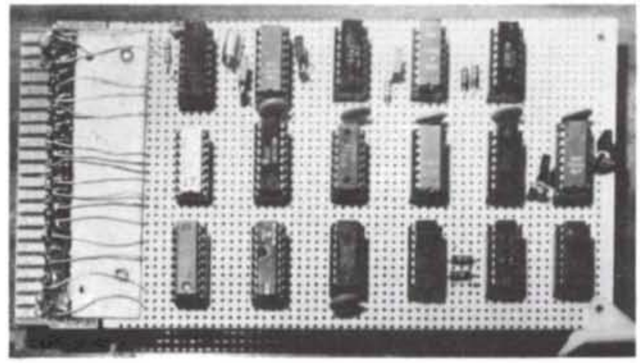


Fig. 5. The interface for the TRS-80. If you have the expansion interface, this circuit is not needed since the signals, D1 to D7, busy and strobe, are available from the expansion interface. Numbers in parentheses correspond to the TRS-80 expansion port edge connector.



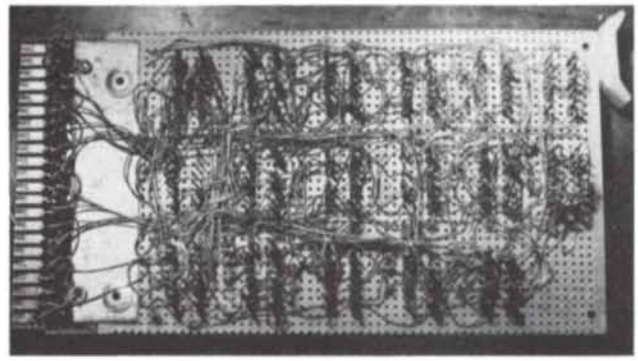


Photo 4 and Photo 5. The driver circuit board. Wire wrap sockets and a slit-n-wrap tool were used. The 44-pin edge connector is for use in a home made card cage.

Construction

Use any convenient method. The layout isn't critical, so printed circuitry or perfboard is suitable. I used wire-wrap sockets and a slit-n-wrap tool (Photos 4 and 5).

I brought all logic signals out through a 44-pin edge connector for use in a homemade card cage. The transistor drivers, interface circuit and power supply were constructed on separate perfboards.

I had some difficulty locating a 40-pin connector to plug into the TRS-80 expansion port. I finally used an AP edge connector (part number APP924065-36) with a 36-inch ribbon cable permanently attached.

Fig. 8 shows the TRS-80 expansion port pin designations. The TRS-80 uses odd pin numbering, which is printed on the circuit board.

Be especially careful when wiring the 40-conductor ribbon cable. I used an ohmmeter to trace each wire and engraved the word "top" on the upper surface of the 40-pin connector to ensure that I would not insert it upside down.

I used 25-pin D plugs and sockets with a 25-conductor ribbon cable to connect the Selectric. Fig. 8 shows my pin designations for the D connectors. My Selectric had a 50-pin plug attached to a heavy metal bracket. I substituted the 25-pin D socket and hooked up the cable.

I wanted to build the complete circuitry into the Selectric typewriter cabinet, but didn't have enough room. If you have the space, you might consider this option.

✓Reader Service—see page 226

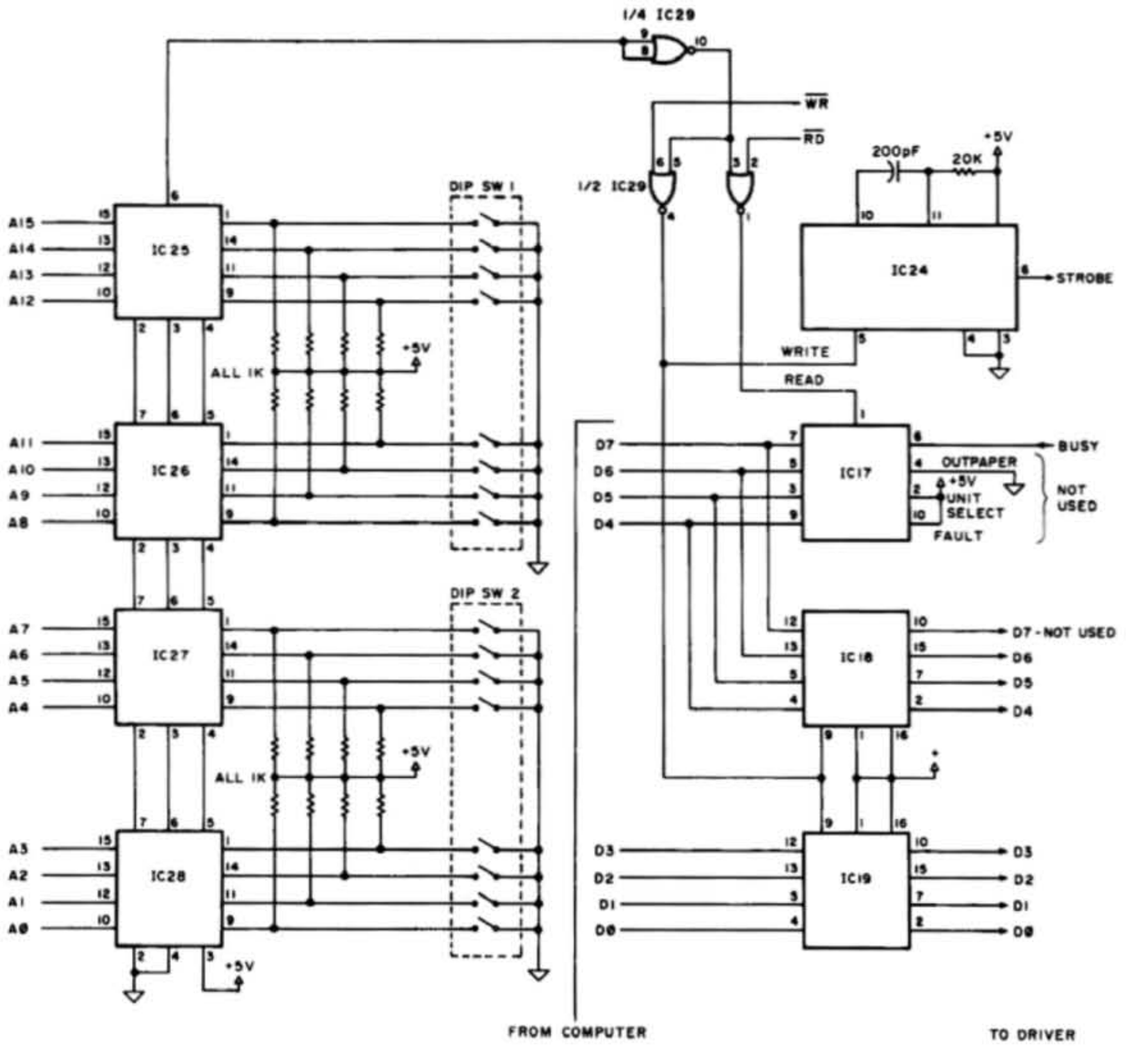


Fig. 6. This circuit will operate with computers that use a memory mapped output to the line printer. The memory location is programmed on SW1 and SW2.

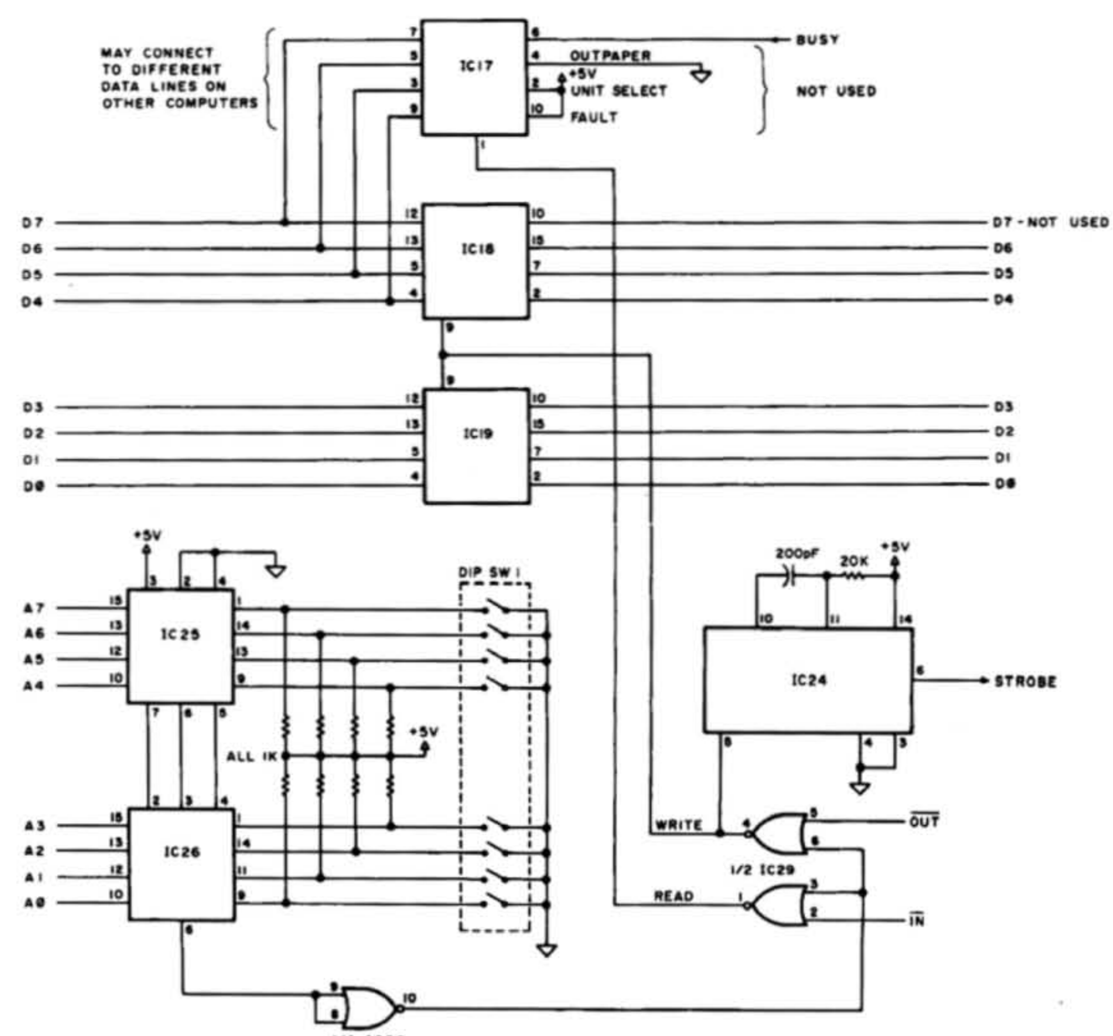


Fig. 7. This circuit will operate with computers that use a port addressing scheme to output to the line printer. The port address is programmed on SW1.

AUTHORIZED TRS-80® DEALER A301

COMPUTER SPECIALISTS

10% DISCOUNT Off List
26-4002
64K 1 Drive
$3499.00

26-1056 16K Level II System with Keypad	$688.00
26-1145 RS-232 Board	84.00
26-1140 "O" K Interface	249.00
26-1141 "16" K Interface	365.00
26-1142 "32" K Interface	476.00
26-1160 Mini Disk - Drive O	424.00
26-1161 Mini Disk - Additional	424.00
26-1154 Lineprinter II	720.00
26-1156 Lineprinter III	1799.00
26-1180 Voice Synthesiser	339.00
26-1181 VOXBOX	145.00
26-1104 Factory Upper Lower Case Modifaction Installed	70.00
26-1506 Scripsit - Tape	60.00
26-1563 Scripsit - Disk	85.00

15% DISCOUNT Off List
26-1054
4K Level II
$552.00

CENTRONICS

Fast 100 CPS Centronics
730 Printer - $675.00
Text Quality Centronics
737 Printer - $850.00

ALL OTHER R.S. SOFTWARE FURNITURE, STANDS, CABLES AND ACCESSORIES DEDUCT 10% FROM CATALOG PRICE

Novation Cat Modem . . $149.00
CCA Data Management System 72.00
Adventure Games
Games 1-9 each 14.00

Model II Cobol Compiler
$360.00
Cobol Run Time Package
$36.00

Acorn Software Products, Inc.

GAMES:	
Alien Invasion	$9.00
Stock Market	9.00
Star Trek	9.00
Block 'Em	9.00
Ting-Tong	9.00
UTILITIES:	
System Savers	14.00
EDUCATION:	
Language Teacher	18.00

MICROSOFT

Model I Basic Compiler	$180.00
Model II Basic Compiler	360.00

BASF

10-5¼" Diskettes	$45.00
10-8" Diskettes	47.00

1-800-841-0860 Toll Free Order Entry

MICRO MANAGEMENT SYSTEMS, INC. ✓72

No Taxes on Out Of State Shipments

Immediate Shipment From Stock.

DOWNTOWN PLAZA SHOPPING CENTER
115 C SECOND AVE. S.W.
CAIRO, GEORGIA 31728
(912) 377-7120 Ga. Phone No.

Full Factory Warranty on All Items Sold.

Largest Inventory In the S.E. U.S.A.

*TRS-80 is a registered trademark of the Tandy Corp.

✓Reader Service—see page 226

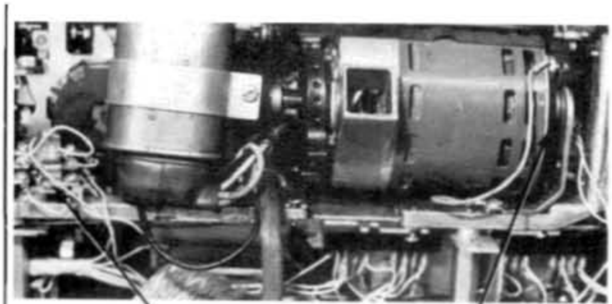


Photo 6. Rear view of the Selectric showing location of the index, space/backspace and tab feedback contacts.

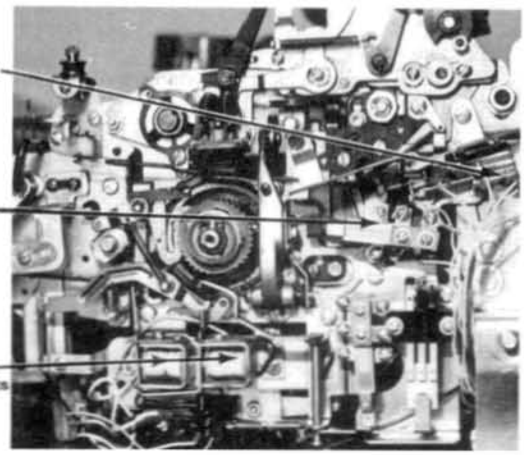


Photo 7. Right side view of the Selectric showing shift feedback contacts, shift electromagnets and carriage return feedback contacts.

My Selectric had a maze of wiring to feedback contacts, electromagnets and keyboard contacts, which I removed or disconnected. I brought the common connection to the electromagnets out to pin 1 on the D socket. My Selectric already has suppression diodes. Check the polarity of these diodes since it might be connected differently. If you have any questions, use suppression diodes on the transistor driver board instead.

Locate the printing and operational electromagnets and wire them to the D socket. Fig. 9 shows the location of these electromagnets. Next, locate all five of the feedback contacts and the two shift feedback contacts, and wire them according to the schematic diagram (Fig. 3). See Photos 6, 7 and 8 and Fig. 10 to locate these contacts.

The final connection is the ground. Attach it directly to the frame. I used the connector bracket screw.

You can program the read-only memories with the data in Table 5. If you don't have the facilities, I will sell preprogrammed 74S387 ROMs. I'll also consider programming these memories for special features.

Operation

This interface can be used directly with Basic's LLIST and LPRINT commands. Therefore, programs that use the TRS-80 line printer will work without modification.

If you have a program with a large number of PRINT statements that you wish to output to the Selectric, you can retype all PRINT statements, substituting LPRINT.

Here's a neat trick that will accomplish the same thing:

```
POKE 16414,141:POKE 16415,5
```

Address	Data	115	137
12	200	116	146
13	200	117	131
14	200	120	105
15	200	121	104
30	200	122	135
32	200	123	121
40	200	124	147
41	177	125	156
42	125	126	136
43	176	127	120
44	171	130	157
45	165	131	101
46	175	132	167
47	025	133	145
50	160	136	200
51	161	137	100
52	174	140	066
53	106	141	034
54	014	142	040
55	000	143	054
56	026	144	055
57	011	145	045
60	061	146	016
61	077	147	017
62	066	150	041
63	076	151	024
64	071	152	007
65	065	153	044
66	064	154	051
67	075	155	037
70	074	156	046
71	060	157	031
72	115	160	005
73	015	161	004
74	051	162	035
75	006	163	021
76	017	164	047
77	111	165	056
100	166	166	036
101	134	167	020
102	140	170	057
103	154	171	001
104	155	172	067
105	145	212	202
106	116	213	202
107	117	214	202
110	141	215	202
111	124	230	220
112	107	232	204
113	144	240	201
114	151	336	210

Table 5. Programmable read only memory (PROM) programming data. A 1702 or 2708 ultraviolet erasable PROM could be used, but I recommend a more permanent bipolar PROM such as the 74S387. All values in this table are octal for easy decoding into binary.

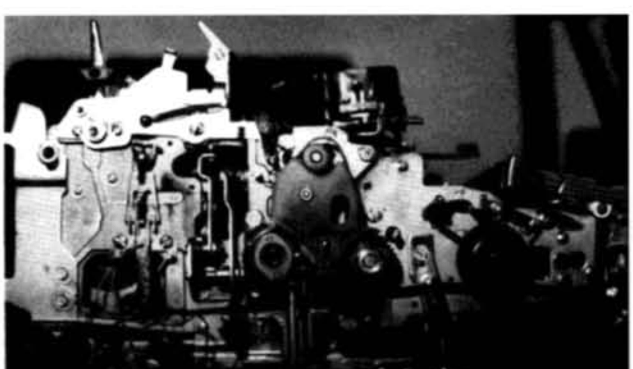


Photo 8. Left side view of the Selectric. The set of contacts in the middle of the photo operate off of the bilobed cam. The inner contacts and cam are used for the print feedback contacts.

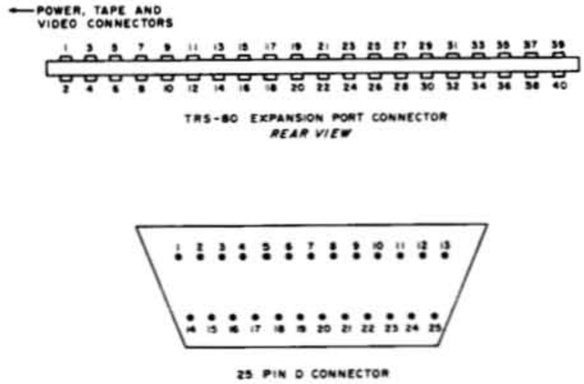


Fig. 8. Pin numbering for the TRS-80 expansion port edge connector and the 25-pin D connector used to connect the driver to the Selectric typewriter.

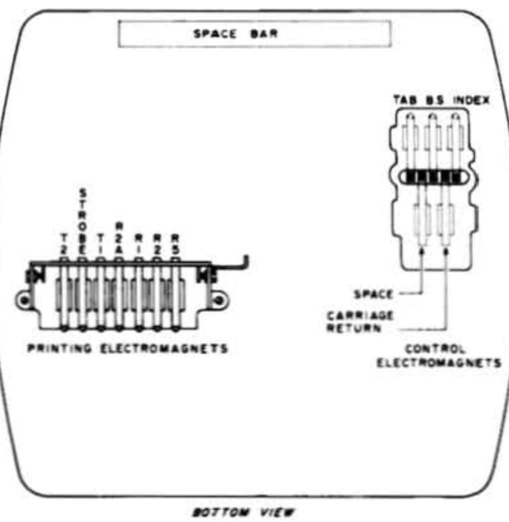


Fig. 9. Location of the printing and control electromagnets viewed from the bottom of the typewriter.

Now, all output ordinarily displayed on the video screen is typed by the Selectric.

If you type these POKE statements while in the command mode in BASIC, be sure to type them together with a colon separating them. Otherwise, after entering the first POKE, you will lose the video display.

If this happens, correctly type the second POKE statement (with the video screen blank) and the Selectric will respond with the familiar READY. Now, anything typed on the TRS-80 keyboard will be echoed by the Selectric.

A better way to use this feature is to place the POKE commands as the first statement in the BASIC program you wish to alter. This will change all PRINTs to LPRINTs. If you wish to disable this feature (PRINT = PRINT) then use the following command:

```
POKE 16414,88:POKE 16415,4
```

This can be used anytime in a program, in the command mode, or in the final statement to reactivate the video before terminating the program.

In addition, these POKE commands can be used to change LPRINTs to PRINT by using:

```
POKE 16414,88:POKE 16415,4
```

To revert back to normal

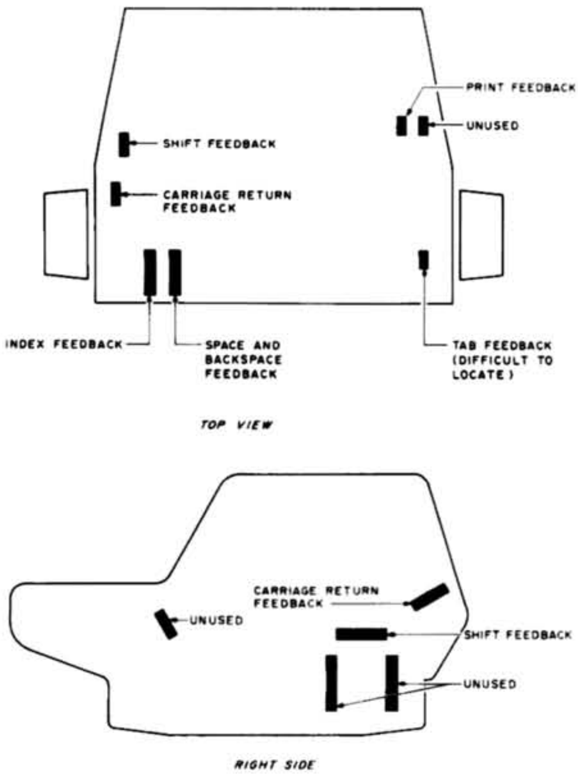


Fig. 10. Location of the feedback contacts.

(LPRINT = LPRINT) use:

```
POKE 16414,141:POKE 16415,5
```

Another way to output to the Selectric is by POKEing to memory location 14312. Program Listing 1 causes the Selectric to type all printable characters.

This program outputs ASCII values 33 to 91 and 97 to 122. After POKEing each number to location 14312, the program must wait until D7, the busy signal, is cleared. This is done by the PEEK command in line 50, which causes an endless loop to line 50 until the busy signal goes low.

You can also tell the Selectric to carriage return and line feed: POKE 14312,10, back space: POKE 14312,24, index: POKE 14312,26, space: POKE 14312,32 and tab: POKE 14312,94. These features may be valuable if you are writing a word processor in BASIC.

To demonstrate upper and lowercase characters, try the following program:

```
10 INPUT ST$:LPRINT ST$
```

When you run this program you will be asked to input any string. Suppose you respond by typing "Kilobaud Microcomputing," pressing the shift button for "K" and "M," as you would on a normal typewriter. The Selectric will respond by typing "kILOBAUD mICROCOMPUTING."

The TRS-80 will invariably switch cases. To correct this you must write your own input routine.

To display and output upper and lowercase characters properly, you'll need to modify the TRS-80's hardware and software. A number of modification kits are on the market. The one I have been using is KVP by Lance Micklus.

Conclusion

I've been using the Selectric with my interface and driver for several months and it has performed flawlessly.

At first, an occasional mistyped character, or an extra space or dash inserted itself at random, but these mistakes resulted from a misaligned typewriter. I carefully realigned my Selectric and, other than a broken rotate tape, I have had no further problems.

Excellent references for this project include IBM's Service Manual (part no. 241-5257-0), Parts Manual (241-5990-0) and their Price List (241-5158-3).

The Service Manual is particularly well written and well illustrated, and is almost a necessity if your machine needs adjusting. Since IBM will not service machines that do not bear the IBM decal, you may have to make your own adjustments. ■

```
10 FOR I = 33 TO 91:GOSUB 40:NEXT I:I = 10:GOSUB 40
20 FOR I = 97 TO 122:GOSUB 40:NEXT I:I = 10:GOSUB 40
30 END
40 POKE 14312,I
50 IF PEEK (14312) g100 GOTO 50 ELSE RETURN
```

Program Listing 1.

NEW

DISC DRIVES
FOR TRS—80*
"FLIPPY"—250K bytes/disc
FAST—5ms Track to Track Access

REAL VALUE

AEROCOMP offers the best value in microcomputer disc drives on the market today! Reliability, features and cost tough to beat. We deliver...and we stand behind our products, as evidenced by the only FREE TRIAL OFFER in the industry. Examine your systems needs and order today!

MYSTERY REMOVED

There appears to be some confusion in the terminology used to describe disc drives and their features. Here's what we mean:

- **FLIPPY** Allows the use of both sides of a diskette with a single-headed drive by simply turning the diskette over (Model 40-1).
- **TRACK DENSITY** specified in tracks per inch (TPI). Refers to the number of tracks per radial inch on the diskette. Typically 48 TPI=40 usable tracks and 96 TPI=80 useable tracks.
- **DOUBLE DENSITY** refers to recording density in bits per inch (bpi). Typically single density means data can be recorded up to 2,938 bpi; double density means data can be recorded up to 5,876 bpi.
- **DOUBLE-SIDED** refers to number of read/write heads. Single-sided is one head, read/write one side only; double-sided is dual heads allowing read/write operations on both sides of the diskette. A double sided drive appears as two separate drives to the controller.
- **CAPACITY** unformated capacity is the total amount of storage space available on a diskette. Typically 125K bytes on a 40 track 5.25in. diskette. Formated capacity is the total USABLE storage space on a diskette. Typically 102K bytes on a 40 track 5.25in. diskette.
- **ACCESS TIME** the time required for the head to move from one track to the next. Typically 5 to 40 milliseconds (ms).

FREE TRIAL OFFER

Order your AEROCOMP Disc Drive and use it with your system for up to 14 days. If you are not satisfied for ANY REASON (except misuse or improper handling), return it, packed in the original shipping container, for a FULL REFUND. We have complete confidence in our products and we know you will be satisfied! ORDER TODAY!

WARRANTY

We offer you a 90 day unconditional warranty on parts and labor against any defect in materials and workmanship. In the event service, for any reason, becomes necessary, our service department is fast, friendly and cooperative.

100% TESTED

AEROCOMP Disc Drives are completely assembled at the factory and ready to plug in when you receive them. Each drive is 100% bench tested prior to shipment. We even enclose a copy of the test checklist, signed by the test technician, with every drive. AEROCOMP MEANS RELIABILITY!

ORDER NOW

- **MODEL 40-1** DISC DRIVE $339.95ea. Single-sided, "Flippy", 48TPI. (40 track; single density unformated 125K bytes/side; double density unformated 250K bytes/side).
- **MODEL 80-2** DISC DRIVE $439.95ea. Double-sided, 48TPI. (80 track/40 per side; single density unformated 250K bytes; double density unformated 500K bytes).
- **MODEL 80-1** DISC DRIVE $439.95ea. Single-sided, "Flippy", 96TPI. (80 track; single density unformated 250K bytes/side; double density unformated 500K bytes/side).
- **MODEL 160-2** DISC DRIVE $595.95ea. Double-sided, 96TPI. (160 track/80 per side; single density unformated 500K bytes; double density unformated 1 megabyte).

All models are capable of single or double density and are complete with power supply and silver enclosure. Send for information on AEROCOMP 2- and 3-drive systems available in 40 and 80 track.

- **SELECT EITHER A 2-DRIVE OR 4-DRIVE CABLE FOR USE WITH YOUR DRIVE(S):**
  - **2-DRIVE CABLE** (for use with 1-or 2-drive systems) **$24.95ea.**
  - **4-DRIVE CABLE** (for use with 1-, 2-, 3-or 4-drive systems) **$34.95ea**
  - Add $1.25 shipping and handling
- **MINI DISKETTES** (5.25 in), Box of 10 **$29.95** add $1.25 shipping and handling
- **DISC OPERATING SYSTEMS**
  - NEWDOS+(40 TRACK) $109.00
  - NEWDOS/80 (80 TRACK) $149.00

★ SPECIAL COMBINATION OFFER ★

● Model 40-1 Disc Drive	$339.95
● 2 Drive Cable	24.95
● Disc Operating System (NEWDOS+)	109.00
● Freight	5.25

Reg. $479.15
Special $399.95

SPECIAL COMBO EXCEPT WITH NEWDOS/80
Reg. $519.15 **Special $429.95**

To order by mail, specify Model Number(s) of Drive, cable, etc. (above), enclose check, money order, VISA or MASTERCHARGE card number and expiration date, or request COD shipment. Texas residents add 5% sales tax. ADD $4.00 per drive for shipping and handling. Please allow 2 weeks for personal checks to clear our bank. No personal checks will be accepted on COD shipments-cash, money orders or certifed checks only. You will receive a card showing the exact COD amount before your shipment arrives. Be sure to include your name and shipping address. WE SHIP PROMPTLY! In the event there is a slight delay, you will be notified of the shipping date and we will NOT deposit your money order or charge your bankcard until the day we ship!

CALL TOLL FREE FOR FAST SERVICE
(800) 824-7888, OPERATOR 24
FOR VISA/MASTERCHARGE/C.O.D. ORDERS

California dial (800) 852-7777, Operator 24. Alaska and Hawaii dial (800) 824-7919, Operator 24.

TOLL FREE LINES WILL ACCEPT ORDERS ONLY!

For Applications and Technical information, call (214) 337-4346 or drop us a card.

COMPARE AND BUY AEROCOMP!

	"FLIPPY"	ACCESS TIME (track to track)	HEAD LOAD SOLENOID	DISC EJECTOR	CAPACITY (unformated single density)	EASY-ENTRY DOOR	FREE TRIAL
AEROCOMP	YES	5ms.	YES	YES	250K bytes (both sides)	YES	YES
RADIO SHACK*	NO	40ms.	YES	NO	109K bytes	NO	NO
PERCOM	YES	25ms.	YES	NO	250K bytes (both sides)	YES	NO
MPI	NO	5ms.	YES	YES	125K bytes	YES	NO
SHUGART	NO	40ms.	YES	NO	109K bytes	NO	NO
SIEMENS	NO	25ms.	YES	NO	125K bytes	YES	NO
TANDON	NO	5ms.	NO	NO	125K bytes	NO	NO
PERTEC	YES	25ms.	YES	NO	250K bytes (both sides)	NO	NO
BASF	NO	12ms.	YES	NO	125K bytes	NO	NO

Factual material from current manufacturer's data sheets is believed reliable but cannot be guaranteed, comparing Aerocomp Model 40-1 to similar models.

The TRS-80* expansion interface limits the track to track access time to 12ms.

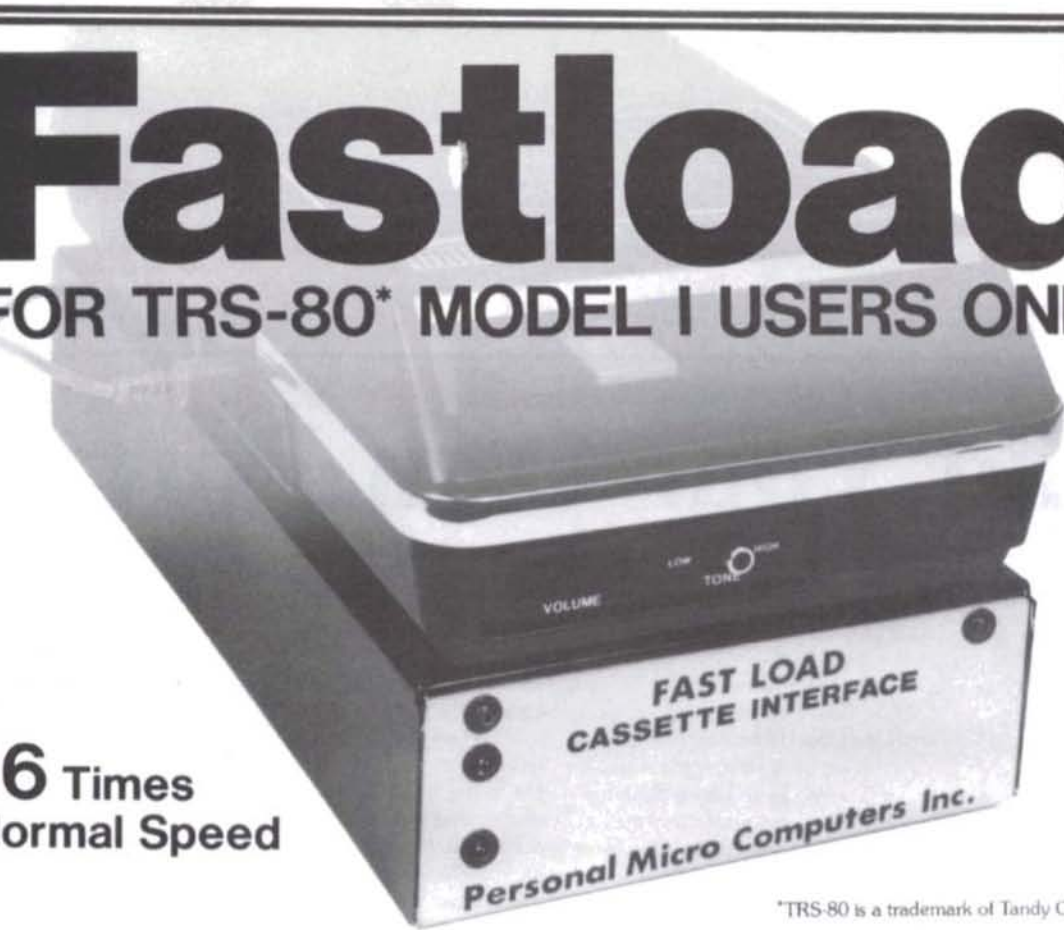
*Trademark of Tandy/Radio Shack.

Dealers inqiries invited

AEROCOMP
Redbird Airport, Bldg. 8
P.O. Box 24829
Dallas, TX 75224 ✓387

Fastload

FOR TRS-80* MODEL I USERS ONLY



16 Times
Normal Speed

*TRS-80 is a trademark of Tandy Corp.

- High speed load TRS-80* Level II cassettes
- Input 15K byte Level II program in 15 seconds
- Search BASIC or SYSTEM programs by name

Unlike other high speed tape input devices, FASTLOAD uses standard format cassettes. Therefore, there is no need to re-record on other media. At 8000 baud, FASTLOAD is faster than disk for short programs. FASTLOAD reads tapes at the fast-forward speed of the CTR-41 cassette recorder. The recorder can also be used for CSAVE at the normal speed.

FASTLOAD connects to the 40 pin I/O or to the Expansion box. The control program does not use computer memory because it is in a built-in PROM. Other valuable features are keyboard debounce program, automatic key repeat routine and keybeep via cassette speaker. Price is $188.00 for FASTLOAD and $95.00 for the modified CTR-41 recorder.

✓112

Personal Micro Computers Inc.

475 Ellis Street, Mountain View, CA 94043 (415) 968-1604

An interface to the outside world.

Build Your Own Port

James S. Hawkes, Ph.D. and Grady Reese

A computer port is similar to a sailing port, because both serve as points of exchange. At a port a ship unloads its cargo onto the dock where it is stored and later shipped to its mainland destination.

Similarly, the computer port receives information from an external device and holds it until it is called by the controlling machine. It also stores information sent by the controlling machine, which it can later send to an external device.

A port that only transmits information is an output port; while a port that can only receive information is an input port. Certain bi-direction ports can both receive and transmit.

The Parallel Port

A port permits you to communicate with external devices. Frequently, ports are used in conjunction with other equipment to monitor environmental conditions such as temperature, wind velocity, voltage, and pressure.

Ports can also be used to control simple on-off functions for hot water heaters, televisions and lights. Through a parallel port eight bits of information (in most microcomputers) are transmitted simultaneously. In some ports data transmission can reach speeds of over 100,000 bytes per second. Because of these high speeds control signals are necessary between the computer and the external device. These are called handshaking signals.

There are two types of parallel I/O. One is called "accumulator I/O" because the byte to be handled by the port must be loaded into the accumulator before it can be transmitted to the port. The second is called "memory mapped I/O" because a memory read or write instruction is used to access the peripheral. The TRS-80 parallel printer port on the expansion interface is a memory mapped port.

With a small investment of time and money you can build your own port, which can be directly attached to the TRS-80 expansion bus located at the rear of the keyboard, or to the screen printer bus on the expansion interface.

The simple application is made possible by a large scale integrated circuit called a programmable peripheral interface (PPI). The PPI used in this design is the Intel 8255.

The 8255 PPI Chip

The 8255 PPI, originally cre-

INPUT 1	INPUT 2	OUTPUT
LOW	LOW	LOW
LOW	HIGH	HIGH
HIGH	LOW	HIGH
HIGH	HIGH	LOW

Table 1.

Table 2
Port Selection

TRS-80 Address Lines A1	A0	8255 Port Selection
0	0	A
0	1	B
1	0	C
1	1	Control

Table 2. Port Selection.

```
10 REM *** TESTING AND DEMO PROGRAM FOR THE PPI-80 ***
20 REM *** OPS - RECEIVING PORT          ***
30 REM *** P   - PORT NUMBER             ***
40 REM *** N --- NUMBER TRANSMITTED      ***
50 REM *** E --- VALUE RECEIVED          ***
60 REM ***********************************
70 OUT 131,137:REM *** INITIALIZE PORT ***
80 CLS:N=0:OP$=" ":P=0:E=0: REM *** INITIALIZE VARIABLES ***
90 INPUT "ENTER NUMBER TO BE TRANSMITTED";N
100 IF N>255 OR N<0 THEN 90
110 INPUT "SELECT OUTPUT PORT (A OR B)";OP$
120 IF OP$="A" THEN P=128: REM *** SET PORT NUMBER ***
130 IF OP$="B" THEN P=129: REM *** SET PORT NUMBER ***
140 IF P<>128 AND P<>129 THEN 110: REM *** TESTS FOR ILLEGAL ENTRY ***
150 OUT P,N: REM *** SEND NUMBER TO SELECTED OUTPUT PORT ***
160 E=INP(130) : REM *** READ PORT C ***
170 PRINT @ 120,E
180 FOR I=1TO1000:NEXTI: REM *** DELAY LOOP ***
190 GOTO 80
```

Program Listing.

ated to be used with the 8080A microprocessor, can be used with most of the current microcomputer chips including the Z-80.

By sending the appropriate control word in BASIC or machine language to the 8255's control port, you configure the chip's 24 I/O pins in one of three ways: as three independent unidirectional input or output ports; as two uni-directional input or output ports with handshaking signals; or as one uni-directional input or output port plus one bi-directional port with complete handshaking signals on both.

The three major parts of the 8255 (see Fig. 1), are the computer interface, the peripheral interface and the internal control logic.

The computer interface consists of eight data lines, six control lines, and two power supply lines. The data lines D0–D7 are connected to the computer's data lines and are used to transfer data to and from the computer's data bus.

Next there are the address lines A0 and A1. These two lines use their four possible binary combinations (00,01,10,11) to select one of the four ports within the 8255.

A fundamental requirement for the parallel port is that it must reside at a specific address or addresses. The address chosen is a function of the type of parallel I/O you select. The chip select active low line makes use of the decoded address to select the PPI for an I/O operation.

This brings us to the read (active low) and the write (active low) lines. These two lines are used to tell the chip whether to read or write. They are connected to the computer's memory read (RD) and memory write (WR) lines for memory mapped I/O or to the IN and OUT lines for accumulator I/O. The RESET (active high) will set all ports to their input mode. This initializes the chip and allows it to be reconfigured.

The next group of pins supports the peripheral interface. Data is transferred to and from external I/O devices through 24 pins which constitute three eight-bit ports. These are called port A (PA0–PA7), port B (PB0–PB7), and port C (PC0–PC7). The user is able to control the function of the I/O lines by sending a control word to the chip's programmable control port.

As stated above, the control word determines for each port one of three operating modes:

(1) Mode 0—Simple input and output. In this mode the 8255 provides two eight-bit I/O ports (A and B) and two four-bit ports which can operate as one eight-bit port (C). Each port is latched in its output configuration and unlatched in its input configuration. Additionally, each port can be individually defined as input or output.

(2) Mode 1—Strobed input and output. This mode provides two uni-directional 8-bit input or output ports (A and B) with port C providing the necessary handshaking signals. Data is latched for both input and output at ports A and B.

(3) Mode 2—Strobed bi-directional I/O. In this mode there are two usable ports: one bi-directional I/O port (A) and one uni-directional I/O port (B), both with full handshaking provided by port C.

PPI-80 Port Design

The port contains four fundamental parts: the decoder logic, the 8255 PPI, the reset logic, and the power supply.

Because the port is designed to be used with accumulator I/O, only eight address lines A0–A7 must be decoded. Normally, addresses are decoded by using inverters and eight input NANDs (see Rod Hallen's article in *Kilobaud Microcomputing*, January, 1980). While this technique is both straightforward and inexpensive, there is one drawback; the port address cannot be changed without physically rewiring the circuit.

To be able to use the port for a number of different applications, and because of other peripherals which could conflict with some of the four addresses needed to implement the port, it would be advantageous to make it easy to change the port's address.

A simple and inexpensive way to do this uses a DIP switch (S1), a small resistor network (R1–R6), and two exclusive-OR (XOR) ICs, U3 and U4, instead of the traditional inverter. (See Fig. 1.)

The decoding circuit utilizes two 74LS86 integrated circuits, which are quad-two input XOR gates. Remember that the XOR gate produces 1 (high) if the two input signals are of the opposite type, and 0 (low) otherwise. See Table 1, XOR truth table.

If one of the inputs is high, then the other input must be low to obtain a high as an output. Setting the DIP switch (S1) changes the input to one side of XOR gate and causes it to function as an inverting or non-inverting buffer with respect to

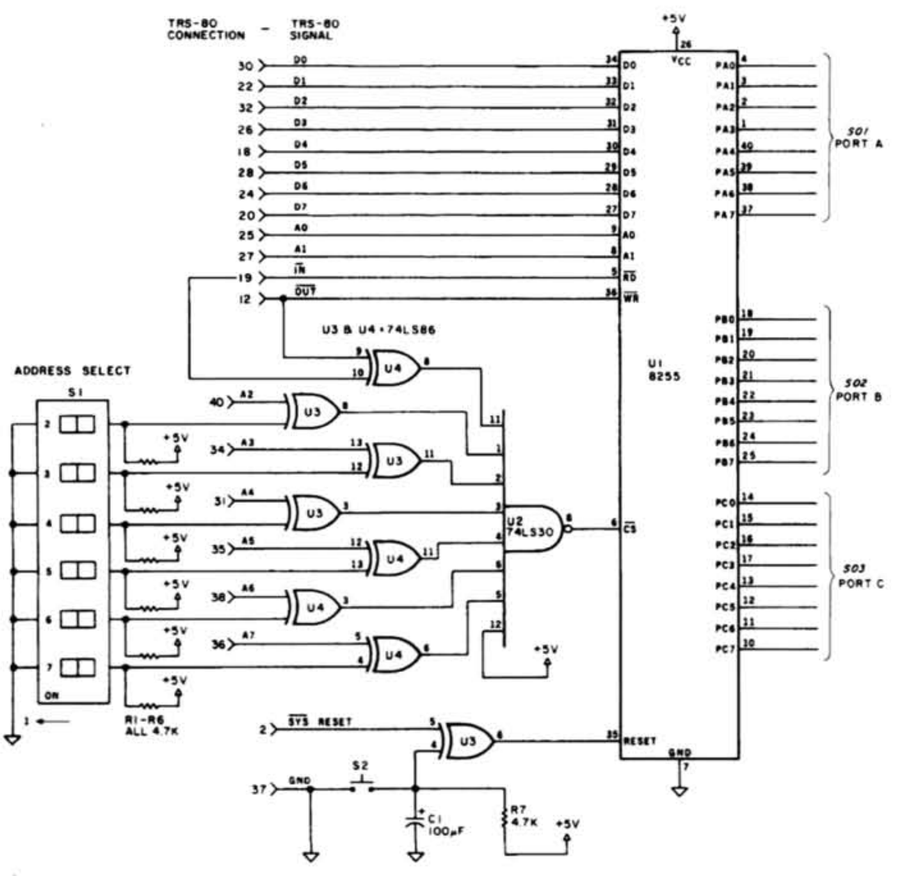


Fig. 1

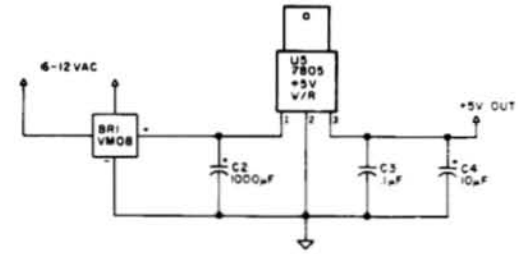


Fig. 1a.

the other input presented at the gate.

To invert an address line, set the appropriate switch on S1 so that the input line to the XOR gate is pulled high (one). Then if one is presented on the other input, an address line, the gate will output at 0, and if a low is presented, it will output a 1. In this state the XOR is functioning as an inverter. Similarly, by setting S1 so that one of the inputs to the gate is low, it acts as a noninverting buffer. By setting S1 switch to invert or not to invert, you can select the address at which you wish to locate your port.

Since the TRS-80 is continuously selecting ROM and RAM addresses, the address lines are always changing. It is likely that the address you select with S1 will be decoded, since the port decodes the lower eight lines only. For this reason we XOR'd the input (IN active low) and Out (OUT active low) signals and used the resulting output as an input to the 8-input NAND.

For the correct address to be decoded and supplied to the 8255 as the CS (active low), it must be present, and either the IN or OUT lines must be low. When the port is properly addressed by the TRS-80, all of the inputs at U2 will be high, and the resulting output will enable the PPI.

After the 8255 is enabled, it must be told what function to perform. That is, if the TRS-80 should read information from the PPI, the IN line must be low. When the CS and IN go low, all that is left to be done is to specify whether I/O port PA, PB, PC or the control port is involved.

The address lines A0 and A1 define which of these ports will be selected. (See Table 2.)

In order to affect an input to the TRS-80 from port C; IN, CS, and A0 must be low while A1 is high. When this condition occurs, the 8255 latches the contents of port C on the data lines, D0–D7, until IN goes high. Amazingly, all this can be done with one BASIC statement.

Sending data to the PPI from the TRS-80 is accomplished in much the same fashion except the OUT (active low) is used instead of the IN.

If you are wondering whether the PPI will be able to receive the data as fast as it can be sent by the CPU, rest your fears. All 8255 input and output can be done in less than 500 nanoseconds. Moreover, all of the 8255's inputs and outputs are TTL compatible.

But, the output drive capability of any port line is limited to one milliamp. This will not even light an LED. Consequently, if you plan to drive more than one TTL load you must provide the proper buffer and/or level shift.

The reset logic utilizes an XOR gate in a very unique way. TRS-80 reset (active low) is tied to one of the inputs of the gate while the other input is tied to an RC network consisting of C1 and R7 (see Fig. 1). This second input of the XOR is also connected to system ground through S2.

During normal operations, the system reset line is held high and the input from the RC network will normally be held high. Thus, the resulting output from the XOR gate will be low. The output of the XOR gate is then fed to pin 35 (reset active high) of the 8255, enabling normal operation of the chip.

This design allows the PPI-80 to be reset in several ways. First, by pressing the TRS-80 reset button, driving the system reset line low, you cause the XOR output to go high, resetting the 8255. The same condition will occur on power up of the TRS-80.

During power up the RC time constant, formed by C1 and R7, is sufficiently long to insure that the RC input to the XOR gate will remain low longer than the system reset. This causes the 8255 to reset. Thus, the PPI-80 will be reset whenever it is powered up. S2 can also be used to manually reset the PPI-80 by forcing the output of the RC network low.

The remaining part of the PPI-80 is the power supply. Refer to Fig. 1b. The construction is simple because of a 7805 IC. This is a single-package, three-terminal voltage regulator. It contains all the electronics to implement tight voltage regulation plus over voltage and thermal-shutdown protection. The remainder of the circuitry consists of a full-wave bridge rectifier (BR1) and filter capacitors (C2–C4).

Software Considerations

The primary purpose of a port is to communicate with the outside world. The 8255 can do this in one of its three modes.

Mode 0 represents simple I/O. That is, each of the ports in Fig. 1 can perform input or output, but not both. In this mode the data to be transmitted is latched to the ports.

In other words, if you write a byte to Port A, that bit pattern would stay on the pins PA0–PA7 until another was transmitted. When the TRS-80 reads port C, whatever is latched to pins PC0–PC7 is transferred to the TRS-80.

This type of I/O is used primarily for controlling devices that can be turned on or off. An input port can sense the condition of an on-off switch, and an output port can close or open the switch. This mode is the simplest of the three.

Mode 1 offers one important additional feature—handshaking. In this mode port A and port B can be assigned to input or output data. Port C provides the handshaking signals that synchronize the dialog between two machines. (The handshaking signals present on the 8255 in Modes 1 or 2 are given in Table 3.)

Suppose Port A is an output port. When writing data to an output port, previously transmitted information may not have been read by the external device, causing an overwrite. To prevent overwriting, the 8255 produces a standard handshaking signal, output buffer full (OBF) which is active low. This signal is produced on PC1 for

Resistors		
R1–R7	4700 ohm, ¼ watt, 5%	RS# 271-1330
Capacitors		
C1	100 uf, 25V electrolytic	RS# 272-1028
C2	1000 uf, 35V electrolytic	RS# 272-1032
C3	.1 uf, 50V ceramic disc	RS# 272-135
C4	10 uf, 16V tantalum cap	RS# 272-1071
Semiconductors		
BR1	1A, 50PIV full wave bridge	RS# 276-1161
U1	8255 PPI	Jameco Electronics
U2	74LS30 8-input NAND gate	RS# 276-1914
U3–U4	74LS86 Exclusive OR gate	Jameco Electronics
U5	7805 voltage regulator	RS# 276-1770
Miscellanous		
S1	6-position DIP switch	RS# 275-1301
S2	SPST momentary switch	RS# 275-1547
IC sockets	3 14-pin IC sockets	RS# 276-1999
	3 16-pin IC sockets	RS# 276-1998
	1 40-pin IC socket	RS# 276-1996
Transformer	1 modular wall plug type	RS# 60-3053
Cable	1 40 conductor cable	AP# 924065-36-R
Board	1 suitable perf board	RS# 276-152

NOTE: A drilled and etched board is available from Quant Systems, P.O. Box 628, Charleston, SC 29402.

Parts List.

✓Reader Service—see page 226

port B and PC7 for port A in Mode 1.

The user must know when the external device has received the byte sent from the TRS-80; the acknowledge (ACK), which is active low, provides this information. The logic for this signal must be available on the external device.

The remaining signal usually found on an input port is interrupt (INTR), which is active high. This signal goes high when the TRS-80 outputs a byte to the 8255 and remains in this state until the ACK is received from the external device.

When functioning in Mode 1 as an input port, the roles of the handshaking signals are reversed. The 8255 must be signaled when a device is ready to transmit information. This is done by the strobe (STB), which is active low. Upon receiving a strobe, the 8255 stores the byte and sets the input buffer full (IBF) at active high.

The IBF is monitored by the sending device to determine when it can transmit the next byte. As long as this signal remains high, it must wait; but once the TRS-80 reads a byte, this signal is driven low.

By now you must be wondering how the TRS-80 becomes aware of data waiting at the port. Polling techniques can be used for software while the active high interrupt request signal (INTR) is used for hardware. After data is received the INTR is set high (logic 1), but just like the IBF, the INTR is reset by the 8255 after a read.

The most sophisticated form of parallel I/O is bi-directional, which is provided by the 8255 in Mode 2. Bi-directional I/O is used with an intelligent external device, such as another computer. The 8255 permits port A to function as the data port, while port C provides all the handshaking signals discussed earlier: OBF, IBF, STB, ACK, and INTR. They function in exactly the same manner as in Mode 1 except they are bi-directional.

Using the Port

To read the contents of a port in BASIC you must use INP(port number). This statement is used like a function, such as ABS(X) or SQR(124). It must be used in conjunction with another statement.

In the following example

```
10 G = INP(127)
```

the variable G would be assigned the value of the contents of the port 127.

To send a value to the port use the statement OUT port number, value. Any value that is sent must be between 0 and 255. The OUT statement is a complete BASIC statement and requires only a line number to be included in a program. See line 70 in Fig. 2.

Four port addresses must be reserved. The two low-order address lines, A0 and A1, define which port is being addressed. The remaining address lines can be configured to suit your own needs, if they do not conflict with other port definitions.

	Mode 1				Mode 2	
Bit	INPUT Function		OUTPUT Function		INPUT/OUTPUT Function	
0	INTR	PORT B	INTR	PORT B	Unused	
1	IBF		$\overline{OBF}$		Unused	
2	$\overline{STB}$		$\overline{ACK}$		Unused	
3	INTR	PORT A	INTR	PORT A	INTR	PORT A
4	$\overline{STB}$		Unused		$\overline{STB}$	
5	IBF		Unused		IBF	
6	Unused		$\overline{ACK}$		$\overline{ACK}$	
7	Unused		$\overline{OBF}$		$\overline{OBF}$	

Depend on Port B definition

Table 3. Port C Functions in Modes 1 and 2.

For example, look at addresses 128, 129, 130, and 131. The decimal address 128 is equivalent to 10000000 binary. Notice that the two least significant bits are 0. Table 2 reveals that if the two low-order bits, corresponding to A0 and A1, are 0 then port A is selected. Ports B, C, and the control port are selected by 129, 130, and 131 respectively.

The 8255 is a programmable port. That is, before using it you must define the mode of operation for each of the ports. By sending the control port the appropriate message, you select the mode of operation for each port.

Table 4 contains control-word bit definitions for the three modes. This is the beauty of the 8255. With a simple software command you can configure the ports as you need to use them.

The software you develop will depend upon the port's application. In Mode 0 the principal operation will be sensing and operating switches. BASIC permits this to be done with relative ease with the INP and OUT instructions discussed earlier.

In Mode 1, however, handshaking signals are required.

Polling Handshake Signals

One technique for handling the handshaking signals is called polling. This requires the controlling program to periodically read the handshaking signals at port C and act on them. The INP command described earlier retrieves the status byte located at port C.

After the TRS-80 sends a byte to port A, it must not send another until the receiving device retrieves that byte. This can be monitored by scanning the OBF bit in port C. If the OBF bit for port A is low (logic 0), then the TRS-80 should not send a byte because the previous byte has not been read yet. A program can perform some other task and check the status of the bit periodically or can check it continuously.

When the external device receives the byte, it must return the ACK signal to the 8255 which drives the OBF high.

To use port B as an input port, the TRS-80 must know when the external device has sent a byte. This can be determined by checking the IBF signal associated with port B.

Since the IBF is an active high signal, the TRS-80 must scan port C until IBF goes high, which means port B has received a new byte from the external device. Reading the contents of port B will cause the 8255 to drive the IBF low. The sending device must have sufficient logic to send data only when IBF is low.

In Mode 2 operation, port A becomes a bi-directional I/O port. Handshaking signals on port C must be monitored in exactly the same fashion as Mode 1, checking the status of the IBF and OBF signals.

One of the most exciting possibilities of the port is the rapid exchange of information between two computers. The software required is very simple and even BASIC software can create transmission speeds so fast

Bit	Definition
0	Port C Lower PC0-PC3, used in Mode 0 or 1 1 = INPUT 2 = OUTPUT
1	PORT B Definition PB0-PB7, used in all modes 0 = OUTPUT, 1 = INPUT
2	Mode Select for Group B: PC0-PC3 and PB0-PB7 0 = MODE 0, 1 = MODE1
3	Port C Upper PC4-PC7, used in Mode 0 or 1 0 = OUTPUT, 1 = INPUT
4	Port A Definition PA0-PA7, used in Mode 0 and 1 0 = OUTPUT, 1 = INPUT
5-6	Mode select for Group A: PC4-PC7 and PA0-PA7 00 = Mode 0, 01 = Mode 1, 10 = Mode 2, 11 = Mode 2
7	Operation CODE 1 = Mode Set (write control word) 0 = Bit SET/RESET (used to generate clock signals)

Table 4. Control Port Bit Definition.

that the two machines appear to function as one.

Construction

The prototype circuit for this article (Fig. 1) was built on a Radio Shack board (part #276-152). However, since the operation of the circuit is not critical, a good perf board or vectorboard, with either point-to-point wiring or wire wrap should produce good results. It is also possible to purchase an etched and drilled circuit board from Quant Systems, Charleston, SC. IC sockets are recommended regardless of the construction method you use.

Begin assembly by installing the IC sockets for U1–U4 and S1, then solder the resistors in place. Next, mount C2 and wire it into the circuit as indicated in Fig. 1.

Choose connectors for the three 8-bit ports then mount them to the board and wire them to the peripheral interface pins of the 8255. Sixteen-pin IC sockets make convenient connectors for the three I/O ports.

In addition to the eight I/O lines, you may want two power supply lines to provide power to a satellite board. Depending on software definition, you may require as many as three handshaking signals.

Since 13 of the 16 available lines may be utilized, connection to an external device will require a 16-line jumper cable. This method of connection should always yield several spare lines for those oversights that always come up at the last minute.

The circuit requires a +5 volt power supply. While this is available on the TRS-80 bus, it is not recommended for the source because of its limited current capability. Build your own by following the schematic in Fig. 1b, or purchase a suitable plug-in modular supply. Use a +5 volt regulator chip as a safety feature for the ICs.

Switch S2 is used to reset the PPI-80 at the project board. If this capability is not desired, S2 may be omitted. The same reset function is accomplished at power up of either the TRS-80 or the PPI-80 and can also be accomplished by pushing the TRS-80 reset button.

Connecting the board to the TRS-80 is straightforward (see Fig. 1). First you need a 40-pin edge connector with cable. These cables were as scarce as dinosaurs, but now are available from several sources, including Hobby World, 19511 Business Center Drive, Northridge, CA 91324.

After making the connections to your board, double-check your work. The TRS-80 edge connector is numbered with even numbers on the bottom, starting at the edge nearest the center of the keyboard. Check the edge connector to the associated IC socket using an ohmmeter.

Testing

Determine if the PPI-80 is working properly by using one of the ports to send data to another. All you need to do is connect the corresponding port lines. (For example, connect PC0–PC7 to PA0–PA7.) Make sure the dip switch is set for port A, to be located at port 128, and enter Program Listing 1.

Note that line 70 writes the control word 137 to the 8255's control port at address 131. This specifies Mode 0 for all ports and defines port C as an input port and ports A and B as output ports.

The remainder of the program permits selection of a number between 0 and 255 to be sent to either port A or B and then to be read by port C. The number received by port C will be displayed and should be equal to the number entered.

Applications

Applications for the PPI-80 are numerous. It can be used to create real time computer games. It can be used for analog to digital conversion to monitor and control a home heating system. Connected to a Sears or Radio Shack remote home controller, it can switch lights and appliances on and off in an empty house. The authors are using their PPI-80 to construct a music synthesizer.

The possibilities are up to you. ■

WE WILL NOT BE UNDERSOLD

16K MEMORY UPGRADE KITS $49

for TRS-80*, Apple II, (specify): **Jumpers $2.50**

PRINTERS

NEC Spinwriter

Letter Quality High Speed Printer
Includes TRS-80* interface software, quick change print fonts, 55 cps, bidirectional, high resolution plotting, graphing, proportional spacing: R.O. **$2689**

R.O. with Tractor Feed **$2789** KSR with Tractor Feed **$3200**

779 CENTRONICS TRACTOR FEED PRINTER **$969**
Same as Radio Shack line printer I

737 CENTRONICS FRICTION & PIN FEED PRINTER **$839**
9 x 7 matrix

730 CENTRONICS FRICTION & PIN FEED PRINTER **$639**
7 x 7 matrix Same as Radio Shack line printer II

P1 CENTRONICS PRINTER **$269**
Same as Radio Shack quick printer

PAPER TIGER (IP440) **$939**
Includes 2K buffer and graphics option

TI-810 Faster than Radio Shack line printer III
Parallel and serial w/TRS-80* interface software **$1575**
with upper and lower case and paper tray **$1665**

OKIDATA Microline 80 Friction and pin feed **$559**
Tractor Feed, friction, and pin feed **$679**

EATON LRC 7000 + 64 columns, plain paper **$349**

ANADEX DP-9500 **$1389** DP-8000 **$869**

DISK OPERATING SYSTEMS

PATCHPAK #4 by Percom Data **$ 8.95**

CP/M for Model I, Zenith **$145** • for Model II, Altos **$169.00**

NEWDOS Plus — with over 200 modifications 35track **$ 89.00**
and corrections to TRS-DOS 40 or 70 Track **$ 99.00**

DISK DRIVES $314

More capacity than Radio Shack 35 Track (80 K Bytes) drives. Fully assembled and tested. Ready to plug-in and run the moment you receive it. Can be intermixed with each other and Radio Shack drive on same cable. TRS-80* compatible silver enclosure. **External card edge included.**

90 DAY WARRANTY. ONE YEAR ON POWER SUPPLY.

FOR TRS-80*

CCI-100	5¼", 40 Track (102K Bytes) for Model I	**$314**
CCI-200	5¼", 77 Track (197K Bytes) for Model I	**$549**
CCI-800	8" Drive for Model II (½ Meg Bytes)	**$795**
For Zenith Z89		
CCI-189	5¼", 40 Track (102K Bytes) add-on drive	**$394**
Z-87	Dual 5¼" add-on drive system	**$995**

DISKETTES — Box of 10 (5¼") - with plastic library case **$24.95**
8" double density for Model II (box of 10) **$36.49**

COMPLETE SYSTEMS

TRS-80* LEVEL II-16K with keypad **$709**
TRS-80* Expansion Interface **$269**
HEWLETT PACKARD HP-85 **$3199**
ZENITH Z89, 48K all-in-one computer **$2595**
ZENITH Z19 **$740**
TELEVIDEO **912B $745** **920B $769**
ATARI 400 $489 **ATARI 800 $799**
MATTEL INTELLIVISION **$249**
Software available for the above systems

CAT MODEM Originate and answer same as Radio Shack Telephone Interface II **$148**

LEEDEX MONITOR Video 100 **$119**

SOFTWARE FOR THE TRS-80* (Software w/Manual / Manual Alone)

CCI-INVESTMENT PORTFOLIO MANAGER: This is what investors have been waiting for! This powerful program was developed by security analysts working with software designers. It comes on one cassette — 16K LEVEL II BASIC on one side, 32K DISK BASIC on the other. Store and report data; Review your portfolio; Produce detailed status, value, gain, and security analysis; Compare alternatives **$49.95/$10**

INTELLIGENT TERMINAL SYSTEM ST-80-III BY LANCE MIKLUS: Enables a TRS-80* to act as a dial-up terminal on any standard time sharing network. Provides a TRS-80* with control key, ESC Key, Repeat Key, Rub Out Key, Break Key, full upper and lower case support, selectable printer output and program selectable transmission rates **$139/$10**

CCA-DATA MANAGEMENT SYSTEM: Automate your information processing tasks. You can create a file of customer information; quickly and easily add, delete or update records; search a file; keep a file in order of the value in any field; and print records and labels in any desired sequence or from just a part of a file. Requires 32K TRS-80 and one drive **$74.95/$10**

CSA-MAILIST SYSTEM: Creates, maintains and efficiently utilizes a name, address and telephone number file. 400 individual name/address entries can be maintained on a single density mini-floppy, and are manipulated directly by record number (direct access file method). Sorts can be performed, name + address combinations can be coded. Listing-directories and labels can be printed. A conversion facility is provided to convert most sequential name, address file formats into direct. Requires 32K TRS-80 and one drive. **$49.95/$10**

S & M SYSTEMS
INSEQ-80™*: Indexed Sequential Access Method (ISAM) for the TRS-80 Model I. A must for anyone writing business programs. Eliminate wasted disk space from direct record processing. Split second access to any record. Access data records instantly via alpha/numeric "key" eg. Part NR, zip code or sequentially in ascending key sequence. Add/modify records in any order. Access up to three files per program — Files may be spread over multiple disks. Machine language processing from your basic program. Utility program to convert direct files to INSEQ-80 format. **$49.95/$10**

FULLY INTERACTIVE ACCOUNTING PACKAGE: ISAM (INSEQ-80) based. Includes General Ledger, Accounts Payable, Accounts Receivable and Payroll. System runs "stand alone" or "co-ordinated G/L" at users option. Based on Osborne accounting method. Requires 32K, TRS-80, 2 or 3 drives. N/A CA

General Ledger	**$99/$10**
Accounts Receivable	**$99/$10**
Accounts Payable	**$99/$10**
Payroll	**$99/$10**

Osborne books: Req'd as additional documentation **$20 ea**

INVENTORY Requires 32K, TRS-80, 1 drive **$125/$10**

INSORT-80: Callable form BASIC via USR. Sorts "Random" Disk Files. "Disk" to "Disk" sort times — 350 records in 35 secs, 1000 records in 6 minutes, 3500 records in 12 minutes. Machine language processing. Up to 35 sort keys ascending/descending. Utility to build BASIC program. Runs under NEWDOS. **$49.95/$10**

ACCESSORIES

HEAD CLEANING DISKETTE: Cleans drive Read/Write head in 30 seconds. Diskette absorbs loose oxide particles, fingerprints, and other foreign particles that might hinder the performance of the drive head. Lasts at least 3 months with daily use. Specify 5¼" or 8". **$20 ea/$45 for 3**

FLOPPY SAVER: Protection for center holes of 5¼" floppy disks. Only 1 needed per diskette. Kit contains centering post, pressure tool, tough 7-mil mylar reinforcing rings. Installation tools and rings for 25 diskettes. **$11.95**
Re-orders of rings only: **$ 7.95**

CP/M BASED SOFTWARE for Zenith, Altos, Radio Shack, Apple (Software w/Manual / Manual Alone)

MICROSOFT
BASIC-80: Disk Extended BASIC ANSI compatible with long variable names, WHILE/WEND, chaining, variable length file records. **$325/$25**

BASIC COMPILER: Language compatible with BASIC-80 and 3-10 times faster execution. Produces standard Microsoft relocatable binary output. Includes MACRO-80. Also linkable to FORTRAN-80 or COBOL-80 code modules. **$350/$25**

FORTRAN-80: ANSI 66 (except for COMPLEX) plus many extensions. Includes relocatable object compiler, linking loader, library with manager. Also includes MACRO-80 (see below) **$425/$25**

COBAL-80: Level 1 ANSI '74 standard COBAL plus most of Level 2. Full sequential, relative and indexed file support with variable file names. STRING, UNSTRING, COMPUTE, VARYING/UNTIL, EXTEND, CALL, COPY, SEARCH, 3-dimensional arrays, compound and abbreviated conditions, nested IF. Powerful interactive screen-handling extensions. Includes compatible assembler, linking loader, and relocatable library manager as described under MACRO-80. **$700/$25**

Z-80 SOFTCARD FOR APPLE: Your key to future software expansion. Get the best of both worlds, Apple's 6502 and CP/M Z-80. Plug in the card and get a Z80. Supports Apple language card and all Apple peripherals. Comes with set of three manuals. **$339/$75**

CCI-TELNET VERSION 5: A communication Package which enables microcomputer users to communicate both with Large Mainframes and other microcomputers. Extensive commands make it useful in many applications where communication between computers is necessary. Powerful terminal mode enabling user to save all data from a session on disk. Completely CP/M compatible. Multiple communication protocols supported. Able to transfer files in both directions without protocol where the other machine does not support any protocol. Extensive ON-SCREEN help. Source code provided. **$149/$15**

MICROPRO-WORD-STAR: Menu driven visual word processing system for use with standard terminals. Text formatting performed on screen. Facilities for text paginate, page number, justify, center and underscore. User can print one document while simultaneously editing a second. Edit facilities include global search and replace. Read/Write to other text files, block move, etc. Requires CRT terminal with addressable cursor positioning. **$399/$40**

BDS 'C' COMPILER: Supports most features of language, including structures, arrays, pointers, recursive function evaluation, and overlays. Package contains: compiler, linker, library manager; sample source files include games, a terminal emulator with disk I/O plus the source for many standard library functions; BDS C User's Guide; Book - The C Programming Language by Dennis Ritchie and Brian Kernigham. Requires at least 24K of RAM **$125/$20**

CONFIGURABLE BUSINESS SYSTEM BY DMA: CBS is a data management system that allows true transaction processing. The system features a screen menu generator and a comprehensive report generator which can be used to produce invoices, purchase orders, re-order reports, mailing labels or other special reports specific to the application. Good documentation and a demonstration inventory system supplied. Requires at least 48K memory. Does not require any support language. **$295**

ACCESSORIES

EXTERNAL DATA SEPARATOR: Eliminates data separation problems(crc). Improves reliability. This plug-in unit comes fully assembled and tested. **$29.95**
RS 232 **$89.00**
DISK-DRIVE EXTENDER CABLES: Fits all mini-disk drives. **$15.95**

AC LINE CORD FILTER & 6 PRONG POWER STRIP **$38.00**
DISK DRIVE CABLES: 2 drive-**$29.00** 4 drive-**$35.00**
DUST COVERS: TRS-80/Apple **$ 7.95**
TRS-80 & OTHER MYSTERIES **$19.95**
RF MODULATOR: Adapts video to TV **$35.00**

DEALER (NATIONAL/INTERNATIONAL) INQUIRIES INVITED **Send for FREE Catalogue**

The CPU SHOP

✓298

TO ORDER CALL TOLL FREE 1-800-343-6522
Massachusetts residents call **(617) 242-3361**
For detailed technical information, call 617/242-3361
Hours: 10AM-6PM (EST) M-F (Sat. till 5)
*TRS-80 is a Tandy Corporation Trademark

5 Dexter Row, Dept. K9M
Charlestown, Massachusetts 02129

Massachusetts residents add 5% sales tax

Quantities on some items are limited

Children associating with a computer?

Kidstuff

Dan Keen
Dave Dischart
RD 1, Box 432
State Highway 83
Cape May Ct. House, NJ 08210

Small kids can be hard on things. But with proper supervision, a microcomputer can be a powerful teaching aid.

This program, for example, will help your preschooler associate objects with sounds. It's a crucial skill, if he is to learn how to read. The program also makes the child pick out letter shapes on the keyboard.

He'll have fun, too; many of the objects are animated. A car travels down the highway, an airplane takes off, a train chugs down the tracks, and a dragon breathes fire.

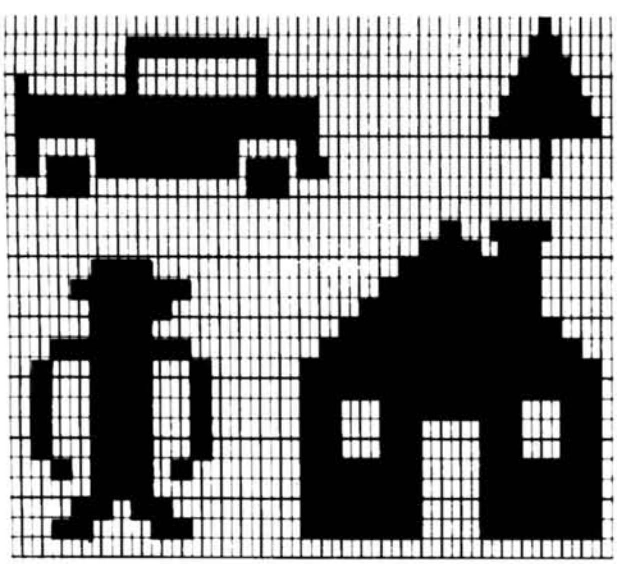


Fig. 1. Sample Shapes

Kid Testing

We designed the program for kindergarten children. But "kid testing" proved that even 2- and 3-year-olds like it. They may not know the letters, but with parental supervision they quickly learn to associate the shapes with names. In fact, they soon learn the sequence and will frequently tell you what object will be displayed before it is even drawn.

Since most children in this age bracket can't read, the computer has trouble communicating with them. An adult must show the child how to respond.

Once into the program, operation is simple. The program uses INKEY$ so that the child never has to hit the enter key. When he sees a flashing question mark, he types the letter that the object displayed begins with. If he is right, the word RIGHT and a smiling face appear.

If he's not, the screen prints WRONG—TRY AGAIN and draws a sad face. The computer then waits for him to type another letter. He can make any number of guesses.

Before an object is shown, a dummy INKEY$ picks up any

OBJECT	ACCEPTABLE LETTERS
plane	Airplane, Plane, Jet
boat	Boat, Choo-choo
heart	Heart, Valentine
tree	Tree, Christmas tree
cowboy	Cowboy, Bandit, Man, Outlaw
house	House, Building
car	Car, Automobile
castle	Castle, Building
Santa	Santa
dragon	Dragon, Monster
steps	Stairs
clock	Clock, Watch
letter	Letter, Mail, Postcard
toothbrush	Toothbrush, Brush

Table 1. Objects and Their Answers

stray keyboard input. Thus, if the child accidentally leans on a key during animation, the letter won't register as a bad guess when the next object appears.

Most objects have several acceptable answers. Table 1 gives a list of subjects and answers.

Consonants are emphasized, since beginning vowel sounds are more difficult. We didn't try to check every letter in the alphabet.

You'll need 16K Level II or 32K Disk BASIC to run the program. You must put the *exact* number of Xs inside the quotes on strings. Otherwise, the entire program will bomb. Also be sure to include the second question mark.

One Caution

Because of its string packing routines, this program will not run with Apparat's NEWDOS + unless you put all the strings at the beginning of the program. Otherwise, all the string locations in later lines would be out in the interface RAM, causing VARPTR to return an error.

Don't be alarmed at the weird screen scrolling when you list after running the program. This is caused by the string packing routines. ■

```
5 CLEAR1000.S=0:GOSUB50000
100 A1$="XXXXX"
101 RESTORE:K=VARPTR(A1$):AD=PEEK(K+2)*256+PEEK(K+1)
102 DATA138,140,140,157,132
103 FORP=0TO4:READX:POKEAD+P,X:NEXTP
104 A2$="XXXXXXXXXXXX"
105 K=VARPTR(A2$):AD=PEEK(K+2)*256+PEEK(K+1)
106 DATA128,144,128,160,26,8,8,8,131,131,139,131
107 FORP=0TO11:READX.POKEAD+P,X:NEXTP
108 A3$="XXXXXXXXXX"
109 K=VARPTR(A3$):AD=PEEK(K+2)*256+PEEK(K+1)
110 DATA180,176,184,176,26,8,8,8,128,130
111 FORP=0TO9:READX:POKEAD+P,X:NEXTP
112 A0$="XXXXX"
113 K=VARPTR(A0$):AD=PEEK(K+2)*256+PEEK(K+1)
114 DATA128,128,128,128,128
120 CLS.FORP=0TO4:READX:POKEAD+P,X:NEXT:FORL=896TO915:PRINT@L,A3$;:GOSUB190:NEXT
130 FORL=915TO264STEP-60:PRINT@L-1,A0$;:PRINT@L,A3$;:GOSUB190:PRINT@L,CHR$(30);:PRINT@L,A1$;:GOSUB190:PRINT@L,CHR$(30);:PRINT@L-64,A
2$;:GOSUB190:NEXTL
150 CLS:PRINT@400,A1$:GOSUB60000:IF(A$="P")OR(A$="A")OR(A$="J")THENGOSUB62000:GOTO200
160 GOSUB61000:GOSUB60000:GOTO150
190 FORQ=1TO10:NEXT:PRINT@L-2,"     ";:PRINT@L+64,"     ";:PRINT@L-64,"     ":RETURN
200 F1$="XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"
202 K=VARPTR(F1$):AD=PEEK(K+2)*256+PEEK(K+1)
204 DATA176,176,176,176,176,128,26,24,24,24,24,24,24,24,24,184,191,179,191,191,191,191,189,184,149,128,26,24,24,24,24,24,24,24,24,24
,24,24,130,143,185,191,191,191,191,135,130,133,128
206 FORP=0TO48.READX.POKEAD+P,X.NEXTP
210 F2$="XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"
212 K=VARPTR(F2$):AD=PEEK(K+2)*256+PEEK(K+1)
214 DATA176,176,176,176,176,128,26,24,24,24,24,24,24,24,24,184,191,179,191,191,191,191,189,184,134,128,26,24,24,24,24,24,24,24,24,24
,24,24,130,143,185,191,191,191,191,135,130,137,128
216 FORP=0TO48.READX.POKEAD+P,X:NEXTP
220 F3$="XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"
222 K=VARPTR(F3$):AD=PEEK(K+2)*256+PEEK(K+1)
224 DATA176,176,176,176,176,128,26,24,24,24,24,24,24,24,24,184,191,179,191,191,191,191,189,176,140,128,26,24,24,24,24,24,24,24,24,24
,24,24,130,143,185,191,191,191,191,135,128,131,128
226 FORP=0TO48:READX:POKEAD+P,X:NEXTP
230 F4$="XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"
232 K=VARPTR(F4$):AD=PEEK(K+2)*256+PEEK(K+1)
234 DATA176,176,176,176,176,128,26,24,24,24,24,24,24,24,24,184,191,191,191,191,191,191,189,184,149,128,26,24,24,24,24,24,24,24,24,24
,24,24,130,143,185,191,191,191,191,135,130,133,128
236 FORP=0TO48:READX:POKEAD+P,X:NEXTP
255 BO$="XXXXXXXXXXX"
260 K=VARPTR(BO$):AD=PEEK(K+2)*256+PEEK(K+1)
265 DATA128,176,176,188,188,191,188,188,176,176,176
270 FORP=0TO10:READX:POKEAD+P,X:NEXTP:CLS.PO=512
275 PRINT@PO,BO$;:FORDE=1TO20:NEXTDE:PO=PO+1:IFPO<562THEN275
280 CLS:PRINT@458,BO$:GOSUB60000:IF(A$="B")OR(A$="S")THENGOSUB62000:GOTO300
285 GOSUB61000:GOSUB60000:GOTO280
300 TR$="XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"
305 K=VARPTR(TR$):AD=PEEK(K+2)*256+PEEK(K+1)
310 DATA128,188,188,140,140,148,160,171,151,26,24,24,24,24,24,24,24,24,24,24,128,160,191,191,191,191,191,191,191,191,144,26,24,24,24
```

Program continues

```
,24,24,24,24,24,24,24,128,130,135,128,128,128,128,130,135
315 FORPO=0TO50:READX:POKEAD+PO,X:NEXTPO
320 CLS:LO=450:FORPO=640TO703STEP2:PRINT@PO,CHR$(151);:PRINT@PO+1,CHR$(131);:NEXT:PRINT@704,STRING$(64,131);
325 PRINT@LO,TR$;:GOSUB375
330 PRINT@LO,TR$;:PRINT@LO-57,"*";:GOSUB375
335 PRINT@LO,TR$;:PRINT@LO-58," ";:PRINT@LO-123,"*";:GOSUB375
340 PRINT@LO,TR$;:PRINT@LO-124," ";:GOSUB375
345 IFLO>500THEN400ELSEGOTO325
375 FORDE=0TO10:NEXTDE:LO=LO+1:RETURN
400 CLS:PRINT@400,TR$:GOSUB60000
410 IFA$="T"ORA$="C"THENGOSUB62000:GOTO1000
420 GOSUB61000:GOSUB60000:GOTO410
1000 CLS:LO=502:SW=0
1010 PRINT@LO,F1$;:GOSUB1100
1020 PRINT@LO,F2$;:GOSUB1100
1030 PRINT@LO,F3$;:GOSUB1100
1035 IFLO<452THEN1500
1040 SW=SW+1:IFSW<2THEN1010ELSEPRINT@LO,F4$;:GOSUB1100:SW=0:GOTO1020
1100 FORQ=1TO15:NEXTQ:LO=LO-1:RETURN
1500 CLS:PRINT@400,F1$;:GOSUB60000
1600 IFA$="F"THENGOSUB62000:GOTO2000
1700 GOSUB61000:GOSUB60000:GOTO1600
2000 HE$="XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"
2010 K=VARPTR(HE$):AD=PEEK(K+2)*256+PEEK(K+1)
2020 DATA168,131,131,137,152,131,131,169,26,24,24,24,24,24,24,24,137,144,128,128,128,152,129,26,24,24,24,24,24,24,130,164,160,134,26
,24,24,129
2030 FORP=0TO37:READX:POKEAD+P,X:NEXTP
2040 CLS:PRINT@400,HE$:GOSUB60000
2050 IF(A$="H")OR(A$="V")THENGOSUB62000:GOTO3040
2060 GOSUB61000:GOSUB60000:GOTO2050
3040 TE$="XXXXXXXXXXXXXXXX"
3050 K=VARPTR(TE$):AD=PEEK(K+2)*256+PEEK(K+1)
3060 DATA128,160,190,180,26,24,24,24,24,24,128,136,143,175,143,141
3070 FORP=0TO15:READX:POKEAD+P,X:NEXT
3080 CLS:PRINT@400,TE$:GOSUB60000
3090 IF(A$="C")OR(A$="T")THENGOSUB62000:GOTO3100
3095 GOSUB61000:GOSUB60000:GOTO3090
3100 CO$="XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"
3110 K=VARPTR(CO$):AD=PEEK(K+2)*256+PEEK(K+1)
3120 DATA176,188,188,188,176,176,26,24,24,24,24,24,24,24,176,176,191,191,191,179,176,26,24,24,24,24,24,24,24,24,191,32,32,191,191,19
1,32,32,191
3130 DATA26,24,24,24,24,24,24,24,24,24,143,176,32,191,191,191,32,176,143,26,24,24,24,24,24,24,24,24,24,176,188,143,131,131,131,143,1
88,176
3140 FORP=0TO76:READX:POKEAD+P,X:NEXTP
3174 CLS:PRINT@400,CO$:GOSUB60000
3176 IF(A$="C")OR(A$="M")OR(A$="B")OR(A$="O")THENGOSUB62000:GOTO4000
3180 GOSUB61000:GOSUB60000:GOTO3176
4000 HH$="XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXX"
4010 K=VARPTR(HH$):AD=PEEK(K+2)*256+PEEK(K+1)
4020 DATA128,176,128,160,176,176,144,26,24,24,24,24,24,24,24,24,24,24,128,176,188,191,191,191,188,191,191,26,24,24,24,24,24,24,24,24
,24,24,24,24,128,176,188,191,191,191,191,191,191,191,191,191,188,176,26,24,24,24,24,24,24,24,24,24,24,24,24,24,24
4030 DATA128,191,191,143,143,191,191,191,191,191,191,191,143,143,191,191,26,24,24,24,24,24,24,24,24,24,24,24,24,24,24,24,128
4035 DATA191,191,176,176,191,191,131,131,131,191,191,176,176,191,191,26,24,24,24,24,24,24,24,24,24,24,24,24,24,24,24,24,128,191,191,
191,191,191,191,128,128,128,191,191,191,191,191,191
4050 FORP=0TO151:READX:POKEAD+P,X:NEXT
4060 CLS:PRINT@336,HH$:GOSUB60000
4070 IF(A$="B")OR(A$="H")THENGOSUB62000:GOTO4100
4080 GOSUB61000:GOSUB60000:GOTO4070
4100 CC$="XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"
4110 K=VARPTR(CC$):AD=PEEK(K+2)*256+PEEK(K+1)
4120 DATA156,140,140,140,140,140,172,26,24,24,24,24,24,24,24,24,24,24,24,24,170,188,188,188,188,189,188,188,188,188,188,190,188,188,
188,140,26
4130 DATA24,24,24,24,24,24,24,24,24,24,24,24,24,24,24,24,130,133,188,188,130,143,143,143,143,143,143,133,188,188,130,141
4140 FORP=0TO68:READX:POKEAD+P,X:NEXT
```

Program continues

✓Reader Service—see page 226

```
4150 CLS:PRINT@400,CC$:GOSUB60000
4160 IF(A$="A")OR(A$="C")THENGOSUB62000:GOTO5000
4170 GOSUB61000:GOSUB60000:GOTO4160
5000 CLS:PRINT@704,STRING$(64,131)
5100 PO=455:SW=0
5110 PRINT@695,"*";:PRINT@635," ";:GOTO5130
5120 PRINT@695," ";:PRINT@635,"*";
5130 PRINT@PO,TE$;:PRINT@554,CC$;:PO=PO+1
5140 IFPO>503THEN5160
5150 SW=SW+1:IFSW>1THENSW=0:GOTO5110ELSE5120
5160 PRINT@500,CHR$(30);:PRINT@564,CHR$(30);:PRINT@628,CHR$(30);:PRINT@692,CHR$(30);
5200 PO=330:SW=0
5210 PRINT@695,"*";:PRINT@635," ";:GOTO5230
5220 PRINT@695," ";:PRINT@635,"*";
5230 PRINT@PO,HH$;:PRINT@554,CC$;:PO=PO+1
5240 IFPO>370THEN5300
5250 SW=SW+1:IFSW>1THENSW=0:GOTO5210ELSE5220
5300 CLS:FORX=88TO127:FORY=21TO47:SET(X,Y):NEXTY,X
5302 FORX=90TO91:FORY=20TO22:RESET(X,Y):NEXTY,X
5304 FORX=94TO95:FORY=21TO22:RESET(X,Y):NEXTY,X
5306 FORX=98TO117:FORY=21TO28:RESET(X,Y):NEXTY,X
5308 FORX=120TO121:FORY=21TO22:RESET(X,Y):NEXTY,X
5310 FORX=124TO125:FORY=21TO22:RESET(X,Y):NEXTY,X
5312 FORX=98TO99:FORY=34TO35:RESET(X,Y):NEXTY,X
5314 FORX=116TO117:FORY=39TO40:RESET(X,Y):NEXTY,X
5316 FORX=118TO122:FORY=15TO16:SET(X,Y):NEXTY,X
5318 FORY=17TO20:SET(122,Y):NEXT
5330 S=1:GOSUB60000:IF(A$="C")OR(A$="B")OR(A$="F")THENGOSUB62000:GOTO5400
5340 GOSUB61000:GOSUB60000:GOTO5330
5400 CLS:SET(50,0):SET(51,0):FORX=46TO49:SET(X,1):NEXT:SET(52,1):SET(53,1):SET(44,2):SET(45,2):SET(54,2):SET(55,2):FORX=41TO43:SET(X
,3):NEXT
5410 SET(56,3):FORX=37TO40:SET(X,4):NEXT:SET(56,4):SET(57,4)
5420 FORX=34TO36:SET(X,5):NEXT:SET(58,5):SET(59,5):FORX=31TO33:SET(X,6):NEXT
5430 SET(60,6):SET(61,6):FORX=28TO30:SET(X,7):NEXT:SET(62,7):SET(63,7):FORX=22TO24:SET(X,8):NEXT:SET(26,8):SET(27,8)
5440 FORX=60TO66:SET(X,8):NEXT:SET(21,9):FORX=25TO42:SET(X,9):NEXT:FORX=52TO59:SET(X,9):NEXT:FORX=67TO71:SET(X,9):NEXT
5450 SET(20,10):SET(27,10):FORY=10TO14:SET(38,Y):NEXT:FORX=49TO51:SET(X,10):NEXT:SET(55,10):SET(72,10):SET(73,10):SET(19,11):SET(20,
11):SET(26,11)
5460 FORX=46TO48:SET(X,11):NEXT:SET(74,11):FORX=20TO25:SET(X,12):NEXT:FORX=42TO45:SET(X,12):NEXT:FORX=63TO65:SET(X,12):NEXT
5470 SET(72,12):SET(74,12):SET(22,13):SET(41,13):FORX=55TO57:SET(X,13):NEXT:SET(60,13):SET(62,13):FORX=65TO67:SET(X,13):NEXT
5480 SET(71,13):SET(73,13):FORX=37TO40:SET(X,14):NEXT:SET(47,14):SET(53,14):SET(54,14):FORX=57TO59:SET(X,14):NEXT:SET(61,14)
5490 FORX=68TO70:SET(X,14):NEXT:SET(72,14):SET(37,15):SET(39,15):SET(46,15):SET(48,15):SET(49,15):SET(51,15):SET(52,15):SET(61,15):S
ET(72,15):SET(73,15)
5500 SET(38,16):SET(45,16):SET(50,16):SET(66,16):SET(67,16):SET(74,16):SET(75,16):SET(82,12):SET(81,13):SET(81,14):FORY=13TO16:SET(8
3,Y):NEXT
5510 SET(79,15):SET(80,15):SET(78,16):FORX=35TO37:SET(X,17):NEXT:SET(44,17):SET(64,17):SET(65,17):FORX=49TO51:SET(X,17):NEXT:FORX=68
TO70:SET(X,17):NEXT:SET(76,17):SET(77,17)
5520 SET(82,17):SET(42,18):SET(43,18):SET(54,18):FORX=71TO75:SET(X,18):NEXT:SET(80,18):SET(81,18):SET(36,19):SET(37,19):SET(41,19):F
ORX=45TO48:SET(X,19):NEXT:SET(54,19)
5530 FORX=57TO60:SET(X,19):NEXT:FORX=63TO70:SET(X,19):NEXT:SET(78,19):SET(79,19):SET(36,20):SET(40,20):FORX=44TO49:SET(X,20):NEXT:SE
T(53,20):SET(56,20):SET(61,20):SET(62,20)
5540 FORX=75TO77:SET(X,20):NEXT:SET(80,20):FORX=38TO40:SET(X,21):NEXT:SET(56,21):SET(60,21):SET(61,21):FORX=72TO74:SET(X,21):NEXT:SE
T(81,21):SET(33,22):SET(41,22)
5550 FORX=53TO55:SET(X,22):NEXT:FORX=57TO59:SET(X,22):NEXT:FORX=63TO70:SET(X,22):NEXT:SET(82,22):SET(34,23):SET(35,23):SET(42,23):FO
RX=50TO52:SET(X,23):NEXT:SET(66,23):SET(68,23)
5560 SET(82,23):SET(34,24):FORX=36TO39:SET(X,24):NEXT:SET(42,24):FORX=45TO49:SET(X,24):NEXT:FORX=61TO65:SET(X,24):NEXT:SET(68,24):FO
RY=24TO26:SET(83,Y):NEXT
5570 SET(35,25):SET(36,25):FORX=40TO43:SET(X,25):NEXT:SET(57,25):SET(58,25):SET(60,25):SET(66,25):SET(67,25):FORX=37TO41:SET(X,26):N
EXT
5580 FORX=46TO56:SET(X,26):NEXT:FORX=61TO65:SET(X,26):NEXT:FORX=42TO45:SET(X,27):NEXT:FORY=27TO33:SET(84,Y):NEXT:SET(43,28):SET(44,2
9):SET(45,30):SET(46,31):SET(47,31)
5590 SET(47,32):SET(48,32):SET(82,33):FORY=34TO42:SET(80,Y):NEXT:SET(82,36):SET(81,35):FORY=34TO36:SET(83,Y):NEXT:SET(81,38):SET(52,
31):SET(51,32):SET(52,32):FORX=49TO51:SET(X,33):NEXT:SET(53,33)
5594 SET(51,34):SET(54,34):SET(55,35):SET(56,36):SET(57,37):SET(58,37):FORX=59TO61:SET(X,38):NEXT:FORX=62TO64:SET(X,39):NEXT:FORX=64
TO67:SET(X,40):NEXT
```

Program continues

```
5596 FORX=68TO71.SET(X,41).NEXT.FORX=72TO75.SET(X,42).NEXT.FORX=76TO80.SET(X,43).NEXT.SET(81,44)
5600 GOSUB60000.IFA$="S"THENGOSUB62000.GOTO5800
5700 GOSUB61000.GOTO5400
5800 MG$=CHR$(188)+CHR$(191)+CHR$(189)+CHR$(188)+STRING$(2,140)+CHR$(26)+STRING$(8,24)
5810 MG$=MG$+CHR$(176)+STRING$(3,191)+CHR$(135)+STRING$(2,131)+CHR$(26)+STRING$(12,24)
5820 MG$=MG$+CHR$(160)+CHR$(184)+CHR$(188)+STRING$(4,191)+CHR$(159)+CHR$(129)+CHR$(26)+STRING$(13,24)
5830 MG$=MG$+CHR$(176)+CHR$(188)+CHR$(190)+STRING$(7,191)+CHR$(135)+CHR$(26)+STRING$(13,24)
5840 MG$=MG$+CHR$(176)+CHR$(190)+STRING$(10,191)+STRING$(3,188)+CHR$(132)+CHR$(26)+STRING$(19,24)
5850 MG$=MG$+CHR$(176)+CHR$(188)+STRING$(10,191)+CHR$(135)+STRING$(5,131)+CHR$(129)+CHR$(26)+STRING$(22,24)
5860 MG$=MG$+CHR$(176)+CHR$(188)+STRING$(11,191)+CHR$(159)+CHR$(129)+CHR$(26)+STRING$(18,24)
5870 MG$=MG$+CHR$(176)+CHR$(188)+STRING$(13,191)+CHR$(135)
5880 M1$=CHR$(160)+CHR$(184)+CHR$(190)+STRING$(6,143)+CHR$(191)+CHR$(159)+STRING$(5,143)+CHR$(191)+CHR$(159)+CHR$(133)+CHR$(26)+STRI
NG$(25,24)
5890 M1$=M1$+STRING$(3,176)+CHR$(184)+CHR$(190)+CHR$(143)+CHR$(131)+"        "+CHR$(191)+CHR$(181)+"     "+CHR$(191)+CHR$(181)
5900 CLS:PRINT@416,MG$;:PRINT@903,M1$;
5910 RESET(67,19):X=422
5920 A$=INKEY$:IFA$="D"ORA$="M"THENGOSUB62000:GOTO6000
5925 IFA$<>""THENGOSUB61000:GOTO5900
5930 PRINT@448,"?";:PRINT@X,"*";:FORDE=1TO10:NEXT:PRINT@448," ";:PRINT@X," ";:FORDE=1TO10:NEXT
5940 X=X+1
5950 IFX>434THENX=422
5960 GOTO5920
6000 CLS:X=86:FORQ=XTO99:SET(Q,20):SET(Q,21):NEXT
6010 X=X-4:FORQ=XTO99:SET(Q,22):SET(Q,23):NEXT
6020 X=X-4:FORQ=XTO99:SET(Q,24):SET(Q,25):NEXT
6030 X=X-4:FORQ=XTO99:SET(Q,26):SET(Q,27):NEXT
6035 X=X-4:FORQ=XTO99:SET(Q,28):SET(Q,29):NEXT
6040 GOSUB60000:IFA$="S"THENGOSUB62000:GOTO6100
6050 GOSUB61000:GOTO6040
6100 CLS:PRINT@165,"1";:PRINT@232,"2";:PRINT@361,"3";:PRINT@488,"4";:PRINT@550,"5";:PRINT@606,"6";
6110 PRINT@535,"7";:PRINT@468,"8";:PRINT@339,"9";:PRINT@212,"10";:PRINT@151,"11";:PRINT@94,"12";:
6120 FORX=61TO75:SET(X,16):NEXT:FORY=6TO15:SET(61,Y):NEXT
6130 SET(60,7):SET(59,8):SET(62,7):SET(63,8):SET(73,14):SET(74,15):SET(74,17):SET(73,18)
6140 FORX=36TO86:SET(X,2):SET(X,30):NEXT:FORY=30TO2STEP-1:SET(34,Y):SET(35,Y):SET(86,Y):SET(87,Y):NEXT
6150 GOSUB60000:IFA$="C"ORA$="W"THENGOSUB62000:GOTO6200
6160 GOSUB61000:GOTO6150
6200 CLS:FORX=48TO127:SET(X,11):SET(X,33):NEXT:FORY=33TO11STEP-1:SET(48,Y):SET(49,Y):SET(126,Y):SET(127,Y):NEXT
6210 FORX=115TO122:SET(X,13):SET(X,18):NEXT:FORY=14TO17:SET(115,Y):SET(122,Y):NEXT
6220 PRINT@378,".15";:PRINT@282,"CLARK KENT";:PRINT@346,"DAILY PLANET";:PRINT@410,"METROPOLIS";:PRINT@487,"SANTA CLAUS";:PRINT@551,"
RD1 BOX1";:PRINT@615,"NORTH POLE";
6230 GOSUB60000:IFA$="P"ORA$="L"ORA$="M"THENGOSUB62000:GOTO6300
6240 GOSUB61000:FORX=48TO127:SET(X,33):NEXT:GOTO6230
6300 CLS.V$=STRING$(16,176)+STRING$(5,186)+CHR$(144):PRINT@460,V$
6310 GOSUB60000:IFA$="B"ORA$="T"THENGOSUB62000:GOTO6400
6320 GOSUB61000:GOTO6310
6400 END: ' DELETE THIS LINE IF YOU WANT AN ENDLESS LOOP
6410 GOTO100
50000 DEFINTA-Z:CLS.PRINTCHR$(23):PRINT"          MOM & DAD:"
50010 PRINT:PRINT"INSTRUCT CHILD TO HIT THE":PRINT"APPROPRIATE KEY. IF A"
50020 PRINT"TRAIN APPEARS HE SHOULD HIT":PRINT"THE LETTER  T  WHEN":PRINT"A QUESTION MARK FLASHES."
50030 PRINT.PRINT"HIT ENTER WHEN CHILD IS READY."
50040 A$=INKEY$:IFA$=""THEN50040ELSECLS:RETURN
54000 BREAK
60000 DUMMY$=INKEY$
60005 A$=INKEY$:PRINT@448,"?";:FORX=1TO100:NEXT
60010 PRINT@448," ";.FORX=1TO100.NEXT.IFA$=""THEN60005ELSERETURN
61000 SET(16,41).SET(17,40):SET(26,40):SET(27,41).FORX=18TO25:SET(X,39).NEXTX
61005 SET(21,37):SET(17,35):SET(18,35):SET(24,35).SET(25,35):PRINT@904,"WRONG!";:PRINT@967,"TRY AGAIN";
61008 IFS=1THENS=0:FORX=0TO30.FORY=35TO42:RESET(X,Y).NEXTY,X:RETURN
61010 FORDE=0TO800.NEXT.PRINT@704,CHR$(30):PRINT@768,CHR$(30).PRINT@832,CHR$(30):PRINT@896,CHR$(30).PRINT@960,CHR$(30);
61020 RETURN
62000 FORX=18TO25.SET(X,40).NEXT.SET(17,39):SET(16,38).SET(26,39).SET(27,38)
62005 SET(21,37).SET(17,35).SET(18,35).SET(25,35):SET(26,35)
62010 PRINT@904,"RIGHT!",.FORDE=1TO800.NEXT:PRINT@704,CHR$(30):PRINT@768,CHR$(30):PRINT@832,CHR$(30).PRINT@896,CHR$(30),
62020 RETURN
```

✓Reader Service—see page 226

H & E COMPUTRONICS INC.

•••EVERYTHING FOR YOUR TRS-80™•••

TRS-80 is a trademark of the Radio Shack Division of Tandy Corporation

COORDINATED BUSINESS SYSTEMS

SMALL BUSINESS SYSTEMS GROUP

- EACH MODULE CAN BE OPERATED INDIVIDUALLY OR AS A COORDINATED SYSTEM.
- TURN-KEY ERROR CATCHING OPERATION FOR BEGINNERS
- FREE 30-DAY TELEPHONE CONSULTATION WITH SBSG
- EACH MODULE CAN BE FORMATTED TO SPAN DATA ON UPTO 4 DISK DRIVES.
- COMPLETE MANUAL AND DOCUMENTATION ACCOMPANY EACH MANUAL
- MINIMUM SYSTEM REQUIREMENTS - 2 DISK DRIVES FOR MODEL I...1-DISK DRIVE FOR MODEL II

ACCOUNTS PAYABLE

The accounts payable system receives data concerning purchases from suppliers and produces checks in payment of outstanding invoices. In addition, it produces cash management reports. This system aids in tight financial control over all cash disbursements of the business. Several reports are available and supply information needed for the analysis of payments, expenses, purchases and cash requirements. All A/P data feeds General Ledger so that data is entered into the system just once. These programs were developed 5 years ago for the Wang micro-computer and have been tested in many environments since then. The package has been converted to the TRS-80™ and is now a well documented, on-line, interactive micro-computer system with the capabilities of (or exceeding many larger systems.

CAPABILITIES

★ menu driven; easy to use; full screen prompting and cursor control
★ invoice oriented; everything revolves around the invoice; handles new invoice or credit memo or debit memo
★ invoice information recorded; invoice #, description, buyer, check register #, invoice date, age date, amount of invoice, discount (in %), freight, tax ($), total payable
★ transaction print and file maintenance procedures insure accuracy
★ flexible check calculation procedure; allows checks to be calculated for a set of vendors - or - for specific vendors
★ program prints your checks; contiguous computer checks with your company letterhead can be purchased from SBSG
★ reports include (samples on back):
  - open item listing/closed item listing - both detail and summary
  - debit memo listing/credit memo listing
  - aging
  - check register report (to give an audit trail of checks printed)
  - vendor listing and vendor activity (activity of the whole year)
★ fully linked to GENERAL LEDGER; each invoice can be distributed to as many as five (5) different GL accounts; sysem automatically posts to cash and A/P accounts

ACCOUNTS RECEIVABLE

The objective of a computerized A/R system is to prepare accurate and timeley monthly statements to credit customers. Management can generate information required to control the amount of credit extended and the collection of money owed in order to maximize profitable credit sales while minimizing losses from bad debts. The programs composing this system were developed 5 years ago, especially for small businesses using the Wang Microcomputer. They have been tested in many environments since then. Each module can be used stand alone or can feed General Ledger for a fully integrated system.

CAPABILITIES

★ menu driven; easy to use; full screen prompting and cursor control
★ invoice oriented; invoices can be entered before ready for billing, when ready for billing, after billing or after paid
★ allows entry of new invoice, credit memo, debit memo, or change/delete invoice
★ allows for progress payment
★ transaction information includes:
  - type of A/R transaction
  - customer P.O. #
  - description of P.O.
  - billing date
  - general ledger account number
  - invoice amount
  - shipping/transportation charges
  - tax charges
  - payment
  - progress payment information
  - transaction print and file maintenance procedures insure accuracy
★ customer statements printed; computer statements with your company letterhead can be purchased from SBSG
★ reports include; (samples on back)
  - listing of invoices not yet billed
  - open items (unpaid invoices)
  - closed items (paid invoices)
  - aging
★ fully linked to General Ledger; will post to applicable accounts: debits A/R, credits account you specify

PAYROLL

Payroll invoices many complex calculations and the production of reports and documents, many of which are required by government agencies. It is an ideal candidate for the computer. With this Payroll system in-house, you can promptly and accurately pay your employees and generate accruate documents/reports to management, employees, and appropriate government agencies concerning earnings, taxes, and other deductions. The package has been converted to the TRS-80™ and is now a well documented, on-line, interactive micro-computer system with the capabilities of (or exceeding) many larger systems.

CAPABILITIES:

★ performs all necessary payroll tasks including:
  - file maintenance, pay data entry and verification
  - computation of pay and deduction amounts
  - printing of reports and checks
★ can handle salaried and hourly employees
★ employees can receive:
  - hourly or salary wage
  - vacation pay
  - holiday pay
  - piecework pay
  - overtime pay

(Continued on next page)

✓Reader Service—see page 226

H & E COMPUTRONICS INC.

•••EVERYTHING FOR YOUR TRS-80™•••

TRS-80 is a trademark of the Radio Shack Division of Tandy Corporation

MICROSOFT BASIC COMPILER

With TRS-80™ BASIC Compiler, your Level II programs will run at record speeds! Compiled programs execute an average of 3-10 times faster than programs run under Level II. Make extensive use of integer operations, and get speeds 20-30 times faster than the interpreter.

Best of all, BASIC Compiler does it with BASIC, the language you already know. By compiling the same source code that your current BASIC interprets, BASIC Compiler adds speed with a minimum of effort.

And you get more BASIC features to program with, since features of Microsoft's Version 5.0 BASIC interpreter are included in the package. Features like the WHILE...WEND statement, long variable names, variable length records, and the CALL statement make programming easier. An exclusive BASIC Compiler feature lets you call FORTRAN and machine language subroutines much more easily than in Level II.

Simply type in and debug your program as usual, using the BASIC interpreter. Then enter a command line telling the computer what to compile and what options to use.

Voila! Highly optimized, Z-80 machine code that your computer executes in a flash! Run it now or save it for later. Your compiled program can be saved on disk for direct execution every time.

Want to market your programs? Compiled versions are ideal for distribution. You distribute only the object code, not the source, so your genius stays fully protected.

BASIC Compiler runs on your TRS-80™ Model I with 48K and disk drive. The package includes BASIC Compiler, linking loader and BASIC library with complete documentation **$195.00**

1980 INCOME TAX PAC

Completely Revised - Latest Tax Tables - Fully Tested - Complete Manual and Documentation. The new version of the Income Tax Pacs are full of error catching codes making it impossible to make an error. Follow the simple Step By Step procedure that makes tax preparation simple.

INCOME TAX PAC A .... ($19.95...Cassette)

For Level II 16K Cassette Only

Does Form 1040 and 1040A

- Schedule A itemized deductions
- Schedule B interest and dividends
- Output to video display
- Schedule TC tax computation

INCOME TAX PAC B .... $49.95...Cassette or Diskette)

For Level II 16K with or without printer...cassette or disk has all features of Income Tax Pac A **Plus** works with or without line printer.

- Formats Form 1040 and 1040A for standard tax forms
- Schedule C income from a personally owned business
- Form 2106 employee business expense

PROFESSIONAL INCOME TAX PAC C .... $99.95...Diskette

For Level II 32K with disk and printer (optional)

Has all features of Income Tax Pac B **Plus** automatic memory storage for income tax preparers.

- 22 additional schedules and forms
- Formats forms for individual or tractor feed printing

MOD II CPA VERSION $199.95

GUARANTEED PROFIT 91% WINS PLACES SHOWS 32% AVERAGE PROFIT AT ALL TRACKS-1978

THE HORSE SELECTOR II (FLATS) (By Dr. Hal Davis **$50.00**

New simplified version of the original Horse Selector. The first Horse Selection System to actually calculate the estimated odds of each horse.

HIGHER PROFITS (OVER 100%) POSSIBLE THROUGH SELECTIVE BETTING ON:

- Rates each horse in 10 seconds.
- Easy to follow rules.
- Can be used with any Apple II Computer.
- 100% money back guarantee (returned for any reason).
- Uses 4 factors (speed rating, track variant, distance of the present race, distance of the last race).
- Using the above factors, the Horse Selector calculates the estimated odds. BET on horses whose actual payoff (from the Tote Board or Morning Lines) is higher than payoff based on estimated odds.
- Using the above factors, the Horse Selector calculates the estimated odds. BET on any selected horse with an estimated payoff (based on Tote Board or Morning Lines) higher than calculated payoff (based on Horse Selector II).
- Source listing for the TRS-80™, TI-59, HP-67, HP-41, Apple and BASIC Computers.
- No computer or calculator necessary (although a calculator would be helpful for the simple division used to calculate estimated odds).

FREE Dutching Tables allows betting on 2 or more horses with a guaranteed profit.

NEWDOS/80

A **New** enhanced NEWDOS for TRS-80™ Model I for the 1980's

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 with variable record lengths up to 4095 Bytes long.
- New BASIC commands that supports 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 disk drives or 8" disk drives, or any combination.
- A security boot-up for BASIC or machine code application programs. User never sees "DOSREADY" 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 program chaining.
- Device hanging for routing to display and printer simultaneously.
- CDE function; simultaneous striking of the C, D and E keys will allow 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.

.... **$149.00**

STOCK MARKET MONITOR

Galactic Software Ltd.

CASSETTE VERSION $89.00
DISK VERSION $99.00

1. The system is designed for the active "trader" not the "long term" investor, as the system is "technically" oriented.
2. For the TRS-80™ Model I, Level II, 16K or more. Available in both disk and tape versions.
3. Tracks user selected issues, in a technical system that reflects the issue's performance against the overall market.
4. Set up data is input by the user from the Standard and Poors stock guide or Value Line.
5. Daily issue data, "high", "low", "close" and "volume" are input from any newspaper containing this information.
6. Daily overall market, "volume" and "closing Dow" are also provided from a newspaper.
7. Volume and price changes of an issue, as they compare to volume an price changes of the overall market, are the basis of this system's analysis of the given issue.
8. Comparisons of the issue against itself are also done. This may allow the user to spot "unusual" activity on this issue.
9. Clear indications are given as to whether the issue is "out performing", "under performing" or "performing" with the market.
10. Complete video and printed output is provided.
11. This program is intended to be a guide to indications, and is not to be used as a sole recommendation to buy, sell or hold an issue. These decisions are the responsibility of the user and his brokerage.

H & E COMPUTRONICS INC.
MATHEMATICAL APPLICATIONS SERVICE™

NEW TOLL-FREE ORDER LINE
(OUTSIDE OF N.Y. STATE)
(800) 431-2818

~9

50 N. PASCACK ROAD
SPRING VALLEY, NEW YORK 10977

24 HOUR ORDER LINE
(914) 425-1535

★ *All orders processed within 24-Hours*
★ *30-Day money back guarantee on all Software (less $3 penalty for handling)*

ADD $2.00 FOR SHIPPING IN UPS AREAS
ADD $3.00 FOR C.O.D. OR NON-UPS AREAS
ADD $4.00 OUTSIDE U.S.A., CANADA & MEXICO

H&E COMPUTRONICS INC.

•••EVERYTHING FOR YOUR TRS-80™•••

TRS-80 is a trademark of the Radio Shack Division of Tandy Corporation

NEW

BUSINESS PAC 100

100 Ready-To-Run Business Programs

★ *All orders processed within 24-Hours*
★ *30-Day money back guarantee on all Software (less a $3 penalty for handling)*

(ON CASSETTE OR DISKETTE).....Includes 110 Page Users Manual.....5 Cassettes (Or Diskettes)
Inventory Control.....Payroll.....Bookkeeping System.....Stock Calculations.....
Checkbook Maintenance.....Accounts Receivable.....Accounts Payable.....

BUSINESS 100 PROGRAM LIST

1	RULE78	Interest Apportionment by Rule of the 78's
2	ANNU1	Annuity computation program
3	DATE	Time between dates
4	DAYYEAR	Day of year a particular date falls on
5	LEASEINT	Interest rate on lease
6	BREAKEVN	Breakeven analysis
7	DEPRSL	Straightline depreciation
8	DEPRSY	Sum of the digits depreciation
9	DEPRDB	Declining balance depreciation
10	DEPRDDB	Double declining balance depreciation
11	TAXDEP	Cash flow vs. depreciation tables
12	CHECK2	Prints NEBS checks along with daily register
13	CHECKBK1	Checkbook maintenance program
14	MORTGAGE/A	Mortgage amortization table
15	MULTMON	Computes time needed for money to double, triple, etc.
16	SALVAGE	Determines salvage value of an investment
17	RRVARIN	Rate of return on investment with variable inflows
18	RRCONST	Rate of return on investment with constant inflows
19	EFFECT	Effective interest rate of a loan
20	FVAL	Future value of an investment (compound interest)
21	PVAL	Present value of a future amount
22	LOANPAY	Amount of payment on a loan
23	REGWITH	Equal withdrawals from investment to leave 0 over
24	SIMPDISK	Simple discount analysis
25	DATEVAL	Equivalent & nonequivalent dated values for oblig.
26	ANNUDEF	Present value of deferred annuities
27	MARKUP	% Markup analysis for items
28	SINKFUND	Sinking fund amortization program
29	BONDVAL	Value of a bond
30	DEPLETE	Depletion analysis
31	BLACKSH	Black Scholes options analysis
32	STOCVAL1	Expected return on stock via discounts dividends
33	WARVAL	Value of a warrant
34	BONDVAL2	Value of a bond
35	EPSEST	Estimate of future earnings per share for company
36	BETAALPH	Computes alpha and beta variables for stock
37	SHARPE1	Portfolio selection model-i.e. what stocks to hold
38	OPTWRITE	Option writing computations
39	RTVAL	Value of a right
40	EXPVAL	Expected value analysis
41	BAYES	Bayesian decisions
42	VALPRINF	Value of perfect information
43	VALADINF	Value of additional information
44	UTILITY	Derives utility function
45	SIMPLEX	Linear programming solution by simplex method
46	TRANS	Transportation method for linear programming
47	EOQ	Economic order quantity inventory model
48	QUEUE1	Single server queueing (waiting line) model
49	CVP	Cost-volume-profit analysis
50	CONDPROF	Conditional profit tables
51	OPTLOSS	Opportunity loss tables
52	FQUOQ	Fixed quantity economic order quantity model

	NAME	DESCRIPTION
53	FQEOWSH	As above but with shortages permitted
54	FQEOQPB	As above but with quantity price breaks
55	QUEUECB	Cost-benefit waiting line analysis
56	NCFANAL	Net cash-flow analysis for simple investment
57	PROFIND	Profitability index of a project
58	CAP1	Cap. Asset Pr. Model analysis of project
59	WACC	Weighted average cost of capital
60	COMPBAL	True rate on loan with compensating bal. required
61	DISCBAL	True rate on discounted loan
62	MERGANAL	Merger analysis computations
63	FINRAT	Financial ratios for a firm
64	NPV	Net present value of project
65	PRINDLAS	Laspeyres price index
66	PRINDPA	Paasche price index
67	SEASIND	Constructs seasonal quantity indices for company
68	TIMETR	Time series analysis linear trend
69	TIMEMOV	Time series analysis moving average trend
70	FUPRINF	Future price estimation with inflation
71	MAILPAC	Mailing list system
72	LETWRT	Letter writing system-links with MAILPAC
73	SORT3	Sorts list of names
74	LABEL1	Shipping label maker
75	LABEL2	Name label maker
76	BUSBUD	DOME business bookkeeping system
77	TIMECLCK	Computes weeks total hours from timeclock info.
78	ACCTPAY	In memory accounts payable system-storage permitted
79	INVOICE	Generate invoice on screen and print on printer
80	INVENT2	In memory inventory control system
81	TELDIR	Computerized telephone directory
82	TIMUSAN	Time use analysis
83	ASSIGN	Use of assignment algorithm for optimal job assign.
84	ACCTREC	In memory accounts receivable system-storage ok
85	TERMSPAY	Compares 3 methods of repayment of loans
86	PAYNET	Computes gross pay required for given net
87	SELLPR	Computes selling price for given after tax amount
88	ARBCOMP	Arbitrage computations
89	DEPRSF	Sinking fund depreciation
90	UPSZONE	Finds UPS zones from zip code
91	ENVELOPE	Types envelope including return address
92	AUTOEXP	Automobile expense analysis
93	INSFILE	Insurance policy file
94	PAYROLL2	In memory payroll system
95	DILANAL	Dilution analysis
96	LOANAFFD	Loan amount a borrower can afford
97	RENTPRCH	Purchase price for rental property
98	SALELEAS	Sale-leaseback analysis
99	RRCONVBD	Investor's rate of return on convertable bond
100	PORTVAL9	Stock market portfolio storage-valuation program

☐ **CASSETTE VERSION $ 99.95**
☐ **DISKETTE VERSION $ 99.95**
☐ **MODEL II VERSION $149.95**

ADD $2.00 FOR SHIPPING IN UPS AREAS
ADD $3.00 FOR C.O.D. OR NON-UPS AREAS
ADD $4.00 OUTSIDE U.S.A, CANADA & MEXICO

NEW TOLL-FREE ORDER LINE
(OUTSIDE OF N.Y. STATE)
(800) 431-2818

COMPUTRONICS INC.
MATHEMATICAL APPLICATIONS SERVICE™

50 N. PASCACK ROAD
SPRING VALLEY, NEW YORK 10977

VISA

24 HOUR ORDER LINE
(914) 425-1535

Self help routines for frustrated programmers.

My Way

Robert V. Meushaw
4188 Brittany Dr.
Ellicott City, MD 21043

I'm sure that many of you have been faced with situations in which the program you were writing just wouldn't perform to your expectations because of limitations which seemed beyond your control. In some cases, processor speed or inefficient data storage techniques may limit your capabilities; but in almost every hopeless situation I work on, I find that, with perseverance, there is always a better way.

One of the most important lessons I have learned while programming is that you should use your computer to help you analyze and resolve your programming problems.

I would like to describe several programming applications I have faced with my Level I TRS-80 and the approaches that I used to investigate such limitations.

Problem One

Computing vector magnitudes when given the X and Y-coordinates (Fig. 1) requires a square root function that is unfortunately not built-in to the Level I. Fortunately, the appendix of the Level I manual includes a square root function (Listing I), but it performs much slower than I would have liked. Watching the results being printed, I can almost feel the burden of the routine. After this I begin to investigate how to improve the speed of my program.

Looking at Listing 1, you can see that the routine makes successive approximations to the square root of X which eventually converge (within the limits of the computer's accuracy) to the square root of X. The first approximation, Y, is taken as X/2.

The value of W keeps track of the error in the approximation and whenever the error is zero or is the same as the previous iteration, the subroutine returns with Y as the square root of X.

This seems fairly straightforward, but, as you can see, there is really quite a lot going on in one iteration, which is reason enough for its lack of speed.

Wondering how many iterations of such a routine are necessary to compute a square root, I decided to make some tests.

The test program is shown in Listing 2. It generates random numbers to be input to the square root subroutine. Each time a new number is generated, a counter is set to zero. Then the square root subroutine is called, and modified to increment the counter with each iteration.

When the subroutine returns, C contains the number of iterations necessary to compute the square root of X. The value of C is used to increment the array element A(C).

Using this technique, the array element A(I) keeps count of the number of values of X which require iterations in the square root subroutine to return an answer.

The first three lines in the program initialize the array elements to zero. Line 50 determines the value of X.

This line is not fully specified in the listing because I ran the program a number of times with different expressions for X. In each case the program examines 15,000–20,000 values of X. This often requires several hours of run time. (On several occasions I let the test run overnight.)

Usually, when a reasonable number of samples has been examined, I interrupt the program and execute the routine beginning on line 2000 which displays the percentage of samples requiring iteration counts of one to one hundred.

Some Interesting Results

The program is first run with X = RND(0). This sets X equal to

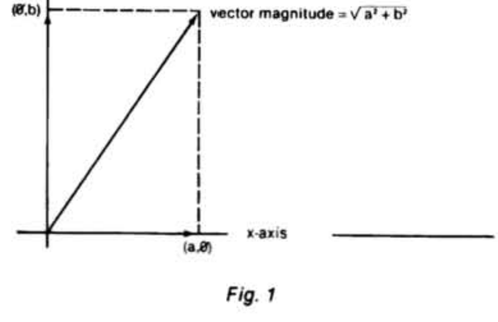


Fig. 1

a random number between 0 and 1. The resulting iterations are shown in Table 1. Beside each iteration value is its percentage.

We can see that 34.6 percent of the numbers require six iterations, 48 percent require seven iterations, and so on. The highest number of iterations recorded is fourteen, and no number requires five or less.

Table 2 shows the results using X = 1/RND(30000). The range of iterations is between seven and fourteen, with the largest percentage of the numbers requiring more than twelve.

Table 3 shows the results of X = RND(30000). These results appear similar to those of Table 2 except that the range of iterations is offset slightly.

The last set of results, shown in Table 4, is the result of using X = RND(0)*1E10. I chose this to give some very large values for X. Again, the results appear similar to those in Table 2 with a larger offset than in Table 3.

```
30010    REM *SQUARE ROOT* INPUT X, OUTPUT Y
30020    REM ALSO USES W AND Z INTERNALLY
30030    IF X=0 THEN Y=0: RET.
30040    IF X>0 THEN 30060
30050    PRINT "ROOT OF NEGATIVE NUMBER?": STOP
30060    Y=X*.5:    Z=0
30070    W=(X/Y-Y)*.5
30080    IF (W=0) + (W=Z) THEN RETURN
30090    Y=Y+W:  Z=W:  GOTO 30070
```

Listing 1. TRS-80 Level I manual square root subroutine (shorthand notation has been expanded).

The Search Begins

Now that I have a reasonable understanding of how the square root subroutine is working, and why it is consuming so much time, the problem remains to find a technique to reduce the time.

```
  10     FOR I=1 TO 100                          INITIALIZE ARRAY WHICH
  20     A(I)=0                                    COUNTS NUMBER OF SAMPLES
  30     NEXT I                                    FOR EACH ITERATION COUNT
  40     J=1                                     J COUNTS NUMBER OF SAMPLES
  50     X=(...one of several statements...)     PICK RANDOM VALUE FOR X
  60     C=0                                     INITIALIZE NUMBER OF ITERATIONS
  70     GOSUB 1000                              COMPUTE SQUARE ROOT
  80     A(C)=A(C)+1                             UPDATE PROPER COUNTER
  90     PRINT J;" VALUES HAVE BEEN USED"        PRINT NUMBER OF SAMPLES USED
 100     J=J+1                                   UPDATE SAMPLE COUNT
 110     GOTO 50                                 KEEP GOING
 120     END

1000     Y=X*.5:  Z=0                            COMPUTE APPROX. TO SQ. ROOT
1010     W=(X/Y-Y)*.5                            COMPUTE ERROR
1020     C=C+1                                   UPDATE ITERATION COUNT
1030     IF (W=0) + (W=Z) THEN RET.              IF DONE, RETURN
1040     Y=Y+W:  Z=W:  GOTO 1010                   ELSE COMPUTE NEW APPROX.

2000     T=0                                     SET TOTAL TO ZERO
2010     FOR I=1 TO 100                          COMPUTE THE SUM
2020     T=T+A(I)                                  OF THE NUMBER OF SAMPLES
2030     NEXT I                                    FOR ALL ITERATION COUNTS
2040     FOR K=1 TO 100                          SET LOOP TO PRINT ALL
2050     PRINT K, 100*A(K)/T                       ITERATION COUNTS IN
2060     NEXT K                                    PERCENTAGE OF TOTAL
```

*Listing 2. Program used to investigate the operation of the square root subroutine. The program was run with the value of X in line 50 given as X = RND(0), X = 1/RND(30000), X = RND(30000), and X = RND(0)*1E10. The program starting at line 2000, run after a sufficient number of samples were taken, prints the percentage of numbers for each iteration count.*

```
  10     FOR I=1 TO 100                          INITIALIZE ARRAY WHICH COUNTS
  20     A(I)=0                                    NUMBER OF SAMPLES FOR EACH
  30     NEXT I                                    ITERATION COUNT
  40     J=1                                     J COUNTS NUMBER OF ITERATIONS
  50     A=(...various expressions...)           PICK RANDOM VALUES
  51     B=(...various expressions...)             FOR A AND B
  60     X=A*A+B*B                               COMPUTE SQUARE OF VECTOR MAG.
  70     C=0                                     INIT. NUMBER OF ITERATIONS
  80     GOSUB 1000                              COMPUTE SQUARE ROOT
  90     A(C)=A(C)+1                             UPDATE PROPER COUNTER
 100     PRINT J;" VALUES HAVE BEEN USED"        PRINT MESSAGE
 110     J=J+1                                   UPDATE SAMPLE COUNT
 120     GOTO 50                                 KEEP GOING
 130     END

1000     Y=A+B:  Z=0                             COMPUTE FIRST APPROX.
1010     W=(X/Y-Y)*.5                            COMPUTE ERROR
1020     C=C+1                                   UPDATE ITERATION COUNT
1030     IF (W=0) + (W=Z) THEN RET.              IF DONE, RETURN
1040     Y=Y+W:  Z=W:  GOTO 1010                   ELSE MAKE NEW APPROX.
```

Listing 3. Program used to compute the square root of X using the first approximation Y = A + B.

My first thought is to investigate other methods of computing the square root. One technique, which often yields good results, is a power series approximation. These approximations are used in calculating many functions including sine, cosine, natural log and other functions. The advantage of such a technique is the elimination of the numerous iterations the computation requires—thus saving time.

Since it has been many years since I encountered such approximation techniques, I find myself digging out college books, which I had hoped never to see again. Though it was not as clear to me now how a Taylor Series or Maclaurin Series approximation works, I eventually convinced myself that there is no simple expansion which I can use.

As is clear from Tables 1-4, larger numbers require more iterations to compute the square root. Part of the reason for this is that the first approximation to the square root is taken as X/2, which becomes a worse approximation as X increases or decreases. Again, I consider a power series approximation such as:

```
IF (X<=10) THEN Y=X/2
IF (X>10) AND (X<=100) THEN Y=X/4
IF (X>100) AND (X<=1000) THEN Y=X/20
```

I experiment with several variations of this form, but none yield a significant reduction in the number of iterations.

A Glimmer of Light

Finally, I realize that I need to look for a solution to my specific application and not a general one.

Looking back at Fig. 1, I notice that I am not trying to find the square root of just any number, I am trying to find the length of the hypotenuse of a right triangle—knowing the length of each side!

Eureka!! I have found the first approximation—the sum of the lengths of the sides of the triangle (i.e., A + B).

My next task is to determine how good my approximation method really is. Listing 3 shows the program which I used

NUMBER OF ITERATIONS	% OF NUMBERS TESTED
<=5	0
6	34.6
7	48
8	13
9	3.3
10	.87
11	.16
12	.056
13	.01
14	.004
>=15	0

Table 1. Distribution of square root iteration counts for X = RND(0).

NUMBER OF ITERATIONS	% OF NUMBERS TESTED
<=6	0
7	.03
8	.04
9	.33
10	1.2
11	5.4
12	19.3
13	47.8
14	25.9
>=15	0

Table 2. Distribution of square root iteration counts for X = 1/RND(30000).

NUMBER OF ITERATIONS	% OF NUMBERS TESTED
<=3	0
4	.004
5	.02
6	.076
7	.3
8	1.15
9	4.85
10	19
11	47.4
12	27.2
>=13	0

Table 3. Distribution of square root iteration counts for X = RND(30000).

NUMBER OF ITERATIONS	% OF NUMBERS TESTED
<=14	0
15	.05
16	.17
17	.95
18	3.51
19	14.12
20	47.1
21	34.1
>=22	0

*Table 4. Distribution of square root iteration counts for X = RND(0)*1E10.*

NUMBER OF ITERATIONS	% OF NUMBERS TESTED
<=2	0
3	1.1
4	13
5	52.3
6	33.6
>=7	0

Table 5. Distribution of square root iteration counts using A = RND(0) and B = RND(0) and modified first approximation.

to generate the lengths of each side of the right triangle and the number whose square root I needed. As in the previous set of test cases, program statements 50 and 51 are incomplete because I used four sets of values to test the same conditions as before.

The program is first run using A = RND(0) and B = RND(0). The value of X is computed as (A*A + B*B). Table 5 shows the result of using the new first approximation for the square root.

Compared with Table 1, the number of iterations is significantly reduced. In this case, most numbers require five or fewer iterations whereas Table 1 shows that most numbers require seven or more iterations.

Table 6 shows the results of the new approximation when A = RND(3000) and B = RND (30000). Notice that the results are remarkably similar to those obtained in Table 5.

Table 7 shows the results obtained using A = 1/RND(30000) and B = 1/RND(30000), and Table 8 shows the results obtained using A = RND(0)*1E10 and B = RND(0)*1E10.

I am astounded that the results obtained using the new approximation are almost identical in the four cases that I tested. I noticed two other important facts. Most of the numbers required five or six iterations, and also no numbers more than six iterations.

This second fact is most important since it tells me that I can construct a new square root routine eliminating the logical tests necessary to determine completion of the routine.

Referring to Listing 1, I can remove line 30080 by changing the basic structure of the routine to

"The savings in storage amounts to about 25 bytes, which is enough to give me the breathing room I need."

a FOR-NEXT loop using six iterations. Listing 4 gives the resulting square root routine, a faster version.

It seems wasteful to perform six iterations when some numbers require less, but looking at the statistics of Tables 5–8 you can see that 85 percent of the numbers require five or six iterations. Therefore, the instances of inefficiency are small.

NUMBER OF ITERATIONS	% OF NUMBERS TESTED
<=2	0
3	1.1
4	12.7
5	52.1
6	34.1
>=7	0

Table 6. Distribution of square root iteration counts using A = RND(30000) and B = RND(30000) and modified first approximation.

NUMBER OF ITERATIONS	% OF NUMBERS TESTED
1	0
2	.01
3	.96
4	13.16
5	51.6
6	34.27
>=7	0

Table 7. Distribution of square root iteration counts using A = 1/RND(3000) and B = RND(30000) and modified first approximation.

✓Reader Service—see page 226

Now that I am over the major hurdle, I want to make my computations even more efficient. My primary target is the parentheses which appear in line 120 of Listing 4.

I first experiment with alternate methods of writing the approximation equations. Fig. 2 shows some of the equation derivations. The first equation shows the value of the second approximation to the square root of X. In this equation, Y2 is the second approximation and Y1 is the first. As you can see, Y2 can be simply computed from Y1.

Continuing to the next approximation, Y3 is simply expressed in terms of Y2. In equation 3, the substitution is made for Y2. This equation reveals something very interesting. If we assign the value of (Y1 + X/Y1) to Z, then two successive approximations can be simply computed using the equations shown in Fig. 3.

The fact that we have condensed two approximations into this set of equations allows us to rewrite the square root routine as shown in Listing 5. Notice that only 3 iterations of the FOR-NEXT loop are required and that the parentheses are no longer included.

My final modification to increase the speed of the routine is shown in Listing 6. Here, the FOR-NEXT loop is replaced with three sets of equations which compute six approximations. This straight line coding requires more bytes of storage than the previous routine, but is slightly faster.

Listing 7 is a test of the various modifications which I have developed. It times them. Five subroutines are used for the various tests. The first subroutine is an immediate RETURN. I use this to determine how much time the main routine requires.

The four other routines are the normal square root routines from the Level I manual: the normal routine, using a modified first approximation and a FOR-NEXT loop, the routine using the modified approximation equation, and the routine using the straight line coding of the approximation.

$$Y2 = Y1 + \frac{1}{2}\left(\frac{X}{Y1} - Y1\right) = \frac{1}{2}\left(\frac{X}{Y1} + Y1\right)$$

$$Y3 = Y2 + \frac{1}{2}\left(\frac{X}{Y2} - Y2\right) = \frac{1}{2}\left(\frac{X}{Y2} + Y2\right)$$

$$= \frac{1}{2}\left[\frac{X}{\frac{1}{2}\left(\frac{X}{Y1} + Y1\right)} + \frac{1}{2}\left(\frac{X}{Y1} + Y1\right)\right]$$

$$= \frac{X}{\frac{X}{Y1} + Y1} + \frac{1}{4}\left(\frac{X}{Y1} + Y1\right)$$

Fig. 2. Equations for the second and third approximations to the square root of X.

$$Z = Y + X/Y$$
$$Y = Z/4 + X/Z$$

Fig. 3. Equations which combine two successive approximations to the square root of X.

```
100      Y=A+B           COMPUTE APPROX. TO SQ. RT.
110      FOR K=1 TO 6    SET LOOP FOR SIX ITERATIONS
120      W=(X/Y-Y)*.5    COMPUTE ERROR
130      Y=Y+W           COMPUTE NEW APPROX.
140      NEXT K          LOOP
150      RET.
```

Listing 4. Modified square root routine using first approximation and FOR-NEXT loop.

```
100      Y=A+B           COMPUTE APPROX. TO SQ. RT.
110      FOR K=1 TO 3    SET LOOP FOR THREE ITERATIONS
120      Z=Y+X/Y         COMPUTE TWO APPROXIMATIONS
130      Y=Z/4 + X/Z       USING MODIFIED EQUATIONS
140      NEXT K          LOOP
150      RET.
```

Listing 5. Revised square root routine using new first approximation and combined approximation equations. Note that only three loop iterations are required.

```
100      Y=A+B           COMPUTE APPROX. TO SQ. RT.
110      Z=Y+X/Y           USING THREE SETS OF
120      Y=Z/4 + X/Z       MODIFIED EQUATIONS TO
130      Z=Y+X/Y           COMPUTE SIX
140      Y=Z/4 + X/Z       APPROXIMATIONS
150      Z=Y+X/Y
160      Y=Z/4 + X/Z
170      RET.
```

Listing 6. Final square root routine which eliminates FOR-NEXT loop to maximize speed.

✓Reader Service—see page 226

proximation equations.

The results of the test are shown in Table 9. After subtracting the five second time of the main routine from the individual test times, the subroutine times can be compared. Execution speed is cut by a factor of more than eleven in going from the normal square root routine to my final specialized version. An improvement of this magnitude makes the work involved seem well worth the trouble.

Problem 2

The second example I want to discuss involves an application in which my primary concern is program space and accuracy. Because of the limits of the TRS-80 to slightly more than 3500 bytes of usable storage, some programs can become pretty cramped.

My problem is to construct an array which contains values of sin(X) over a range of 0 to π/2 (i.e., 0°-90°). As in the case of the square root function, Level I BASIC does not contain the sine function. So, again I turn to the Level I manual's sine function subroutine. This routine is shown in Listing 8.

The basic sine routine calculation is performed using a power series approximation. Many mathematics text books will show that sin(X) can be expressed as the infinite series:

sin(X) = X − (X**3)/6! + (X**5)/5! − (X**7)/7! + (X**9)/9! − . . .

In the case of listing 8, the power series is computed using terms up to X**9.

```
10      FOR I=1 TO 100                SET LOOP FOR 100 ITERATIONS
20      A=RND(30000): B=RND(30000)    PICK VALUES FOR A AND B
30      X=A*A + B*B                   COMPUTE SQUARE OF VECTOR MAG.
40      GOSUB 100                     COMPUTE SQUARE ROOT
50      NEXT I                        LOOP
60      PRINT "TIMING TEST COMPLETED"  PRINT MSG. WHEN DONE
70      END
```

Listing 7. Program used to test the efficiency of the various square root subroutines.

```
30370   REM *SIN* INPUT X IN DEGREES, OUTPUT Y
30371   REM ALSO USES Z INTERNALLY
30376   Z=ABS(X)/X:  X=Z*X
30380   IF X>360 THEN X=X/360: X=(X-INT(X))*360
30390   IF X>90 THEN X=X/90: Y=INT(X): X=(X-Y)*90: ON Y GOTO
                  30410, 30420, 30430
30400   X=X/57.29578: IF ABS(X)<2.4861E-4 THEN Y=0: RET.
30405   GOTO 30440
30410   X=90-X:  GOTO 30400
30420   X=-X:  GOTO 30400
30430   X=X-90:  GOTO 30400
30440   Y=X-X*X*X/6+X*X*X*X*X/120-X*X*X*X*X*X*X/5040
30450   Y=Y+X*X*X*X*X*X*X*X*X/362880:  IF Z=-1 THEN Y=-Y
30460   RET.
```

Listing 8. TRS-80 Level I manual sine function subroutine.

```
100     Q=X                           Q IS SET TO FIRST TERM IN PWR. SERIES
110     Y=X                           Y IS SUM OF POWER SERIES TERMS
120     FOR J=3 TO 9 STEP 2           SET LOOP TO USE TERMS UP TO X**9
130     Q=-Q * X/J *X/(J-1)           COMPUTE NEXT PWR. SERIES TERM
140     Y=Y+Q                         ADD TERM TO SUM
150     NEXT J                        LOOP
160     RETURN
```

Listing 9. Modified sine routine. The statements of this routine could be used to replace the power series expression on lines 30440 and 30450 of Listing 8.

```
10      FOR I=0 TO 32                 SET LOOP FOR 32 SINE VALUES
20      X=1.570796З * I/32            X IS INPUT (0<=X<= /2)
30      T=X                           SAVE X
40      N=9:  GOSUB 1000              COMPUTE SQUARE ROOT
50      A(1)=Y                        SAVE ANSWER
60      X=T                           RESTORE X
70      N=11:  GOSUB 1000             COMPUTE NEW SQUARE ROOT
80      A(2)=Y                        SAVE ANSWER
90      PRINT A(1),A(2)               PRINT RESULTS
100     IF A(1)<>A(2) P."DIFFERENT":P. PRINT MSG. IF VALUES DIFFERENT
110     NEXT I                        LOOP
120     END

1000    Q=X                           COMPUTE SINE OF X
1010    Y=X                             USING THE HIGHEST
10 20   FOR J=3 TO N STEP 2             POWER TERM SPECIFIED
1030    Q=-Q * X/J *X/(J-1)             BY THE CALLING
1040    Y=Y+Q                           ROUTINE
1050    NEXT J
1060    RET.
```

Listing 10. Program used to test the effect upon SIN(X) of different power series approximations.

The Approach

When it becomes apparent that I need a few more bytes of storage for data, I examine the sine subroutine to see how many bytes I can squeeze out.

Computing the terms of the power series seems fairly repetitive and they seem to lend themselves to a FOR-NEXT loop. After several tries, I arrive at the routine shown in Listing 9.

The variable Q generates the terms of the power series. Each successive value of Q is computed by multiplying the previous value by −X**2 and dividing by the product of O, the loop index and (J-1). The variable Y keeps a running sum of the power series terms.

The savings in terms of storage amounts to about 25 bytes, which is enough to give me the breathing room I need. In fact, more savings are possible if unnecessary lines in the resulting sine routine are eliminated.

There is an added benefit to the modified sine routine. It is very simple and costs nothing in terms of storage, to obtain more accuracy. All that is necessary is to replace the nine in line 120 of Listing 9 with the value of the highest power term desired in the approximation. For example, 15 can be used.

The Final Test

Just to satisfy my curiosity, I decide to try the routine with more terms. I first try using 11 as the highest power term. The test routine is shown in Listing 10. It uses the parameter N to specify for the sine routine the highest power term in the approximation.

For each value of X, the sine routine is called twice—once using terms up to X**9 and once using terms up to X**11. The values obtained using each approximation are printed, and in

NUMBER OF ITERATIONS	% OF NUMBERS TESTED
<=2	0
3	1
4	13.5
5	51.3
6	34.2
>=7	0

*Table 8. Distribution of square root iteration counts using A = RND(0)*1E10 and B = RND(0)*1E10 and modified first approximation.*

SUBROUTINE USED	TOTAL TIME	TOTAL SUBROUTINE TIME
Immediate RETURN	5 sec.	—
Normal square root	107 sec.	102 sec.
Listing 4 routine	26 sec.	21 sec.
Listing 5 routine	16 sec.	11 sec.
Listing 6 routine	14 sec.	9 sec.

Table 9. Execution times of the various square root routines.

addition the two values are compared and a message is printed when the values are different.

The results of this test are both interesting and confusing. The approximation using the X**11 term is more accurate, as I had hoped. However, some results which appear to be the same when printed, produce a message which says that they are different.

I took this to mean that the internal binary representations of the values are slightly different. A portion of the results of the test is shown in Table 10. The confusing part of the results appears in the last computed value. The printed value displays 1.0000008, which contains more digits than I have ever been able to print for a number (the normal number of digits is six).

Next, I try adding another term to the power series approximation (i.e., (X**13)/13!) but it produces no further improvement in accuracy.

Aside from these few puzzling occurrences, which I have yet to fully resolve, the results of my experiments are successful. I have achieved a savings in program space and an increase in accuracy.

The Moral

I hope that the two examples that I have presented will encourage you to find new and better ways to program routines in your applications. Don't assume that the method someone else uses is necessarily the best way for you. I'm sure that you too will discover that finding a better way is perhaps one of the most rewarding aspects of programming.

Most important of all, however, remember to make use of all your resources—including your computer—to improve your work. ■

VALUE OF SIN(X) USING X**9 TERM	VALUE OF SIN(X) USING X**11 TERM	TEST RESULT
.903989	.903898	different
.92388	.92388	different
.941544	.941544	different
.956941	.956941	different
.970032	.970031	different
.980786	.980785	different
.995186	.995185	different
.998798	.998795	different
1	1.0000008	different

Table 10. Partial listing of results from test program shown in Listing 10. Note that some values are not equal in a logical equivalence test even though the printed values are the same.

Article storage and retrieval made easy.

Reference Library Index

James P. Morgan
2386-B Ash Creek
Scott AFB, IL 62225

Tired of wading through a foot-high stack of computer publications looking for a specific article? Why not let your computer do it for you?

Better yet, put your micro to work building a cross-index of your entire inventory of computer publications—past, present and future.

The following program provides a system of cataloging, storing and retrieving specific articles from your library. It is designed for the TRS-80 Level II with 16K.

Program Details

The program begins with a menu. They are as follows: A—Instructions, B—retrieve list from internal data, C—read tape file and list and D—create data file on tape. Selection is made from the keyboard using INKEY$ (lines 110 to 150). An error trap is located at line 150.

The instructions provide general information about the program.

The retrieve list from internal data option first prints the indexing format (Table 1) called from subroutine 1000.

Next, the program asks for three inputs, used to retrieve selected articles from an internal data source beginning at line 3000. Enter the system number, the category number and the topic number in turn. These three codes are compared to each READ execution at line 250 for a positive match. If all three match the data provided, the article is printed by line 260.

By entering a 0 at line 190, the user can select all the articles pertaining to a specific system. This is accomplished at line 245 which branches directly to the PRINT line at 260.

A paging routine using the variable D as a counter is provided by lines 220, 260 and 270 through 300. The advantage of retrieving data from internal storage is speed; its disadvantage is the memory overhead required. I have listed a sample block of data starting at line 3000 which pertains to my own particular interests. A more comprehensive, although appreciably slower, option is discussed in the next paragraph.

The C option, "read tape file and list," allows the user to select and list articles from tape storage. The listing begins at line 440 and runs through line 660. This routine is essentially the same as the internal option, but uses a tape file to provide data.

Once again, the indexing scheme is presented via subroutine 1000. The user enters the system, category and topic numbers that select articles at line 540.

If the user selects 0 in response to the category input, the program provides a complete listing of all articles filed under a specific system. Paging is provided at lines 560–580. An escape option is found at line 585 which allows the user to return to the menu during the paging routine.

Although this method of data storage is not as fast as an internal data bank, it is fast enough to scan an entire file of *Microcomputing* articles for a year's period in about 20 minutes. I've spent about twice that time digging out an article of interest from past issues, so the program results in definite time savings.

The final option is the routine that creates a tape data file. As with the rest of the program, it is self prompting. This segment begins at line 2000 and ends at line 2160. The indexing scheme is presented prior to input so you can assign index values to a given article.

SYSTEM	CATEGORY		
	1. HARDWARE	2. SOFTWARE	3. LANGUAGE
1. TRS-80	1. INTERFACE	1. BUSINESS	1. BASIC
2. KIM	2. CASSETTE	2. HOME	2. MACHINE
3. PET	3. DISK	3. GAMES	3. FORTRAN
4. APPLE	4. PRINTERS	4. GRAPHICS	4. PASCAL
5. HEATH	5. DISPLAYS	5. EDUCATIONAL	5. GENERAL
6. SWTP	6. I/O	6. GENERAL	
7. ALTAIR	7. KEYBOARDS		
8. ELF	8. SYNTHESIZERS		
9. GENERAL	9. GENERAL		

The user first enters the system number, then one of the three categories, and finally the specific topic from the category selected. If a complete listing of all articles for a given system is required, enter a 0 when asked to input the category.

Table 1

I found it very easy to load data using the table of contents of most magazines. Once I had loaded my entire library of publications, it only required about ten minutes to write each new issue's articles to tape storage.

Other modifications could be made to include disk storage in place of cassettes and line or screen printer options. As written, it should prove a more than adequate method of cross-indexing for the TRS-80 owner. ■

Program Listing

```
10 REM  *** ARTICLE INDEXER ***
20 REM  *** BY jAMES p. mORGAN  SEP 79 ***
30 REM  *** FOR TRS-80 LEVEL ii, 16K  ***
40 CLS:CLEAR 200
50 REM  *** POKE STATEMENT FOR REVISION G, TRS-80 ***
60 POKE 16553,255
70 PRINT@20,"COMPUTER ARTICLE CROSS INDEXER":PRINT
80 PRINT"SELECT YOUR OPTION FROM THE FOLLOWING LIST. JU
   ST
90 PRINT"PRESS THE KEY CORRESPONDING TO YOUR CHOICE.":P
   RINT
100 PRINT"A=INSTRUCTIONS","B=RETRIVE LIST FROM INTERNAL
     DATA"
105 PRINT"C=READ TAPE FILE AND LIST","D=CREATE DATA FIL
    E ON TAPE"
110 B$=INKEY$:IF B$="" THEN 110
120 IF B$="A" THEN 1200
130 IF B$="B" THEN 170
140 IF B$="C" THEN 440
145 IF B$="D" THEN 160
150 CLS:PRINT"i DON'T UNDERSTAND. SELECT A,B,C,OR D.":G
    OTO 80
160 GOTO 2000
170 RESTORE:CLS:GOSUB 1000
180 PRINT:INPUT"ENTER SYSTEM #";A1
190 INPUT"ENTER CATEGORY #. ENTER '0' TO SEE ALL SYSTEM
     ARTICLES.";B1
195 IF B1=0 THEN 220
200 INPUT"ENTER TOPIC #";C1
210 REM *** READ AND PRINT DATA ***
220 D=0:CLS: PRINT@20,"SELECTED ARTICLES":PRINT
230 READ A,B,C,A$
240 IF A=0 THEN 310
245 IF A=A1 AND B1=0 THEN 260
250 IF A=A1 AND B=B1 AND C=C1 THEN 260 ELSE 230
260 PRINTA$:D=D+1
270 IF D=12 THEN 280 ELSE 230
280 PRINT"PRESS ANY KEY WHEN FINISHED WITH THIS PAGE."
290 B$=INKEY$:IF B$="" THEN 290
300 GOTO 220
310 RESTORE:PRINT:PRINT"OUT OF DATA. TYPE 'A' FOR NEW L
    IST, 'B' TO INPUT ADDITIONAL
320 PRINT"DATA FROM TAPE, OR 'C' TO EXIT PROGRAM."
330 B$=INKEY$:IF B$="" THEN 330
340 IF B$="A" THEN 170
350 IF B$="B" THEN 450
360 IF B$="C" THEN 1500
370 CLS:PRINT"I DON'T UNDERSTAND. TRY AGAIN.":GOTO 310
440 PRINT"PREPARE RECORDER FOR DATA INPUT.":FORX=1TO150
    0:NEXT
450 CLS:GOSUB 1000
460 PRINT:INPUT"ENTER SYSTEM #";A1
470 INPUT "ENTER CATEGORY #. ENTER '0' TO SEE ALL SYSTE
    M ARTICLES";B1
475 IF B1=0 THEN 510
480 INPUT "ENTER TOPIC #";C1
490 CLS
510 CLS:D=0:PRINT@20,"SELECTED ARTICLES":PRINT
520 INPUT#-1, A,B,C,A$
530 IF A=0 THEN 600
535 IF B1=0 AND A1=A THEN 550
540 IF A=A1 AND B=B1 AND C=C1 THEN 550 ELSE 520
550 PRINTA$ :D=D+1
560 IF D=11 THEN 570 ELSE 520
570 PRINT:PRINT"HIT 'E' TO ESCAPE. HIT ANY KEY WHEN FIN
    ISHED WITH THIS PAGE."
580 B$=INKEY$: IFB$="" THEN 580
585 IF B$="E" THEN 40
590 GOTO 510
600 PRINT"OUT OF DATA. TYPE 'A' FOR NEW LIST, 'B' TO IN
    PUT ADDITIONAL
610 PRINT"DATA FROM ANOTHER TAPE, OR 'C' TO EXIT PROGRA
    M."
620 B$=INKEY$:IF B$="" THEN 620
630 IF B$="A" THEN 450
640 IF B$="B" THEN 500
650 IF B$="C" THEN 1500
660 CLS: PRINT"i DON'T UNDERSTAND, TRY AGAIN.":GOTO 600
1000 CLS:PRINT"SYSTEM                        CATEGORY":PRIN
     T
1010 PRINTTAB(16)"1. HARDWARE","2. SOFTWARE","3. LANGUA
     GE"
1020 PRINT STRING$(60,"-")
1030 PRINT"1. TRS-80","1. INTERFACE","1. BUSINESS","1.
     BASIC"
1040 PRINT"2. KIM","2. CASSETTE","2. HOME","2. MACHINE"
1050 PRINT"3. PET","3. DISC","3. GAMES","3. FORTRAN"
1060 PRINT"4. APPLE","4. PRINTERS","4. GRAPHICS","4. PA
     SCAL"
1070 PRINT"5. HEATH","5. DISPLAYS","5. EDUCATIONAL","5.
      GENERAL"
1080 PRINT"6. SWPT","6. i/o","6. GENERAL"
1090 PRINT"7. ALTAIR","7. KEYBOARDS"
1100 PRINT"8. ELF","8. SYNTHESIZERS"
1110 PRINT"9. GENERAL","9. GENERAL"
1120 RETURN
1200 REM *** INSTRUCTIONS ***
1210 CLS:PRINT"THIS PROGRAM PROVIDES YOU WITH A METHOD
     OF STORING AND
1220 PRINT"RETRIVING SELECTED ARTICLES FROM YOUR REFERE
     NCE LIBRARY.
1230 PRINT"YOU CAN STORE DATA INTERNALLY, CREATE A TAPE
      DATA FILE, AND
1240 PRINT"SEARCH A TAPE FILE FOR SPECIFIC ARTICLES. TH
     E KEY TO THE
1250 PRINT"PROGRAM KS AN INDEXING SYSTEM WHICH CATEGORI
     ZES THE ARTICLES
1260 PRINT"SYSTEMATICALLY. iNTERNAL DATA CAN BE STORED
     FROM LINE
1270 PRINT"3000 ON. IT IS LIMITED ONLY BY THE MEMORY AV
     AILABLE. THE
1280 PRINT"TAPE STORAGE METHOD IS UNLIMITED IN SPACE, B
     UT IS A SLOWER
1290 PRINT"METHOD OF EXTRACTING DATA. THE PROGRAM IS SE
     LF-PROMPTING
1300 PRINT"THROUGHOUT. JUST FOLLOW THE DIRECTIONS AND Y
     OU SHOULD HAVE
1310 PRINT"NO TROUBLE CATELOGING YOUR ENTIRE LIBRARY."
1320 PRINT:PRINT"PRESS ANY KEY TO RETURN TO THE MENU."
1330 B$=INKEY$:IF B$=""THEN 1330
1340 CLS:GOTO70
1500 CLS:END
2000 REM *** CASSETTE DATA WRITE ***
2010 CLS:PRINT"CASSETTE DATA WRITE":PRINT
2020 PRINT"PREPARE CASSETTE AND TAPE FOR RECORDING."
2040 PRINT:PRINT"ENTER SYSTEM #, CATEGORY #, TOPIC #, A
     ND ARTICLE TITLE,
2050 PRINT"PUBLICATION, DATE, AND PAGE. THE ENTIRE ARTI
     CLE MUST BE
2055 PRINT"ENCLOSED IN QUO
2060 PRINT"MARK THE END OF A TAPE WITH A SERIES OF ZERO
     S."
2061 PRINT"TYPE AN 'E' TO LEAVE THE WRITE ROUTINE. HIT
     ANY KEY TO CONTINUE."
2062 B$=INKEY$:IFB$=""THEN 2062
2064 IF B$="E" THEN 2130
2068 CLS:GOSUB 1000
2070 PRINT:INPUT"SYSTEM #";A
2080 IF A=0 THEN 2130
2090 INPUT"CATEGORY, TOPIC #. SEPARATE WITH A COMMA";B,
     C
2100 INPUT"TITLE, PUBLICATION, ISSUE, AND PAGE#--ALL IN
      QUOTES";A$
2110 PRINT#-1, A,B,C,A$
2120 CLS:GOTO 2068
2130 CLS:PRINT"DATA WRITE COMPLETE. NOTE TAPE COUNTER A
     ND LOG IN."
2140 PRINT"HIT ANY KEY TO RETURN TO THE MENU."
2150 B$=INKEY$: IF B$=""THEN 2150
2160 CLS:GOTO 70
3000 DATA 1,2,1,"INVENTORY,TRS-80,KB; FEB 79, P 64"
3010 DATA 9,2,2,"SIMPLER INTEREST, KB, FEB 79, P 116"
3020 DATA 9,2,1,"SIMPLER INTEREST, KB, FEB 79, P 116"
3030 DATA 1,2,3,"KEYBOARD INTERRUPT-TRS 80, KB, MAR 79,
      P.128"
3040 DATA 1,1,9,"A LOOK AT 80 PERIPHERALS, KB, APR 79,
     P. 22"
3050 DATA 1,2,3,"FREE SPEACH LESSONS, 80, KB, APR 79,P.
     66"
3060 DATA 1,1,2,"TWIN CASSETTES FOR 80, KB, APR 79, P.
     84"
3070 DATA 1,2,6,"A LOOK INSIDE THE 80, KB, APR 79, P. 1
     20"
3080 DATA 1,2,1,"A TRS-80 CROSS INDEX, KB, MAY 79, P. 9
     4"
3090 DATA 1,2,4,"GRAPHING WITH THE 80, KB, MAY 79, P. 1
     00"
3100 DATA 1,1,4,"80 SELECTRIC WORD PROCESSOR, KB, JUN 7
     9, P. 32"
3110 DATA 1,2,3,"SARGON MEETS 80, KB, JUL 79, P. 58"
3120 DATA 1,2,4,"GETTING THE MOST OUT OF 80, KB, JUL 79
     , P. 112"
3130 DATA 1,1,2,"TELEPRINTER OUTPUT FOR 80, KB, AUG 79,
      P. 38"
3140 DATA 1,3,2,"MACHINE LANG MONITOR FOR 80,KB, AUG 79
     ,P.114"
3150 DATA 1,1,9,"TRS SPEED UP, KB, SEP 79, P. 138"
3160 DATA 9,2,3,"SUPER MASTER MIND, KB, FEB 79, P.100"
3170 DATA 9,2,3,"CHESS PAWN, KB, MAR 79, P. 76"
3180 DATA 9,2,3,"TWO DIAMONDS, KB, APR 79, P. 115"
3190 DATA 9,2,3,"A GAME OF DARTS, KB, MAY 79, P. 78"
3200 DATA 9,2,4,"VECTOR GRAPHING TECH. KB, JUN 79, P. 6
     4"
3210 DATA 9,2,4,"A CIRCULAR HANDLE ON GRAPHICS, KB,JUL
     79,P.76"
3220 DATA 9,2,3,"NERVES, KB, AUG 79, P. 100"
3230 DATA 0,0,0,"0"
```

SAVE / add-ons for TRS-80® Software and Hardware

NEWDOS 80

A new enhanced NEWDOS for the TRS-80.

The most powerful Disk Operating System for the TRS-80, designed for the sophisticated user and professional programmer who demands the ultimate.

NEWDOS/80 is the planned upgrade from NEWDOS 2.1. Some of the features are:

- New BASIC commands for files with variable record lengths up to 4095.
- Mix or match drives. Use 35, 40 or 80 track 5" disk drives or 8" disk drives, or combo.
- Security boot-up for BASIC or machine code application programs.
- New editing commands.
- Enhanced RENUMber that allows relocation.
- Command chaining.
- Device handling for routing to display and printer simultaneously.
- DFG function; striking of D, F and G keys allows user to enter a mini-DOS without disturbing program.
- Compatible with NEWDOS & TRSDOS.
- Machine language Superzap/80 2.1 utilities and enhanced debug and copy. **$149**

FACTORY CLEARANCE

Demo single or dual head MPI disc drive, complete with Power Supply and Chassis. Full warranty.

TF5D 40 TK **$279**

TDH-1D Dual 35 TK **$419**

LIMITED QUANTITIES

NEW

TF-8 80 TRACK DISK DRIVE **$639**
Double Your Capacity

TF-9 DUAL 80 TRACK DISK DRIVE **$799**
Quadruple Your Capacity

Announcing

8" Floppy Disk Drive System

for Model I

- One SA801 Floppy
- NEWDOS 80
- Cable & Adapter

$1095

Disk Drive Sale!

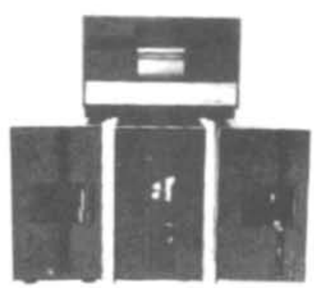
Complete with power supply and chassis.

TF-3 Shugart SA400	$359
Pertec FD200, 40 track	$389
TF 5 MPI B51, 40 track	$379
TF7 Micropolis 77 track	$595
TDH-1 Dual Sided drive 35 track	$499

NEWDOS+ 40 track **$110** 35 track $99

Microconductor, Data Base Mgr. Mod I .... **$249** Mod II $399

AJA Business Pgk	$289
Disk Head Cleaner	$16.95
Disk Alignment Pgm	$109



Disk Expansion System

• 2 Shugart SA400 TF-3	$718
• 1 Two-Drive Cable	$ 25
• 1 Expansion Interface 32K	$489
• 1 35-track DOS+	$ 99
TOTAL LIST PRICE	$1331

SPECIAL PRICE ONLY **$1,199**

Same as above but includes TRS-80® Dual II $1984

MOD II 8" Disk System

- 1 Drive System $899
- 2 Drive Expansion System $1,399

Drives for any Microcomputer

Does not include power supply & cabinet.

Pertec FD200	$282	FD250	$399
Shugart SA400	$279	SA800/801	$479
MPI B51	$279	B52	$349

Printers

Centronic 779	$1,069
Base 2	$599
Centronic 737	$899
Centronics 702-9	$1,895
Anadex 9501	$1,595
Malibu	$2,495

Spinwriter **$2,549**

More Savings

Introductory Offer—Save $250 TRS-80 Graphics Okidata Microline 80 . . .

List $949

Our Price **$699**

Memory Kit (16K)	$64.00
AC Isolator (6 socket)	$47.95
Modem	$164.00
Diskettes (10)	$30.00

↙264

Apparat, Inc.

(303) 741-1778

7310 E. Princeton Ave • Denver, CO 80237 • (303) 758-7275

MTI MICROCOMPUTER TECHNOLOGY INCORPORATED ↙28

3304 W. MacArthur • Santa Ana, CA 92704 • (714) 979-9923

TX 67840 TAB-IRIN

All prices cash discounted/Freight: FOB factory. Ask for our free catalog.

Professionalize your program titles.

Position Display

Jerry Frost
3398 Sir Henry Street
East Point, GA 30344

Taking the lead from James J. Brennan, whose "Digital Time Clock" program appeared in the "Little Bits" column of *Kilobaud* in the January 1979 issue, I have written a subroutine for the TRS-80. I call it POSDIS, an acronym for Position Display. It is useful and saves time.

POSDIS will let you: display a title in your program or have a message stand out from the rest of the text; to left and right justify the title; to position it on the screen by a PRINT@ statement; and finally, to graphically draw a border around it.

Type the Program

To show you the advantages of POSDIS, and to let the subroutine speak for itself, type in the routine (see Program Listing 1), SAVE it, and type the following program:

```
10 CLS
20 A$ = "THIS IS WHERE YOUR
   MESSAGE GOES"
30 GOSUB 30380
40 GOTO 40
```

Now, RUN the program. Notice that your message is automatically left and right justified, a graphic border is drawn around it for a stand-out effect, and the whole thing is displayed at the center of the screen. In a couple of seconds you have a nice looking display heading.

Here is what happened. First, you defined GOSUB A$ as "This is where your message goes." Then you decided you wanted the message to appear at the center of the screen, so you entered GOSUB 30380, which is the routine that transported the message, justified it, sent it to line 30900 to calculate the graphic border and drew it before returning to the next line in your program (in this case line 40).

Suppose you want the message at the top of the screen. Just retype line 30 GOSUB 30140. If you want the message at the bottom of the screen, retype line 30 GOSUB 30620. Try it and see what happens.

The subroutine at lines 30140 through 30620 will print your message at 13 different screen display positions and must follow your A$ message as shown in the sample program.

Three Subroutines

Three other subroutines have been added. Two timing loop series for an automatic effect, and a routine, which will hold the display until the keyboard operator presses ENTER.

For example, retype line 40 GOSUB 30680. Run the program and notice that the message appears with "PRESS ENTER TO CONTINUE" added to the bottom of the screen.

If you want an automatic effect, retype line 40 GOSUB 30820 and add line 50 A$ = "THIS IS WHERE A SECOND MESSAGE GOES" : GOSUB 30380 : GOSUB 30680.

By using various screen display positions, timing loops, and PRESS ENTER TO CONTINUE your next program will have a professional look, ease of operation and simplicity. ■

Program Listing

```
30000 '************************************************
      ***
30010 '*      "POSDIS"   SUBROUTINE  -  VERSION 1.0
      *
30020 '*
      *
30030 '*
      *
30040 '*
      *
30050 '************************************************
      ***
30060 '
30070 '
30080 REM ** SUBROUTINE LINES TO LEFT AND RIGHT JUSTIFY
      **
30090 REM **   <A$> AT 13 DIFFERENT SCREEN POSITIONS
      **
30100 '
30110 '
30120 ' JUSTIFY <A$> AND DISPLAY AT SCREEN POSITION 64
30130 '
30140 P=32-LEN(A$)/2+64:TH=0:BH=8:LV=32-LEN(A$)/2:RV=32
     +LEN(A$)/2+64:GOSUB30900:PRINT@P,A$;:RETURN
30150 '
30160 ' JUSTIFY <A$> AND DISPLAY AT SCREEN POSITION 128
30170 '
30180 P=32-LEN(A$)/2+128:TH=3:BH=11:LV=32-LEN(A$)/2:RV=
     32+LEN(A$)/2+64:GOSUB30900:PRINT@P,A$;:RETURN
30190 '
30200 ' JUSTIFY <A$> AND DISPLAY AT SCREEN POSITION 192
30210 '
30220 P=32-LEN(A$)/2+192:TH=6:BH=14:LV=32-LEN(A$)/2:RV=
     32+LEN(A$)/2+64:GOSUB30900:PRINT@P,A$;:RETURN
30230 '
30240 ' JUSTIFY <A$> AND DISPLAY AT SCREEN POSITION 256
30250 '
30260 P=32-LEN(A$)/2+256:TH=9:BH=17:LV=32-LEN(A$)/2:RV=
     32+LEN(A$)/2+64:GOSUB30900:PRINT@P,A$;:RETURN
30270 '
30280 ' JUSTIFY <A$> AND DISPLAY AT SCREEN POSITION 320
30290 '
30300 P=32-LEN(A$)/2+320:TH=12:BH=20:LV=32-LEN(A$)/2:RV
     =32+LEN(A$)/2+64:GOSUB30900:PRINT@P,A$;:RETURN
30310 '
30320 ' JUSTIFY <A$> AND DISPLAY AT SCREEN POSITION 384
30330 '
30340 P=32-LEN(A$)/2+384:TH=15:BH=23:LV=32-LEN(A$)/2:RV
     =32+LEN(A$)/2+64:GOSUB30900:PRINT@P,A$;:RETURN
30350 '
30360 ' JUSTIFY <A$> AND DISPLAY AT SCREEN POSITION 448
       (SCREEN CENTER)
30370 '
30380 P=32-LEN(A$)/2+448:TH=18:BH=26:LV=32-LEN(A$)/2:RV
     =32+LEN(A$)/2+64:GOSUB30900:PRINT@P,A$;:RETURN
```

```
30390 '
30400 ' JUSTIFY <A$> AND DISPLAY AT SCREEN POSITION 512
30410 '
30420 P=32-LEN(A$)/2+512:TH=21:BH=29:LV=32-LEN(A$)/2:RV
    =32+LEN(A$)/2+64:GOSUB30900:PRINT@P,A$;:RETURN
30430 '
30440 ' JUSTIFY <A$> AND DISPLAY AT SCREEN POSITION 576
30450 '
30460 P=32-LEN(A$)/2+576:TH=24:BH=32:LV=32-LEN(A$)/2:RV
    =32+LEN(A$)/2+64:GOSUB30900:PRINT@P,A$;:RETURN
30470 '
30480 ' JUSTIFY <A$> AND DISPLAY AT SCREEN POSITION 640
30490 '
30500 P=32-LEN(A$)/2+640:TH=27:BH=35:LV=32-LEN(A$)/2:RV
    =32+LEN(A$)/2+64:GOSUB30900:PRINT@P,A$;:RETURN
30510 '
30520 ' JUSTIFY <A$> AND DISPLAY AT SCREEN POSITION 704
30530 '
30540 P=32-LEN(A$)/2+704:TH=30:BH=38:LV=32-LEN(A$)/2:RV
    =32+LEN(A$)/2+64:GOSUB30900:PRINT@P,A$;:RETURN
30550 '
30560 ' JUSTIFY <A$> AND DISPLAY AT SCREEN POSITION 768
30570 '
30580 P=32-LEN(A$)/2+768:TH=33:BH=41:LV=32-LEN(A$)/2:RV
    =32+LEN(A$)/2+64:GOSUB30900:PRINT@P,A$;:RETURN
30590 '
30600 ' JUSTIFY <A$> AND DISPLAY AT SCREEN POSITION 832
30610 '
30620 P=32-LEN(A$)/2+832:TH=36:BH=44:LV=32-LEN(A$)/2:RV
    =32+LEN(A$)/2+64:GOSUB30900:PRINT@P,A$;:RETURN
30630 '
30640 '
30650 '
30660 REM ** ROUTINE TO HALT DISPLAY UNTIL OPERATOR INP
    UT **
30670 '
30680 PRINT@32-25/2+960,"PRESS <ENTER> TO CONTINUE";:IN
    PUTA$:CLS:RETURN
30690 '
30700 '
30710 '
30720 REM ** TIMING LOOP  **
30730 '
30740 FOR X=1 TO 500: NEXT X: RETURN
30750 FOR X=1 TO 1000: NEXT X: RETURN
30760 FOR X=1 TO 1500: NEXT X: RETURN
30770 FOR X=1 TO 2000: NEXT X: RETURN
30780 '
30790 REM ** TIMING LOOP THEN CLOSE SCREEN <CLS> **
30800 '
30810 FOR X=1 TO 500: NEXT X: CLS: RETURN
30820 FOR X=1 TO 1000: NEXT X: CLS: RETURN
30830 FOR X=1 TO 1500: NEXT X: CLS: RETURN
30840 FOR X=1 TO 2000: NEXT X: CLS: RETURN
30850 '
30860 '
30870 '
30880 REM ** SUBROUTINE TO DRAW BORDER AROUND <A$> **
30890 '
30900 FOR X=LV TO RV:SET(X,TH):SET(X,BH)::NEXT X:FOR Y=
    TH TO BH:SET(LV,Y):SET(RV,Y):NEXT Y:RETURN
```

A case for string packing.

Stringy Machine Code

David D. Grimes
12949 W. Montana Dr.
Lakewood, CO 80228

The string packing technique for embedding machine language code in BASIC programs offers exciting possibilities and convenience to TRS-80 users.

For BASIC programs with machine language routines you no longer need to load a separate SYSTEM tape, specify a power up MEMORY SIZE or modify programs for different RAM sizes.

String packing takes care of these problems, making it possible to wed BASIC and machine languages.

Eliminates SYSTEM Tape

For example, printing game instructions on the screen in machine language is difficult and tedious, but in BASIC it's easy. Computing generations in the game of Life can take thirty seconds or more in BASIC, while it takes only fractions of a second in machine language.

Having both languages in one program eliminates the need for SYSTEM tapes, because all data formerly on the SYSTEM tape can be CSAVEd as part of the BASIC program, and the BASIC program knows exactly where to find the machine language in RAM, via the USR command.

Furthermore, since the machine language is stored as part of the BASIC program, it isn't going to overlay itself, so no upper RAM need be saved at power-up time.

Let's examine a practical application using string packing—an audible prompt. The audible prompt (beep) is used when a program needs keyboard input, when an error is detected, or when a program has come to the end of its job, etc.

Listing 1 is a listing of a short BASIC program that packs a 24 byte machine language program into S$. The BASIC portion of the program controls the flow and execution, the machine language contributes only the beep. The audio signal is output to the plug, which normally goes to the AUX jack in the cassette recorder. This plug must connect with an audio amplifier, such as a Radio Shack Microsonic Speaker Amplifier (277-1008).

Table 1 gives a detailed description of the packing of the machine language routine. Note that it is executed only once, not each time that a prompt is sounded.

The Program

Lines 120–210 (Listing 1) set up a prompt for keyboard input. If no entry is made, a prompt sounds every four seconds. If an incorrect entry is made, the prompt sounds instantly.

Line 150 jumps to the packed subroutine, which generates the sound and then returns to BASIC at the next line. If the correct entry is made, the program

```
10 POKE 16553,255         'FIXES ROM DATA READ BUG
20 S$="THIS STRING LENGTH IS 24"
30 I=VARPTR(S$)
40 JJ=PEEK(I+2)*256+PEEK(I+1)
50 IF JJ>32767 THEN JJ=-1*(65536-JJ)
60 FOR P=JJ TO JJ+23
70 READ D:POKE P,D:NEXT
80 DATA 205,127,10,14,60,65,62,01,
        211,255,16,254,65,60,
        211,255,16,254,43,124,181,32,238,201
90 POKE 16526,PEEK(I+1):POKE 16527,PEEK(I+2)
100 '
110 '
120 CLS
130 PRINT@0,"ENTER 'Y' OR 'N' ?"
140 IF DE<100 GOTO 170
150 X=USR(100)                'I BEEP HERE
160 DE=0
170 Z$=INKEY$
180 IF Z$="" THEN DE=DE+1:GOTO 130
190 PRINTZ$
200 IF Z$<>"Y" AND Z$<>"N" THEN 150
210 PRINT"THANK YOU !"
220 FOR DE=1 TO 500:NEXT
300 PRINT"I AM SIMULATING BEING HARD AT WORK ON A PROGR
AM.
310 PRINT"I WILL TELL YOU WHEN I HAVE FINISHED BY 'BEEP
ING'.
320 PRINT"YOU CAN STOP THE BEEP AND RETURN TO THE LEVEL
 II
330 PRINT">READY   BY PRESSING ANY KEY.
340 FOR DE=1 TO 5000:NEXT
350 X=USR(200)                    'I BEEP HERE
360 FOR DE=1 TO 500
370 A$=INKEY$
380 IF A$<>"" THEN 410
390 NEXT
400 GOTO 350
410 END
```

Program Listing 1

Line	Description
10	Fixes ROM bug for reading DATA statements
20	Defines string of sufficient length into which machine language will be packed
30	Finds address of S$ information block
40	Computes decimal address of location of S$
50	Adjustment for RAM sizes up to 16K
60	Sets up loop
70	POKEs DATA into string
80	Machine language code in decimal
90	POKEs address of S$ into USR area, so that BASIC program can find it when USR call is made

Table 1. Subroutine Initializer

continues to line 210.

Lines 300–410 could, as in this case, be put at the end of any program to announce an audible end of job. The beep continues every few seconds until any key is depressed.

Once the program is working, any attempt to LIST it causes the screen to roll and looks somewhat like a bad load from tape. This is because BASIC is attempting to print something which it does not normally see, namely machine language in S$.

At this point, you should CSAVE the program, because we are going to delete part of it to save RAM. After the program has run successfully, the DATA in line 80 are obviously packed into S$ as a working machine language subroutine, so lines 40 through 80 are redundant and can be deleted.

When we CLOAD the program again, the subroutine will be in S$, just as it is now and needn't be packed again. After deleting lines 40 through 80, CSAVE the program again (see Restrictions).

The argument in the USR command determines the length of the tone. Larger values (up to 32767) cause longer beeps. A value of more than 1000 or so causes a very long tone.

The pitch is controlled by the fifth DATA element in line 80. In the example it is 60. Larger values give a lower pitch. The value must not exceed 255.

Restrictions

1. DATA elements to be packed must not have the value 0 or 34. Either terminates the string when packed.

2. If lines 40 through 80 are deleted after the string is packed, certain conditions can cause the program to bomb; if you modify the BASIC program, for example. This can be avoided by not deleting the string packing code until a final version of the program is produced, or by not deleting the lines at all, if RAM use is not a consideration. ■

A look at Tandy's talker.

Eloquent Eighties

TRS-80 Voice Synthesizer
Radio Shack
Tandy Corporation
Ft. Worth, TX 76102
$399

Jim Wright
10140 N.W. 43rd St.
Coral Springs, FL 33065

It's magic, absolute magic! A 12" × 6" × 4" box of capacitors, resistors, transistors and integrated circuits is talking to me in plain English. Not only is it talking, but it says anything that I ask it to, including some very funny sound effects.

I suppose my engineering background and our nation's ability to put men on the moon should have prepared me for this. Nonetheless, I am still amazed at the marvels in our growing level of consumer technology.

Loading the Demo Tape

I picked up one of Radio Shack's Voice Synthesizers, in spite of its $399 price tag.

The Voice Synthesizer is a completely self-contained unit. Like its host, the TRS-80, it uses a separate in-line step-down transformer. The synthesizer can be used on any Level I or II TRS-80 and attaches by a ribbon cable and 40-pin connector expansion port.

For those of you who have the expansion interface, the synthesizer connects to your parallel interface port on the left side of the unit. Once connected, you turn on the synthesizer, adjust the volume and begin programming.

Radio Shack was thoughtful enough to include a demonstration cassette tape with the synthesizer. Unfortunately, the manual I received made no mention of the cassette! However, by increasing the tape player's volume level to about 6.5, the cassette finally loaded.

After typing RUN, the TRS-80 started talking! The program takes several minutes, during which time you get a good idea of what synthesized speech sounds like, as well as how it is structured in a program. While the audio output is quite intelligible, the synthesizer produces a stereotype computer sound much like the Battle Star Galactica's Cylons.

Using the Synthesizer

While the manual that accompanies the synthesizer contains only 13 pages, it provides adequate information for anyone who can program in BASIC to be able to use the synthesizer. Radio Shack's approach to programming their synthesizer is interesting.

First of all, they do all the programming in BASIC. Although this may not be the most efficient method, it certainly makes using the synthesizer very simple.

Secondly, they used 32 bytes of the video monitor's memory map to store the synthesized voice message. The storage is only temporary, in that the synthesizer contains a 32-byte buffer that is filled from the lower portion of the video monitor's memory. Once the synthesizer's buffer is loaded, the memory can be erased or written over. This is done so rapidly that you can't even see what is happening on the monitor.

After the voice programming has been transferred to the synthesizer, the TRS-80 continues with its program, while the synthesizer is speaking. This allows for some very exciting graphics. For example, the rockets may fizzle as your Lunar Lander is in motion. Or a spelling program can draw a word as the synthesizer pronounces it. Business programs can verbally explain sales, as the computer draws a graph.

How does the synthesizer know when it has a message to relate? Based on the program description, it appears that the voice synthesizer acts as a 32-byte block of memory paralleling the video monitor locations 16352D to 16383D. This block, referred to as a "window" in the instruction manual, is opened and closed by the control character, ?.

When a ? appears in memory locations 16352D to 16383D, the synthesizer opens or closes its window. An LED on the synthesizer lets you know if the window is open or closed.

Once the window is opened, any character or characters appearing between the window locations is copied into the synthesizer's buffer, and the synthesizer produces the character's phonetic sounds.

Maximum message length is 32 characters at a time. Depending on the words used, 32 characters produces from one to five seconds of speech. If you desire a longer message, you can string together several groups of characters, separated by a software time delay, allowing the first segment of speech to be completed before the second begins.

This sequencing can be done

```
5 REM *** SET UP PROMPTING PHRASE "PLEASE ENTER YES OR NO" ***
10 A$ = "PL.EZ 3NTR Y43SS D/ NBOU"
15 REM *** PLACE  PHRASE IN SYNTHESIZER ***
20 PRINT@992, "? ";A$;" ?";
25 REM *** CLEAR SCREEN "WINDOW" ***
30 PRINT@992, "                                        ";
```

Program Listing 1.

indefinitely to produce as long a message as your memory allows. Based on an average of three seconds for each group of 32 characters, 1K of memory produces approximately one and a half minutes of speech. This can be increased substantially by using a "dictionary array" of commonly used words.

Add Sound to Your Character

In printed text, a single symbol is called an ASCII character. The "characters" used in the spoken message are called phonemes (fō-nēms). A phoneme is a single unit of sound. It is the combination of phonemes that produces the words the voice synthesizer pronounces.

The TRS-80 Voice Synthesizer contains 62 phonemes that can be combined to pronounce virtually any word in the English language, as well as some foreign words and phrases.

The correct programming method for the synthesizer is to type the ASCII character that represents the phonetic sound desired, for example: TEE ;8R3SS @•&DY VO85&SS SIN = 85;&Z/ is TRS-80 Voice Synthesizer.

The manual includes a table that cross references the phoneme symbol and its ASCII character. The manual suggests that you sound out the word you are trying to program and then select the correct phonemes to use. I have found that this works quite well but still requires some trial and error experimentation.

While the manual does provide a dictionary, with correct phoneme constructions, it includes only 34 words.

You can also cross-reference any dictionary's phonetic symbols with the manual's phoneme symbols/ASCII characters.

Using a Dictionary

I found some phonetic symbols that could not be referenced as single phonemes. These sounds, such as the long *i*, are represented as diphthongs, a combination of vowel phonemes. With a little experimentation, you soon find that by changing or adding a single phoneme you can also create different inflections, such as the word beer, which can be coded BEER or BEIR.

BASIC Programming

Program Listing 1 demonstrates how simple it is to add verbal responses to a BASIC program. It prompts you to enter yes or no. To change the message, it is necessary to change only the A$. Lines 20 and 30 can be a called subroutine using various A$ messages.

Program Listing 2 can be used to experiment with different phoneme structures. Be careful to keep the number of phonemes to less than 28.

If you should type more than 28, the LED on the synthesizer stays lit. This is because the second ? was truncated from the buffer, and the synthesizer's window was not closed.

If this happens, just press the

```
10 CLS
20 PRINT@404, "ENTER PHONETIC PHRASE"
30 PRINT@464, "------------------------------"
40 PRINT@464,;
50 INPUT A$
60 GOSUB 100
70 GOTO 10
100 PRINT@992, "? ";A$;" ?";
110 PRINT@992, "                                        ";
120 RETURN
```

Program Listing 2.

? key and ENTER, and the window closes. The reason for this is that the open window sees the ? as a toggle off-control character. Remember that the first ? toggles the window open and a second toggles the window closed. There are several other examples of useful subroutines included in the synthesizer's manual.

A list of potential applications is included in the manual. These include:

- Computer-Aided Instruction
- Intrusion – Fire Alarm
- Games
- Talking Clock
- Blind User's Terminal
- Verbally Impaired Prosthetic Aid
- Home Environment Audio Response (I programmed mine to ask for a beer!)
- Computer – Phone Voice Interface

But once you've used the Synthesizer, you're going to be intrigued with its other possibilities. How about a verbal calculator? Or a typing instruction program?

Finding fault with the TRS-80 synthesizer is like telling Henry Ford that his first automobile rode rough. True, it is a marvelous invention, but several areas can be improved. The first is the monotone computer sound. It would be nice to have an option for changing the pitch.

Secondly, some control is needed over the speed. I can envision a version of the synthesizer that will be able to sing, as well as speak.

Several phonemes do not really sound as one would expect. The *g* sounds almost identical to the *d* and causes words like gig to sound more like *deed*. The *L* has what I call an Elmer Fudd sound, that is, a soft type *L*, rather than a hard *L*. The *K* also has a tendency to sound a little soft.

While some of these sounds are not perfect, the synthesized voice for the most part is very intelligible. I'd recommend the synthesizer for most any application where an audible prompt is desired. ■

✓Reader Service—see page 226

A tutorial trip through addition and subtraction.

Math Flash

Jim Barbarello
RD #1, Box 241H
Tennent Rd.
Englishtown, NJ 07726

Traditional math flash cards have always been a very useful tool for young students in learning math fundamentals. The math flash card program here helps the student practice and learn on his own without a tutor. The program creates all the requisites of a conventional math drill: random addition or subtraction problems presented with bold, easily readable numerals; a set response time limit; positive reinforcement for correct answers, and a chance to redo an incorrectly answered problem.

For each problem, two variables between 0 and 49 are selected. These variables are either added or subtracted on a random basis and then presented as the problem. The variables' upper limit of 49 restricts the answer to a two digit value.

After each set of nine problems is completed, an interim score is presented. If the student chooses to continue, the next set of nine problems is presented. If the student chooses not to continue, a total score for all problems attempted is provided.

The variables are selected so as to minimize repetition of problems and to maximize exposure to all possibilities. If, for example, the student is encountering specific difficulties, the program can be easily modified to present a specific set of problems. The program presents up to 81 different problems per run.

A Typical RUN

When the program is RUN, the message "GET READY—HERE WE GO!" appears as the random variables are selected. The screen then clears, and the first problem is displayed (ex: 14 + 23 =) in two-inch high numerals. The student has approximately four seconds to enter an answer. The entered answer is displayed to the right of the equals sign in normal height (¼ inch) numerals. Since the answer is entered via the INKEY$ function, the student need only press the appropriate numeric keys to input an answer.

A correct answer causes the reinforcement message "** VERY GOOD**" to appear above the problem. If the student's answer is incorrect, it is removed from the screen and the message "XX IS WRONG. THE RIGHT ANSWER IS:" is displayed in the upper left hand corner of the screen.

At the same time, the correct answer is displayed to the right of the equals sign in bold (two-inch high) numerals. In either event, the problem remains on the screen for an additional three seconds. A correct answer allows a new problem to be displayed. For an incorrect answer, the same problem is repeated, giving the student another chance to answer correctly.

If the student does not respond to any problem in the allotted four seconds, the message "TIME'S UP" appears in the upper left hand corner of the screen and the correct answer is displayed in bold numerals. A "TIME'S UP" situation is treated as an incorrect answer.

If the student does not provide the correct answer in two tries, an incorrect response is scored. At the end of nine problems, the screen clears and the message "YOU GOT X RIGHT OUT OF 9 TRIES THIS TIME. WANT TO TRY AGAIN (YES OR NO)" is displayed.

If the student responds with lems, the screen clears and the message "YOU GOT X RIGHT OUT OF 9 TRIES THIS TIME. WANT TO TRY AGAIN (YES OR NO)" is displayed.

About The Program

Fully half of the program consists of subroutines to create the bold numerals, plus, minus and equals signs (Lines 1000 through 2200). These subroutines use the variable S to position the numerals on the screen. S is initially set to 15682 in Line 65 and is continually readjusted during the program to allow the problem to be serially presented (one character after another). The program automatically adjusts for single or double digit variables/answers so that gaps are not left in the presentation.

The program can be easily customized to a specific student's needs in a number of areas. For the very young student, the variables can be limited to a single digit (0 to 9) by changing the statement A(K) =

RND(49) in line 10 to A(K) = RND (9) and B(K) = RND(49) in line 30 to B(K) = RND(9).

The response time allowed for each problem can be adjusted by varying the FOR-NEXT loop in line 140. As shown, the keyboard is scanned for a response 25 times (FOR M = 1 TO 25). Increasing the loop increases the allowable response time—decreasing it shortens the time allowed.

If a specific set of problems is desired instead of the presently available random selection, lines 10 through 40 can be deleted and replaced with:

```
10 DATA A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P,
   Q,R
20 FOR I = 1 TO 9: READ A(I):READ B(I):
   NEXT I
```

(with A through R being numbers you select)

With this modification, A,C,E,G,I,K,M,O and Q will be the numerals of the first variable and B,D,F,H,J,L,N,P and R will be the numerals of the second variable. For example, if A = 19 would be 19 + 43 (or 43 − 19 if subtraction was randomly selected). If only addition or only subtraction is desired, lines 60 and 70 may be deleted and replaced with 60 C = A + B (for addition, or) 60 C = A − B (for subtraction).

It is possible to multiply and/or divide. However, special care must be taken such that the variable produce an integer answer that is no larger than 99. In addition, appropriate subroutines for the times (×) and divide (÷) signs would need to be added. ■

Program Listing.

```
1 REM****************************************************************
     ********************
2 REM*****
                         *****
3 REM*****                              M A T H   F L A S H   C A R D S
                         *****
4 REM*****
                         *****
5 REM*****                                        A PROGRAM BY
                         *****
6 REM*****                                       JIM BARBARELLO
                         *****
7 REM*****
                         *****
8 REM****************************************************************
     ********************
9 REM
10 CLS:PRINT"GET READY - - HERE WE GO!":RANDOM:FOR K=1 TO 9:A(K)=
     RND(49)
20 FOR J=0 TO K-1:IF A(K)=A(J) THEN K=K-1:NEXT K ELSE NEXT J,K
30 FOR K=1 TO 9:B(K)=RND(49)
40 FOR J=0 TO K-1:IF B(K)=B(J) THEN K=K-1:NEXT K ELSE NEXT J,K
50 FOR K=1 TO 9
60 FOR J=1 TO 9:IF CT=0 THEN D=RND(2)
65 A$="":B$="":S=15682:CLS:A=A(K):B=B(J):C=0:F=0
70 IF D=1 THEN C=A+B ELSE C=A-B
75 IF C<0 THEN A=B(J):B=A(K):C=A-B
80 IF A>9 THEN W=INT(A/10):Y=A-W*10 ELSE Y=A:S=S-8
85 ON W GOSUB 1000,1100,1200,1300,1400,1500,1600,1700,1800
90 S=S+8
95 ON Y GOSUB 1000,1100 [illegible]400,1500,1600,1700,1800
100 IF Y=0 THEN GOSUB 1[illegible]00
105 ON D GOSUB 2000,210[illegible]
110 S=S+16:IF B>9 THEN W=INT(B/10):ON W GOSUB 1000,1100,1200,1300
     ,1400,1500,1600,1700,1800
115 IF B>9 THEN Y=B-W*10 ELSE Y=B:S=S-8
120 S=S+8:ON Y  GOSUB 1000,1100,1200,1300,1400,1500,1600,1700,180
     0
125 IF Y=0 THEN GOSUB 1900
130 S=S-17:GOSUB 2200:S=S-1:W=0:Y=0:A$=INKEY$
140 FOR M=1 TO 25
150 A$=INKEY$
160 IF A$="" THEN 170 ELSE 200
170 FOR L=1 TO 50:NEXT L,M
180 PRINT@ 498,"  ";:PRINT@ 0,"TIME'S UP"
190 CT = CT+1:GOTO 240
200 IF C>9 AND B$="" THEN B$=A$:M=5:PRINT@ 498,B$;:NEXT M:ELSE PR
     INT@ 499,A$;
210 G=VAL(A$)+VAL(B$)*10:IF G=C THEN CT=0:PRINT@ 84,"** VERY GOOD
     **";:GOTO 290
220 IF G<>C THEN PRINT[illegible],G;" IS WRONG. THE RIGHT ANSWER IS":CT=C
     T+1:PRINT@498,"  ";:GOTO 240
230 CT=0
240 S=S+36
250 IF C=0 THEN GOSUB 1900:GOTO 290
260 IF C>9 THEN W=INT(C/10):Y=C-W*10 ELSE Y=C:S=S-8
270 ON W GOSUB 1000,1100,1200,1300,1400,1500,1600,1700,1800
275 S=S+8
280 ON Y GOSUB 1000,1100,1200,1300,1400,1500,1600,1700,1800
285 IF Y=0 THEN GOSUB 1900
290 FOR N=1 TO 750:NEXT N
300 IF CT=0 THEN TTL=TTL+1:NEXT J:GOTO 330
310 IF CT<2 THEN J=J-1:NEXT J:GOTO 330
320 CT=0:NEXT J:GOTO 330
330 CLS:PRINT"YOU GOT";TTL;" RIGHT OUT OF 9 TRIES THIS TIME":TL=T
     L+TTL:TTL=0
340 PRINT:INPUT"WANT TO TRY AGAIN (YES OR NO)";BL$
350 IF LEFT$(BL$,1)="Y" THEN NEXT K
360 IF LEFT$(BL$,1)="N" THEN CLS:PRINT"YOUR TOTAL IS";TL;" RIGHT
     OUT OF";K*9;"TRIES. BYE!":END
370 GOTO 340
1000 REM**  ONE
1010 FOR I=0 TO 4
1020 POKE S+I*64,191:POKE S+1+I*64,191:NEXT
1030 POKE S+62,191:FOR I=1 TO 6:POKE S+253+I,191:NEXT
1040 RETURN
1100 REM**  TWO
1110 FOR I=-2 TO 3
1120 POKE S+I,191:POKE S+128+I,191:POKE S+256+I,191:NEXT
1130 POKE S+67,191:POKE S+190,191
1140 RETURN
1200 REM**  THREE
1210 FOR I=-2 TO 3
1220 POKE S+I,191:POKE S+128+I,191:POKE S+256+I,191:NEXT
1230 POKE S+67,191:POKE S+195,191
1240 RETURN
1300 REM**  FOUR
1310 POKE S+128,191:POKE S+129,191
1320 FOR I=0 TO 4:POKE S+2+I*64,191:POKE S+3+I*64,191:NEXT
1330 FOR I=0 TO 2:POKE S-2+I*64,191:POKE S-1+I*64,191:NEXT
1340 RETURN
1400 REM**  FIVE
1410 FOR I=-2 TO 3
1420 POKE S+I,191:POKE S+128+I,191:POKE S+256+I,191:NEXT
1430 POKE S+62,191:POKE S+195,191
1440 RETURN
1500 REM**  SIX
1510 FOR I=-2 TO 3
1520 POKE S+I,191:POKE S+128+I,191:POKE S+256+I,191:NEXT
1530 POKE S+62,191:POKE S+190,191:POKE S+195,191
1540 RETURN
1600 REM**  SEVEN
1610 FOR I=-2 TO 3:POKE S+I,191:NEXT
1620 FOR I=0 TO 4:POKE S+2+I*64,191:POKE S+3+I*64,191:NEXT
1630 RETURN
1700 REM**  EIGHT
1710 FOR I=-2 TO 3
1720 POKE S+I,191:POKE S+128+I,191:POKE S+256+I,191:NEXT
1730 POKE S+62,191:POKE S+67,191:POKE S+190,191:POKE S+195,191
1740 RETURN
1800 REM**  NINE
1810 FOR I=-2 TO 3
1820 POKE S+I,191:POKE S+128+I,191:POKE S+256+I,191:NEXT
1830 FOR I=0 TO 4:POKE S+3+I*64,191:NEXT
1840 POKE S+62,191
1850 RETURN
1900 REM**  ZERO
1910 FOR I=-2 TO 3
1920 POKE S+I,191:POKE S+256+I,191:NEXT
1930 FOR I=0 TO 4:POKE S-2+I*64,191:POKE S+3+I*64,191:NEXT
1940 RETURN
2000 REM**  PLUS SIGN
2010 FOR I=1 TO 3
2020 POKE S+8+I*64,191:POKE S+9+I*64,191:NEXT
2030 FOR I=0 TO 5:POKE S+134+I,191:NEXT
2040 RETURN
2100 REM**  MINUS SIGN
2110 FOR I=0 TO 5:POKE S+134+I,191:NEXT
2120 RETURN
2200 REM**  EQUALS SIGN
2210 FOR I=0 TO 5:POKE S+88+I,191:POKE S+216+I,191:NEXT
2220 RETURN
```

An M.D.'s bookkeeping system.

Doctor Your Records

Wilbur A. Muehlig, M.D.
726 N. 91 Plaza, #305
Omaha, NE 68114

When I was practicing medicine, my secretary kept a "daybook," entering each payment made by a patient, adding totals for the day, the month and, finally, the year.

When I left private practice, I continued the same general method of keeping track of income checks, but my usual practice was to let the arithmetic go until the end of the year when my tax was due. At this time totaling my income was a rather discouraging job, but ideally suited to a computer.

This bookkeeping program should be of interest to anyone owning a few stocks and bonds. It was written with the retired person in mind, and therefore includes Social Security income and annuity payments.

The program, which handles up to 35 entries a month, accepts eight types of entries: earned income, dividends, interest, annuity funds, tax-free interest, Social Security, life insurance dividends, and miscellaneous items. The first four are taxable, the next three untaxable and the last is for your records only.

The program is written in Level II BASIC for a TRS-80 with 32K and a single disk drive. But it can be easily modified to use a cassette. The printer has a line length of 61 characters, but can be compressed to fit into 40 columns. Though the program can be used without a printer, much of its benefit is lost.

First Steps

Choose MAKE A FILE ENTRY (number three in Table 1) when first using the program. Enter the day, the month and the source of the income; choose the type of entry, the amount (Table 2), and press enter.

When you have finished making your entries, save them to disk using the first menu option and the disk option—SAVE MONTH'S FILE TO DISK (one in Table 1). Before adding more entries, load the previous ones with disk option two.

You can use menu options

```
WHAT DO YOU WANT TO DO?

    1-DISK OPERATIONS
    2-COMPUTATIONS
    3-MAKE A FILE ENTRY
    4-CORRECT AN ENTRY
    5-DELETE AN ENTRY
    6-SEE A PRINTOUT ON VIDEO
    7-MAKE A HARD COPY
    8-LEAVE THE PROGRAM

CHOOSE? _1


WHAT DO YOU WANT TO DO?

    1-SAVE MONTH'S FILE TO DISK
    2-LOAD MONTH'S FILE FROM DISK
    3-SAVE YEARLY TOTALS TO DATE
    4-LOAD YEARLY TOTALS
    5-RETURN TO MENU

CHOOSE? 1_
```

Table 1

```
 1
TO END FILE ENTRIES, TYPE 99

DAY OF MONTH? 5

SOURCE OF INCOME (NOT OVER 24 SPACES)
? AT & T
WHAT TYPE OF ENTRY?
1—EARNED INCOME   5—TAX FREE INTEREST
2—DIVIDEND        6—SOCIAL SECURITY
3—INTEREST        7—LIFE INSURANCE DIVIDEND
4—ANNUITY         8—MISCELLANEOUS ITEM

CHOOSE? 2

AMOUNT OF DIVIDEND? 345.67_

1  DATE 5   AT & T DIVIDEND $ 345.67

IF THE ABOVE IS INCORRECT, TYPE 1. IF THE PART THEN DISPLAYED IS COR-
RECT, PRESS ENTER, OTHERWISE TYPE THE CORRECTION.

NOTE: IF THE ERROR IS IN THE TYPE OF INCOME, DELETE THE WHOLE ITEM
AND REENTER.

IF THE ABOVE IS CORRECT, TYPE 2 TO GO ON?_
```

Table 2

Program Listing

```
1 REM * PERSONAL INCOME LEDGER PROGRAM *
2 REM * WILBUR A. MUEHLIG, M.D. *
3 REM * 726 N. 91 PLAZA, APT. 305,
4 REM * OMAHA, NE 68114
11 REM * OCT. 1979.
12 CLS
13 PRINT:PRINT:PRINT:PRINT:PRINT:PRINT:PRINT
15 PRINTTAB(23)"INCOME LEDGER PROGRAM":PRINT:PRINT:PRIN
T
20 INPUT "DO YOU WANT INSTRUCTIONS FOR USE (Y/N)";Z$
30 IF Z$="Y" GOTO  4000
40 CLEAR 500:DEFDBL E,D,I,A,T,G,L,C,F,R,Q:DEFINT B,J,K,
H,X,Z,P:P1=1
45 DEFSTR S,M,V,Y,U
50 DIM S(35),E(25),DA(35),D(25),I(25),A(25),T(25),G(25)
,L(35),C(25),F(35),R(35),M(12)
52 DATA JAN,FEB,MAR,APR,MAY,JUN,JUL,AUG,SEP,OCT,NOV,DEC

53 FOR H=1 TO 12:READ M$(H):NEXT
55 CLS:PRINT:PRINT:PRINT:PRINT
60 INPUT"YEAR OF THIS REPORT (####)";Y$
80 PRINT:INPUT"NUMBER OF THIS MONTH (1 TO 12)";H
100 CLS:PRINT
110 PRINT"WHAT DO YOU WANT TO DO?

                  1-DISK OPERATIONS
                  2-COMPUTATIONS"
112 PRINT"                3-MAKE A FILE ENTRY
                  4-CORRECT AN ENTRY
                  5-DELETE AN ENTRY
                  6-SEE A PRINTOUT ON VIDEO"
113 PRINT"                7-MAKE A HARD COPY
                  8-LEAVE THE PROGRAM"
114 PRINT:INPUT"CHOOSE";J
120 ON J GOTO 200,1262,600,800,1000,1225,1400,1600
200 CLS:PRINT:PRINT:PRINT:PRINT:PRINT"WHAT DO YOU WANT
TO DO?

                1-SAVE MONTH'S FILE TO DISK
                2-LOAD MONTH'S FILE FROM DISK
                3-SAVE YEARLY TOTALS TO DATE"
210 PRINT"              4-LOAD YEARLY TOTALS
                5-RETURN TO MENU"
220 PRINT:INPUT"CHOOSE";J
230 ON J GOTO 240,270,299,349,235
235 GOTO100
240 IF S(1)="" THEN PRINT"FILE EMPTY.":INPUT"PRESS ENTE
R";Z$:GOTO200
241 OPEN"O",1,M(H)
242 PRINT"SAVING MONTHLY FILE TO DISK....."
245 PRINT#1,P1
250 FOR X=1 TO P1-1
255 PRINT#1,S(X):PRINT#1,X;DA(X);E(X);D(X);I(X);A(X);T(
X);G(X);F(X);C(X)
260 NEXT:CLOSE
265 PRINT:PRINT"MONTHLY FILE SAVED TO DISK":FOR Z=1 TO
500:NEXT:GOTO 100
270 ON ERROR GOTO 1800
272 OPEN"I",1,M(H)
275 INPUT#1,P1
280 FOR X=1 TO P1-1
285 INPUT#1,S(X):INPUT#1,X,DA(X),E(X),D(X),I(X),A(X),T(
X),G(X),F(X),C(X)
287 PRINTX;S(X)
290 NEXT:CLOSE
291 FOR X=1 TO P1-1:L(X)=E(X)+D(X)+I(X)+A(X):R(X)=T(X)+
G(X)+F(X)+C(X):NEXT
295 PRINT:PRINT"MONTHLY FILE LOADED.";:INPUT"    PRESS
ENTER";Z$:GOTO 100
299 IF ES=0 AND DS=0 AND IS=0 AND AS=0 AND TS=0 AND GS=
0 AND FS=0 AND CS=0 THEN INPUT"FILE EMPTY.  PRESS ENTER
.";Z$:GOTO200X
300 V$=M(H)+Y$:OPEN"O",2,V$
305 PRINT:PRINT"SAVING YEARLY TOTALS TO DISK..."
320 PRINT#2,ES;DS;IS;AS;TS;GS;FS:CLOSE
330 GOTO 100
349 ON ERROR GOTO 1800
350 V$=M(H-1)+Y$
355 OPEN"I",2,V$
360 PRINT:PRINT"LOADING YEARLY TOTALS...."
370 INPUT#2,ES,DS,IS,AS,TS,GS,FS:CLOSE
380 GOTO 100
599 REM * TO MAKE A FILE ENTRY *
600 CLS:PRINT:PRINT:PRINT
602 FOR X = P1 TO 50
610 CLS:PRINT:PRINT:PRINT
612 PRINTX
613 PRINT:PRINT"TO END FILE ENTRIES, TYPE 99"
615 PRINT:IF DA(X) <> 0 PRINT DA(X);" ";
620 INPUT"DAY OF MONTH";DA(X)
622 IF DA(X)=99 THEN P1=X:GOTO 100
630 IF S(X)<>"" PRINTS(X);"   ";
640 PRINT:PRINT"SOURCE OF INCOME (NOT OVER 24 SPACES)":
INPUT S(X)
645 IF LEN(S(X)) > 24 PRINT"TOO LONG. TRY AGAIN.":GOTO6
40
660 PRINT"WHAT TYPE OF ENTRY?
1-EARNED INCOME          5-TAX FREE INTEREST
2-DIVIDEND               6-SOCIAL SECURITY
3-INTEREST               7-LIFE INSURANCE DIVIDEND
4-ANNUITY                8-MISCELLANEOUS ITEM
662 ON ERROR GOTO 1700
665 PRINT:INPUT"CHOOSE";B
670 ON B GOTO  6100,6200,6300,6400,6500,6600,6700,6800
680 CLS:PRINT:PRINT
690 PRINT:PRINTX;" DATE";DA(X);"  ";S(X)"  ";
691 IF E(X)>0 PRINT"EARNED INCOME   ";:PRINTUSING"$###,
##.##";E(X)
692 IF D(X)>0 PRINT"DIVIDEND   ";:PRINTUSING"$###,##.##
";D(X)
693 IF I(X)>0 PRINT"INTEREST   ";:PRINTUSING"$###,##.##
";I(X)
694 IF A(X)>0 PRINT"ANNUITY   ";:PRINTUSING"$###,##.##"
;A(X)
695 IF T(X)>0 PRINT"TAX FREE INTEREST   ";:PRINTUSING"$
###,##.##";T(X)
696 IF G(X)>0 PRINT"SOCIAL SECURITY   ";:PRINTUSING"$##
#,##.##";G(X)
697 IF F(X)>0 PRINT"LIFE INS. DIV.   ";:PRINTUSING"$###
,##.##";F(X)
698 IF C(X)>0 PRINT"MISC. DEPOSIT   ";:PRINTUSING"$###,
##.##";C(X)

700 PRINT:PRINT"IF THE ABOVE IS INCORRECT, TYPE 1.  IF
THE PART THEN DISPLAYED"
705 PRINT"IS CORRECT, PRESS ENTER, OTHERWISE TYPE THE C
ORRECTION."
707 PRINT:PRINT"NOTE: IF THE ERROR IS IN THE TYPE OF IN
COME, DELETE THE WHOLE"
708 PRINT"ITEM AND REENTER."
709 REM * INCOME TYPE ERROR CAN CAUSE CONFUSION IF NOT
DELETED *
710 PRINT:INPUT"IF THE ABOVE IS CORRECT, TYPE 2 TO GO O
N";Z
720 IF Z=1 GOTO 610
730 IF Z<>2 GOTO 710
740 NEXT:GOTO100
799 REM * USES SAME CORRECTION SECTION AS FILE ENTRY *
800 CLS:PRINT:PRINT:PRINT:PRINT
810 PRINT"WHICH NUMBER DO YOU WISH TO CORRECT"
815 PRINT:PRINT"             (0 TO RETURN TO MENU)
   ";:INPUT K
817 IF K=0 GOTO 100
820 FOR X=1 TO P1-1
830 IFK=X GOTO 680
840 NEXT
850 GOTO 100
999 REM * DELETES ENTRY AND RENUMBERS *
1000 PRINT"WHICH NUMBER DO YOU WANT TO DELETE?"
1002 PRINT"      (0 TO RETURN TO MENU)             ";:INP
UT X
1005 IF X=0 GOTO100
1010 DA(X)=0:S(X)="":E(X)=0:D(X)=0:I(X)=0:A(X)=0:T(X)=0
:G(X)=0:F(X)=0:C(X)=0:L(X)=0:R(X)=0
1020 FOR X=X+1 TO P1:DA(X-1)=DA(X):S(X-1)=S(X):E(X-1)=E
(X):D(X-1)=D(X):I(X-1)=I(X):A(X-1)=A(X):T(X-1)=T(X):G(X
-1)=G(X):F(X-1)=F(X):C(X-1)=C(X):L(X-1)=L(X):R(X-1)=R(X
)
1025 NEXT
1030 P1=P1-1
1040 PRINT:PRINT"ENTRY DELETED.":FOR Z=1 TO 700:NEXT:GO
TO 100
1200 CLS:PRINTTAB(16)"INCOME LEDGER FOR ";M(H);" ";Y$:P
RINT
1210 PRINT"## DAY  SOURCE OF INCOME              TAXABLE
 NONTAX."
1220 PRINT STRING$(54,45):RETURN
1225 GOSUB 1200   REM * ALLOWS REPRINTING OF HEADING ON
 EACH
PAGE OF VIDEO *
1230 FOR X = 1 TO P1-1
1235 IF X/11=INT(X/11) PRINT:INPUT"PRESS ENTER TO CONTI
NUE";Z$:CLS:PRINT:GOSUB 1200
1240 IF R(X)=0 PRINT USING"## ##  \
  \  $###,##.##";X;DA(X);S(X);L(X)
1250 IF L(X)=0 PRINT USING"## ##  \
     \            $###,##.##";X;DA(X);S(X);R(X)
1260 NEXT:PRINT:INPUT"ENTER 1 TO RETURN TO MENU, 2 TO S
EE YEARLY TOTALS.";Z:IF Z=1 GOTO 100 ELSE 1279
1262 CLS:PRINT:PRINT:PRINT:PRINT"THE FOLLOWING COMPUTES
 INCOME TOTALS FOR THE MONTH AND FOR"
1263 PRINT"THE YEAR.  THE MONTH'S ENTRIES SHOULD BE COM
PLETE, YOUR PRINTER"
1264 PRINT"READY, AND THE YEARLY TOTALS INPUT FROM DISK
 BEFORE CONTINUING."

1265 PRINT:PRINT"TO RETURN TO MENU, TYPE 1, TO CONTINUE
 WITH THE COMPUTATION,"
1266 PRINT"TYPE 2.":PRINT:INPUT Z
1267  IF Z=1 GOTO 100
1268 IF Z<>2 GOTO 1262
1270 GOSUB 7000
1279 CLS
1280 PRINT"   TOTALS FOR ";M(H);" ";Y$;"         MONTH
    YEAR TO DATE"
1282 PRINT
1285 PRINT"TOTAL EARNED INCOME          ";:PRINTUSING"$##
##,###.##  $####,###.##";ET,ES
```

Program continues

```
1290 PRINT"TOTAL DIVIDENDS              ";:PRINTUSING"$##
##,###.##  $#####,###.##";DT,DS
1300 PRINT"TOTAL INTEREST               ";:PRINTUSING"$##
##,###.##  $#####,###.##";IT,IS
1310 PRINT"TOTAL ANNUITY FUNDS          ";:PRINTUSING"$##
##,###.##  $#####,###.##";AT,AS
1320 PRINT"TOTAL TAX FREE INTEREST      ";:PRINTUSING"$##
##,###.##  $#####,###.##";TT,TS
1330 PRINT"SOCIAL SECURITY              ";:PRINTUSING"$##
##,###.##  $#####,###.##";GT,GS
1340 PRINT"TOTAL LIFE INS. DIV.         ";:PRINTUSING"$##
##,###.##  $#####,###.##";FT,FS
1355 PRINT:U$="$#####,###.##"
1360 PRINT"TOTAL TAXABLE INCOME TO DATE          ";:PRIN
TUSINGU$;LS
1370 PRINT"TOTAL NONTAX. INCOME TO DATE          ";:PRIN
TUSINGU$;RS
1380 PRINT"TOTAL INCOME TO DATE                  ";:PRIN
TUSINGU$;Q
1390 PRINT:INPUT"PRESS ENTER TO CONTINUE";Z$:GOTO 100
1400 CLS:PRINT:PRINT:PRINT:PRINT:PRINT:PRINT"IF YOUR PR
INTER ISN'T READY, ENTER 1 TO RETURN TO MENU,"
1402 PRINT:INPUT"          OTHERWISE ENTER 2.";Z:IF Z=1
GOTO 100
1405 CLS:LPRINTTAB(32)"INCOME LEDGER FOR ";M(H);" ";Y$:
LPRINT""
1410 LPRINTTAB(19)"## DAY  SOURCE OF INCOME
TAXABLE   NONTAX."
1420 LPRINTTAB(19) STRING$(54,45)
1430 FOR X=1 TO P1-1
1440 IF R(X)=0 LPRINTTAB(19) USING"## ##  %
        %  $###,##.##";X;DA(X);S(X);L(X)
1450 IF L(X)=0 LPRINTTAB(19) USING"## ##  %
         %              $###,##.##";X;DA(X);S(X);R(X)
1460 NEXT:LPRINTTAB(19) STRING$(54,45):LPRINT""
1480 LPRINTTAB(19)"   TOTALS FOR ";M(H);" ";Y$;"
 MONTH       YEAR TO DATE"
1482 LPRINT""
1485 LPRINTTAB(19)"TOTAL EARNED INCOME         ";:LPRINT
TAB(19)USING"$#####,###.##  $#####,###.##";ET,ES
1490 LPRINTTAB(19)"TOTAL DIVIDENDS             ";:LPRINT
TAB(19)USING"$#####,###.##  $#####,###.##";DT,DS
1500 LPRINTTAB(19)"TOTAL INTEREST              ";:LPRINT
TAB(19)USING"$#####,###.##  $#####,###.##";IT,IS
1510 LPRINTTAB(19)"TOTAL ANNUITY FUNDS         ";:LPRINT
TAB(19)USING"$#####,###.##  $#####,###.##";AT,AS
1520 LPRINTTAB(19)"TOTAL TAX FREE INTEREST     ";:LPRINT
TAB(19)USING"$#####,###.##  $#####,###.##";TT,TS
1530 LPRINTTAB(19)"SOCIAL SECURITY             ";:LPRINT
TAB(19)USING"$#####,###.##  $#####,###.##";GT,GS
1540 LPRINTTAB(19)"TOTAL LIFE INS. DIV.        ";:LPRINT
TAB(19)USING"$#####,###.##  $#####,###.##";FT,FS
1555 U$="$#####,###.##"
1557 LPRINT""
1560 LPRINTTAB(19)"TOTAL TAXABLE INCOME TO DATE
 ";:LPRINTTAB(19)USINGU$;LS
1570 LPRINTTAB(19)"TOTAL NONTAX. INCOME TO DATE
 ";:LPRINTTAB(19)USINGU$;RS
1580 LPRINTTAB(19)"TOTAL INCOME TO DATE
 ";:LPRINTTAB(19)USINGU$;Q
1590 PRINT:INPUT"PRESS ENTER TO CONTINUE";Z$:GOTO100
1600 CLS:PRINT:PRINT:PRINT:PRINT:PRINT:PRINT:PRINT:INPU
T"HAVE YOU SAVED FILES TO DISK (Y/N)";Z$
1610 IF Z$="N" GOTO 100
1620 PRINT:PRINT"OK. SO LONG!":END
1700 PRINT:INPUT"ENTRY ERROR.  PRESS ENTER AND TRY AGAI
N.";Z$:GOTO 665
1800 PRINT:INPUT"NO FILE ON DISK.  PRESS ENTER.";Z$:GOT
O100
4000 CLS:PRINT:PRINT
4010 PRINT"                    PERSONAL INCOME LEDGER":P
RINT
4020 PRINT"1) DISK FILES ARE NAMED AUTOMATICALLY.  THE
MONTHLY FILES ARE"
4030 PRINT"JAN, FEB, ETC.  THE YEARLY TOTAL FILES ARE J
AN1979, FEB1979,"
4040 PRINT"ETC., AND ACCUMULATE THE FIGURES FOR THE YEA
R TO DATE."
4060 PRINT
4090 PRINT"2) MAKE ENTRIES FOR THE MONTH.  SAVE TO DISK
 WITH DISK OPTION"
4100 PRINT"#1.  BEFORE ADDING ENTRIES FOR THE SAME MONT
H, USE DISK OPTION"
4110 PRINT"#2 TO LOAD THE PREVIOUS ENTRIES."
4130 PRINT:INPUT"PRESS ENTER";Z$:CLS:PRINT:PRINT:PRINT
4140 PRINT"3) UNDER TYPES OF INCOME, THE FIRST FOUR ARE
 TAXABLE, THE LAST"
4150 PRINT"FOUR NONTAXABLE AND ARE SO RECORDED BY THE P
ROGRAM."
4160 PRINT"MISCELLANEOUS TYPES SHOULD INCLUDE SUCH THIN
GS AS LOAN"
4170 PRINT"REPAYMENTS, GIFTS, REFUNDS, ETC.  MISCELLANE
OUS ITEMS ARE NOT"
4180 PRINT"TOTALED INTO NONTAXABLE INCOME SINCE THEY DO
N'T REPRESENT"
4190 PRINT"INCOME.":PRINT
4200 PRINT"4) MENU CHOICES 4, 5 AND 6 MAY BE USED DURIN
G PREPARATION OF"
4210 PRINT"MONTHLY FILES."
4220 PRINT:INPUT"PRESS ENTER";Z$:CLS
4230 PRINT:PRINT"5)  WHEN A MONTHLY FILE IS COMPLETE, U
SE DISK OPTION"
```

four (CORRECT AN ENTRY), five (DELETE AN ENTRY), or six (SEE A PRINTOUT ON VIDEO) at anytime while making these entries. Option four is also useful for checking the income type you have chosen, since it is not printed by either the video or hard copy printouts.

After the month's entries are on disk, disk option four loads the yearly totals from the previous month. Menu option two totals each type of income for the month and the year and displays the results. You can use disk option three to save the totals, menu option six for a complete video printout, and seven for hard copy (see Table 3 for sample).

Changes

Change to cassette data files will require alterations between lines 240 and 370. Sequential disk files, which are used, are quite similar to cassette files, and use of the Level II Basic Reference Manual should make this change simple.

Taxable items can be changed to nontaxable or vice versa in lines 7000-7070 and 6099-6820. Note that L( ) and R( ) collect the taxable and nontaxable items for printing in columns. Refer to the list of variables (Table 4) as necessary.

Miscellaneous entries include loan repayments, refunds, and money from the sale of stocks and bonds. The program doesn't handle capital gains because of their special requirements, but you can type the data on the back of the monthly pages.

```
4240 PRINT"#1 TO SAVE IT TO DISK.  NEXT, GET YOUR PRINT
ER READY.  IF THIS"
4245 PRINT"RESULTS IN LOSS OF THE PROGRAM AND/OR MONTHL
Y FILE, RELOAD"
4250 PRINT"THEM.  THEN USE DISK OPTION #4 TO LOAD THE Y
EARLY TOTAL FILE"
4255 PRINT"AND MENU OPTION #2 FOR COMPUTATIONS.  AT THI
S TIME, THE RESULTS"
4260 PRINT"MAY BE CHECKED ON VIDEO (MENU OPTION #6) OR
HARD COPY MADE"
4270 PRINT"(OPTION #7).  THE YEARLY TOTAL SHOULD BE SAV
ED TO DISK BEFORE"
4280 PRINT"LEAVING THE PROGRAM."
4290 PRINT
4310 PRINT"6) IT IS DESIRABLE TO KEEP A SET OF FILES ON
 A SECOND DISK"
4320 PRINT"FOR BACKUP."
4330 PRINT:PRINT"7) REENTRY TO THE PROGRAM, IF NEEDED,
IS AT LINE 100."
4340 PRINT:INPUT"PRESS ENTER";Z$:CLS:GOTO 40
6000 END
6099 REM * CLASSIFY INCOME AND SEPARATE INTO TAXABLE, L
(X),
AND NONTAXABLE, R(X) *
6100 IF E(X)<>0 PRINT E(X);
6110 PRINT:INPUT"AMOUNT OF EARNED INCOME";E(X)
6120 L(X) = E(X):GOTO 680
6200 IF D(X)<>0 PRINT D(X);
6210 PRINT:INPUT"AMOUNT OF DIVIDEND";D(X)
6220 L(X) = D(X):GOTO 680
6300 IF I(X)<>0 PRINT I(X);
6310 PRINT:INPUT"AMOUNT OF INTEREST";I(X)
6320 L(X) = I(X):GOTO 680
6400 IF A(X)<>0 PRINT A(X);
6410 PRINT:INPUT"AMOUNT OF ANNUITY";A(X)
6420 L(X)=A(X):GOTO 680
6500 IF T(X)<>0 PRINT T(X);
6510 PRINT:INPUT"AMOUNT OF TAX FREE INTEREST";T(X)
6520 R(X)=T(X):GOTO 680
6600 IF G(X)<>0 PRINT G(X);
6610 PRINT:INPUT"AMOUNT OF SOCIAL SECURITY";G(X)
6620 R(X)=G(X):GOTO 680
6700 IF F(X)<>0 PRINT F(X);
6710 PRINT:INPUT"AMOUNT OF LIFE INSURENCE DIVIDEND";F(X
)
6720 R(X)=F(X):GOTO 680
6800 IF C(X)<>0 PRINTC(X);
6810 PRINT:INPUT"AMOUNT OF MISCELLANEOUS ITEM";C(X)
6820 R(X)=C(X):GOTO 680
7000 REM * COMPUTATIONS *
7005 FOR X=1 TO P1-1
7010 ET=ET+E(X):DT=DT+D(X):IT=IT+I(X):AT=AT+A(X)
7020 TT=TT+T(X):GT=GT+G(X):FT=FT+F(X)
7025 NEXT
7030 ES=ES+ET:DS=DS+DT:IS=IS+IT:AS=AS+AT
7040 TS=TS+TT:GS=GS+GT:FS=FS+FT
7050 LS=ES+DS+IS+AS
7060 RS=TS+GS+FS
7070 Q=LS+RS
7075 RETURN
```

✓Reader Service—see page 226

NEW FROM

↑SUMWARE↓

AUDEX

SOUND EXPERIMENTER

For TRS-80 Level II, 16K

Gives you complete access to sounds, using multiple indexed routines. Jump to any of them instantly. Each sets up the parameters for a specific sound category. You input a few codes—hear the result immediately. Write the created sounds into your own programs, or add them to existing software.

Needs RS 277-1008 ($10.95 from Radio Shack) or equivalent mini speaker/amp. Easy instructions are part of AUDEX.

CASSETTE, $9.95

Florida residents add 4% sales tax

ORDER TODAY FROM

↑SUMWARE↓

Suite 406, 537 S. Sequoia Drive
West Palm Beach, FL 33409 ✓217

now you can

RAISE

and tilt your

more

"80-LNW"

VERSION NOW AVAILABLE TO SHROUD THE LNW INTERFACE BOARD

$25.50+$2.00 Shipping & Handling

silver-gray

erglass module

fits under the

"80" monitor.

$22.50 + $2.00 shipping and handling with check or money order. NYS residents add 7% tax.

SYRACUSE R&D CENTER

Box 125, DeWitt, N.Y. 13214

"our 10th year in R&D" ✓358

Disk Based Word Processor

750

Available Now!

Mailing List Option

750

PENSADYNE ✓207

4441 West First Ave.
Vancouver, B.C.
V6R 4H9

MasterCard VISA

604-224-3107

PRODUCTS FOR PEOPLE WHO CARE ABOUT THEIR TRS-80'S

NEWDOS-80 — By Apparat. Mix drives, chain commands, enhanced Disk BASIC, SUPERZAP in machine language, editor/assembler, printer spooling, etc. Excellent documentation.
.... #201007 (D) $149.95

NEWDOS — This is TRSDOS with over 200 fixes and enhancements.
35 track.... #201002 (D) $49.95
40 track.... #201001 (D) $59.95
77 track.... #201005 (D) $59.95

NEWDOS+ — Same as NEWDOS, but includes some indispensible utilities such as SUPERZAP, editor/assembler, etc.
35 track.... #201004 (D) $99.95
40 track..... #201003 (D) $110.
77 track..... #201006(D) $110.

BASIC COMPILER — By Microsoft. Develop your program in Disk BASIC, then compile it for up to 30 times the speed!
.... #201207 (D) $195.

mmsFORTH — By Miller Microcomputer Services. Unconventional yet powerful, this language is recursive, interactive, and fast..
.... #206000 (D) $79.95

AIDS-3 — By Meta Technologies. In our opinion the best DBMS for the money. In fact, we use it in our office. Suitable for personal as well as business uses. Performs operations on multiple conditions. Multi-key sort is machine-language assisted (it's **fast**). MAPS-3 (included) lets you generate reports in standard or user-defined formats.
.... #201300 (D) $69.95

CALCS — Adds arithmetic manipulation to your reports.
.... #201301 (D) $24.95

CORVUS TEN MEGABYTE DISK DRIVE — Imagine 95 40-track drives all on line at once! Max file seek time (searching all 95 directories) is six seconds. Includes TRSDOS-compatible DOS.
.... #105000 (call toll-free for price)

THE CONSTELLATION — By Corvus. Each Constellation allows up to 8 TRS-80's (up to 64 TRS-80's with 8 Constellations) to share up to four Corvus drives simultaneously. The standard Corvus DOS supports full communication between host processors.
.... #105001 (call toll-free for price)

MICROLINE-80 PRINTER — By Okidata. Designed especially for the TRS-80, this printer supports true TRS-80 graphics, upper/lower case, software-selectable 40/80/132 columns, pin-feed, roll paper, or typing paper. Relatively quiet and portable. .... #100002 $699.
TRACTOR (use for narrower forms)........ #100001 $110.

INVADERS+ with sound #200140 (C) $19.95 #200141 (D) $24.95

PINBALL with sound . #205111 (C) $14.95 #205122 (D) 20.95

CYLON RAIDER........ #200104 (C) $7.95

MEAN CHECKERS MACHINE —By Lance Micklus
.... #200123 (C) $19.95
.... #200122 (D) $24.95

DUNJONQUEST FANTASIES........ from $19.95

ANIMATED HANGMAN with sound...... #200105 (C) $9.95

LEVEL IV PRODUCTS INC., Dept 1D9
32238 SCHOOLCRAFT, SUITE F4 ✓14
LIVONIA, MICHIGAN 48150

Please add $2.50 for shipping and handling.
COD add another $1.25 plus 25% deposit, balance certified check or cash.

1(800)521-3305 • in Michigan (313) 525-6200 • Source:TCF541

This program considers life insurance dividends as nontaxable income, although they are actually premium refunds. You can change this by omitting " + FS" from line 7060.

Annuity payments can also be modified to record a second earned income by changing the entry name im lines 660, 694, 1310, 1510 and 6410.

The printout is for five and one-half by eight and one-half-inch paper at 12 characters per inch. The printout itself is 54 characters wide with a seven-character margin.

You can narrow the printout by as much as 14 characters by reducing the Source of Income heading from 24 to 10 and removing the same number of spaces from between the percentage signs in lines 1440 and 1450. Abbreviate other terms such as Total Earned Income (line 1485), and decrease the first number in STRING$(54,45) (lines 1420 and 1460).

To use a 40 column printer, get rid of the margin by removing TAB(7) after each LPRINT.

If you decrease the hard copy width and want your video printout to match, make the same changes in lines 1200 to 1390.

About the Program

The program asks for the year and month and automatically generates data files. In any month but January, the yearly total file will be from the preceding month.

The program will allow you to return to the menu if you choose a wrong number. If you press the break key and get into BASIC, type GOTO 100 to get back to the menu without loosing data files.

Line 240 saves the original file if you accidentally choose SAVE MONTH'S FILE TO DISK instead of LOAD MONTH'S FILE FROM DISK. Without some such safeguard the program could file a string of zeros and erase the original file. Line 299 gives a similar safeguard for SAVE YEARLY TOTALS TO DATE.

An ON ERROR GOTO prevents data loss if you choose LOAD MONTH'S FILE FROM DISK when there is no such file or pick LOAD YEARLY TOTALS.

Using TRSDOS 2.2 or 2.3 instead of NEWDOS and without the ON ERROR protection, the above causes the program to be booted out to BASIC rather than to DOS. If that should happen, the files can easily be saved by typing GOTO 100. The protection, however, works the same with TRSDOS as with NEWDOS.

In regard to disk space, none of my summary data has taken up more than one gran. Since the program requires nine grans, the total disk space, including the program, for a year comes to 32 grans and fits nicely onto one disk. Using hard copy, the same disk can be used year after year. ■

Y$	Year
V$	Yearly file name
M( )	String, name of month
H	Number of month
DA( )	Day of month
S( )	String, source of income
E( ),ET,ES	Earned income, monthly total, yearly total
D( ),DT,DS	Dividends, monthly total, yearly total
I( ),IT,IS	Interest
A( ),AT,AS	Annuity
T( ),TT,TS	Tax free interest
G( ),GT,GS	Social Security
F( ),FT,FS	Life insurance dividend
C( )	Miscellaneous items
L( )	All taxable items
LS	Total taxable income to date
R( )	All nontaxable items
RS	Total nontaxable income to date
Q	Total income to date
U$	Formatter
X	File counter
B,J,K,Z	Counters and null
P1	Number of items plus one (99 to end entries)

Table 4. Variables for Ledger Program

##	DAY	SOURCE OF INCOME	TAXABLE	NONTAX.
1	1	TIAA	$ 225.68	
2	1	IA PS	$ 153.00	
3	3	NW PS	$ 212.50	
4	3	SS (PD TO BANK)		$ 418.10
5	3	MASS MUT DIV		$ 33.15
6	5	GAS SERV	$ 160.00	
7	10	NE INV TRUST	$ 178.52	
8	10	TEXACO	$ 50.00	
9	15	OMAHO NAT'L CORP	$ 225.00	
10	15	MUNI INV TRUST		$ 58.10
11	15	TESORO	$ 108.00	
12	17	CORP INC FUND	$ 35.05	
13	21	KC P & L	$ 256.00	
14	29	JAPAN FUND	$ 15.00	

TOTALS FOR MAR 1979	MONTH	YEAR TO DATE
TOTAL EARNED INCOME	$ 0.00	$ 0.00
TOTAL DIVIDENDS	$ 1,393.07	$ 1,740.12
TOTAL INTEREST	$ 0.00	$ 0.00
TOTAL ANNUITY FUNDS	$ 225.68	$ 677.04
TOTAL TAX FREE INTEREST	$ 58.10	$ 495.10
SOCIAL SECURITY	$ 418.10	$ 1,254.30
TOTAL LIFE INS. DIV.	$ 33.15	$ 99.67
TOTAL TAXABLE INCOME TO DATE	$ 2,417.16	
TOTAL NONTAX. INCOME TO DATE	$ 1,849.07	
TOTAL INCOME TO DATE	$ 4,266.23	

Table 3. Income Ledger for Mar 1979

A real time waster.

Delay Loop

Allan S. Joffe
1005 Twining Road
Dresher, PA 19025

If you've ventured into the world of assembly programming you'll eventually need a delay or timing loop. Though there is no "best way" to delay a program, we can explore various methods.

Basically a time delay loads a value into a register (or values into a pair of registers) and then decrements this value to zero. This means we also need a mechanism that tells the computer when it has decremented the initial value to zero, so that it can continue the program.

The Zero Flag

Consider Listing 1, loaded using T-BUG.

The C2 instruction in memory location is a JPNZ,NN which signals the computer that if the comparison of the contents of the D and A registers is not zero that it should go back to the location specified by the NN which in our case is 4A02. This is the location of the decrement instruction.

One thing that this routine does not take into account is that when decrementing a single register, the zero flag is operative. (This is not true when we come to the case of decrementing a pair of registers.)

You notice that to continue the decrementing process, we had to tell the program to go back to a specific location. This means that if you did not load the program into the memory locations specified, you would have had to change the return locations in 4A07 and 4A08.

4A00	16	Load register D with N (value to decrement)
4A01	FF	Hex for 255
4A02	15	Decrement value in D register
4A03	3E	Load the A register with zero . . . . this is
4A04	00	the test value to show loop is done
4A05	BA	Compare value in D with value in A
4A06	C2	If this test shows A and D both zero the
4A07	02	loop is finished. If not pgm goes to 4A02
4A08	4A	and continues decrementing value in D
4A09	C3	When decrement is finished the program
4A0A	80	returns to T-BUG which is at 4380.
4A0B	43	

Listing 1.

FF	-1
FE	-2
FD	-3
FC	-4
FB	-5
FA	-6
F9	-7
F8	-8
F7	-9
F6	-10
F5	-11
F4	-12
F3	-13
F2	-14
F1	-15
F0	-16

Table 1.
Minus Relative Jump Values (twos complement form)

In Listing 2 we will use a relative jump instruction that obviates the need for such changes. This allows the program to relocate, for with the relative jump, the computer knows where it should go even if you change the origin of the routine.

In this listing I've introduced the mnemonic form of the instruction.

This listing is considerably shorter than the first one for several reasons. One is that the first example is used to illustrate the idea of comparing the contents of one register with another to determine when the zero condition is reached. The first listing tells the program where to go if the zero condition has not been reached. This takes three locations, while the new method uses but two locations in memory.

4A00	16	LD D,N
4A01	FF	
4A02	15	DEC D
4A03	20	JRNZ
4A04	FD	
4A05	C3	
4A06	80	
4A07	43	

Listing 2.

The jump relative instruction in location 4A03 works in conjunction with the FD information in location 4A04. We know that to make the loop decrement we have to keep returning to location 4A02. Location 4A02 is three steps back from location 4A04, counting location 4A04 as step one. You have signaled the computer that this is what you want by inserting FD.

What is the significance of FD? It is a minus three in twos complement form. If you had needed a relative jump of minus seven, then the FD would have become F9. Table 1 shows the relative jump values from minus one to minus 16.

The absolute delays produced by the simple timing loop using a single register is just about maximum when the register initial value is FF (255 decimal). I say "just about" because if the initial value in the register is 00, then you have 256 iterations of the loop, because the first decrement takes you from 00 to FF.

This is a bone for the nitpickers in the group.

Longer Delays

If we need longer delays, we can insert another identical loop after the first one, but there is another route to travel. This method decrements a pair of registers. If one register can be packed with FF, then two registers can be packed with FFFF.

Remember, if we use a pair of registers we will have to resort to some sort of a compare operation as with Listing 1, because when you decrement a pair of registers, the zero flag is not automatically working in your behalf. Thus we need some program steps to get the zero flag back from vacation.

Since the idea of the jump relative code seems to have merit we will go that route as well. When possible, I like to use the BC register pair for decrementing. BC seems to shout out "byte counter" and is, for me, a memory jogger.

Running Listing 3, you notice that it takes more time before the program returns to T-BUG.

```
4A00    01    LD BC,NN
4A01    FF
4A02    FF
4A03    OB    DEC BC
4A04    78    LD A,B
4A05    BO    OR B
4A06    B1    OR C
4A07    20    JRNZ
4A08    FA
4A09    C3
4A0A    80
4A0B    43
```

Listing 3. Decrementing the BC register pair.

This tells us that we have achieved a much longer delay than when decrementing the contents of a single register. The delay is in the neighborhood of 1.1 seconds.

The listing also uses the OR function, which is why we had to do what we did in memory location 4A04.

To operate any of the logic functions, you have to call on the services of the A register.

The jump relative figure in location 4A08 is equal to minus six (see Table 1), the proper value to get back to location 4A03, which contains the DEC BC instruction.

The above is but one viable routine, not the only one.

The Interrupt

Now let us get down to the business of combining a single register decrement and the register pair decrement to achieve significantly longer time delays. The game plan is to interrupt the decrement of the register pair while a single register decrements to zero. BC is the register pair in Listing 4 and the D register is used as the interrupt register.

When you load Listing 4 and RUN, be prepared to sit back for two minutes and 32 seconds before it returns you to T-BUG. The key element of the time delay is the value in location 4A05, the value being decremented in the D register.

If you change this value from FF to AA, then the delay time becomes one minute and 43 seconds. If it is changed to 64, then the time delay is one minute and one second. Changing this value to OA is going to give you a total time delay of about eight seconds. Thus this delay routine is quite flexible and should satisfy all but the most unusual needs.

You will get the most good out of this exercise if you use the Breakpoint of T-BUG to examine the program at various points. ■

```
4A00    01    LD BC,NN
4A01    FF
4A02    FF
4A03    OB    DEC BC
4A04    16    LD D,N
4A05    FF
4A06    15    DEC D
4A07    20    JRNZ
4A08    FD
4A09    78    LD A,B
4A0A    BO    OR B
4A0B    B1    OR C
4A0C    20    JRNZ
4A0D    F5
4A0E    C3
4A0F    80
4A10    43
```

Listing 4.

A graphics generator with emphasis on aesthetics.

Divine Proportions

David R. Cecil
Texas A&I University
Kingsville, TX 78363

Have you ever wondered about the size and shape of things, why paintings, books and cards have the dimensions they do?

The programs included with this article are designed to help readers explore aesthetics. The programs generate rectangles, ellipses and boxes. Decide which ratio of adjacent sides—minor to major axis—presents the most pleasing figure to you.

Diminishing Ratios

The programs generate figures of any size within bounds of the video display. The figures will then sequentially shorten one or more sides. Using the square as an example, it becomes a rectangle, then a smaller rectangle and so on with the ratio of width to length going from 1:1 to 0:1. As the video pauses between chânges, any key can be pressed to read the current width to length ratio from the screen.

If preferences for this ratio were randomly distributed, there wouldn't be much point to this experiment. The interesting and remarkable fact is that most people prefer the same width to length ratio. This preference falls into a fairly narrow range centered at 0.618.

Studies as far back as 1876 by G. T. Fechner in Germany confirm this attraction. He found a definite preference at 0.62 and most results in the range 0.57 to 0.67.

An aesthetic preference for 0.618 (actually $2/(1+\sqrt{5})$ was known even before the time of Pythagoras. Its reciprocal is of fundamental importance.

Called the golden ratio or divine proportion, it appears in many and diverse ways throughout mathematics, nature, art and architecture. This golden ratio is an irrational number 1.61803398 and is often expressed algebraically as $a:b = b:(a+b)$. The number is also known as Φ, taken from the first letter of the name of Phidios, the Greek sculptor and planner under Pericles, who utilized the golden ratio in his work.

```
0 REM PROGRAM TO DRAW SUCESSIVELY SMALLER RECTANGLES
1 REM BY DAVID R. CECIL
2 REM A SQUARE IS FORMED FIRST
3 REM IF THE SCALE FACTOR IS ONE THEN THE SQUARE
4 REM IS APPROX. 14.4 CM. PER SIDE
10 INPUT" SCALE FACTOR BETWEEN 0 AND 1";N
20 INPUT" T (FOR TOP) OR L (FOR LEFT)";V$:CLS
21 REM T DENOTES THAT THE TOP IS APPROACHING THE BOTTOM
22 REM (THE HORIZONTAL DIM. IS DECREASING), WHILE
23 REM L DENOTES THE LEFT APPROACHING THE RIGHT (THE
24 REM VERTICAL DIM. IS DECREASING)
30 X1=20+90*N:Y1=41-40*N
35 REM FORM THE TOP LINE FROM (20,Y1) TO (X1,Y1)
40 FOR X=20 TO X1
50 SET(X,Y1):NEXT X
55 REM FORM THE RIGHT SIDE FROM (X1,Y1) TO (X1,40)
60 FOR Y=Y1 TO 40
70 SET(X1,Y):NEXT Y
75 REM FORM THE BOTTOM FROM (X1,40) TO (20,40)
80 FOR X=X1 TO 20 STEP -1
90 SET(X,40):NEXT X
95 REM FORM THE LEFT SIDE FROM (20,40) TO (20,Y1)
100 FOR Y=40 TO Y1 STEP -1
110 SET(20,Y):NEXT Y
120 GOSUB 300
125 REM LINES 140 TO 210 RE USED FOR CHANGING THE TOP,
126 REM WHILE LINES 220 TO 290 ARE FOR CHANGING THE LEF
    T SIDE
130 IF V$="L" THEN 220
135 REM ERASE TOP LINE
140 FOR Y=Y1 TO 40
150 FOR X=20 TO X1
160 RESET(X,Y):NEXT X
165 REM FORM NEW TOP LINE 1 LINE BELOW OLD TOP LINE
170 FOR X=20 TO X1
180 SET(X,Y+1):NEXT X
190 GOSUB 300
200 NEXT Y
210 END
215 REM ERASE LEFT LINE
220 FOR X=20 TO X1
230 FOR Y=Y1 TO 41
240 RESET(X,Y):NEXT Y
245 REM FORM NEW LEFT LINE 1 LINE TO RIGHT OF OLD LEFT
    LINE
250 FOR Y=Y1 TO 40
260 SET(X+1,Y):NEXT Y
270 GOSUB 300
280 NEXT X
290 END
291 REM SUBROUTINE 300 PRODUCES A PAUSE AND ALLOWS YOU
292 REM TO READ, BY PRESSING ANY KEY, THE RATIO OF
293 REM SHORTER SIDE TO LONGER SIDE
300 FOR Z=1 TO 300
310 IF INKEY$="" THEN 340
320 IF V$="L" THEN PRINT@950,(X1-X)/(90*N);:GOTO 340
330 PRINT@950,(40-Y)/(40*N);
340 NEXT Z
350 RETURN
```

Program Listing 1. The Square

The Divine Proportion by H. E. Huntley and *The Geometry of Art and Life* by M. Ghyka have numerous examples of the occurrence of the golden ratio; such as multiple reflections of light, arrangement of leaves on plants, the musical chromatic scale, the great pyramid, the human figure, dynamic symmetry, and on and on. (Both books are published by Dover Books.)

Take a look at the computer programs. The first one generates a square of any desired size. Then, having asked whether to shorten the vertical or the horizontal dimension, it proceeds to do so.

Note that the ratios given on the screen are approximations. Pixels are not square and scale factors are used in evaluating each ratio. If the lengths of lines are measured directly on the video screen, the ratios obtained may differ slightly from those of the program.

The second program generates nine ellipses, pausing after each one. The major axis stays constant and may be designated horizontal or vertical. Use the break key to terminate this program.

Most people prefer an ellipse with a minor to major ratio in the range .57 to .67. The 5.4 value used in line 90 can be changed to something a bit smaller (such as 5.1) to shorten the horizontal width. If it is made higher (such as 5.6), the horizontal dimension is lengthened. Keep this in mind if you find that the ratios are not quite true in the video representation.

Changing the 90 in step 60 to a smaller value will display fewer points (try 45), while more points will be displayed for values larger than 90 (such as 180). The program will only display one ellipse at a time, if 60 is changed to 50 in line 170.

To generate other ellipses, change line 190. Be sure all data values are less than 10 so that the ellipse does not exceed the size of the video screen.

The Parallelepiped

The last program generates a box (rectangular parallelepiped) with square front and back. The top is gradually lowered. Ratios of vertical to horizontal and slant (depth) to the vertical dimension are given.

Fig. 1 should be used in conjunction with Program Listing 3. The numbers in the program's REM statements refer to lines of the box represented in the figure.

Preferences for box sizes are not nearly as pronounced as for the rectangle, but four of the more interesting box dimensions are:

- 1,Φ,Φ—used for the Golden Chamber containing the tomb of Rameses IV.
- 1,1,Φ—found for the volume of a stool of Tutankhamen's tomb.
- 1,Φ,$Φ^2$—called the "Golden Solid" and often used for Egyptian tombs.
- 1,$Φ^2$,$Φ^3$—found to be the volume in many pieces of furniture of the Queen Anne and Chippendale styles. ■

```
10 INPUT"ENTER SCALE FACTOR BETWEEN 0 AND .865";N
20 X1=20+90*N:X2=(X1-20)/3
30 Y1=41-40*N:Y2=(40-Y1)*(4/21):CLS
35 REM DRAW FRONT TOP LINE 1
40 FOR X=20 TO X1
50 SET(X,Y1):NEXT X
55 REM DRAW FRONT RIGHT SIDE 2
60 FOR Y=Y1 TO 40
70 SET(X1,Y):NEXT Y
75 REM DRAW FRONT BOTTOM LINE 3
80 FOR X=X1 TO 20 STEP -1
90 SET(X,40):NEXT X
95 REM DRAW FRONT LEFT SIDE 4
100 FOR Y=40 TO Y1 STEP -1
110 SET(20,Y):NEXT Y
115 REM DRAW RIGHT TOP LINE 5
120 FOR X=X1 TO X1+X2
130 SET(X,Y1-(Y2/X2)*(X-X1)):NEXT X
135 REM DRAW RIGHT BACK VERTICAL LINE 6
140 FOR Y=Y1-Y2 TO 40-Y2
150 SET(X1+X2,Y):NEXT Y
155 REM DRAW RIGHT BOTTOM LINE 7
160 FOR X=X1+X2 TO X1 STEP -1
170 SET(X,40-(Y2/X2)*(X-X1)):NEXT X
175 REM DRAW LEFT TOP LINE 8
180 FOR X=20 TO 20+X2
190 SET(X,Y1-(Y2/X2)*(X-20)):NEXT X
195 REM DRAW BACK TOP LINE 9
200 FOR X=20+X2 TO X1+X2
210 SET(X,Y1-Y2):NEXT X
215 REM DRAW LEFT BACK VERTICAL LINE 10
220 FOR Y=Y1-Y2 TO 40-Y2 STEP 2
230 SET(20+X2,Y):NEXT Y
235 REM DRAW BACK BOTTOM LINE 11
240 FOR X=20+X2 TO X1+X2 STEP 2
250 SET(X,40-Y2):NEXT X
255 REM DRAW LEFT BOTTOM LINE 12
260 FOR X=20 TO 20+X2 STEP 2
270 SET(X,40-(Y2/X2)*(X-20)):NEXT X
272 FOR Z=1 TO 300:NEXT Z
275 REM LOWER THE FRONT TOP LINE 1
280 FOR X=20 TO X1
290 RESET(X,Y1):SET(X,Y1+1):NEXT X
295 REM LOWER RIGHT TOP LINE 5
300 FOR X=X1 TO X1+X2
310 RESET(X,Y1-(Y2/X2)*(X-X1))
320 SET(X,1+Y1-(Y2/X2)*(X-X1)):NEXT X
325 REM LOWER BACK TOP LINE 9
330 FOR X=X1+X2 TO 20+X2 STEP -1
340 RESET(X,Y1-Y2):SET(X,Y1-Y2+1):NEXT X
345 REM LOWER LEFT TOP LINE 8
350 FOR X=20+X2 TO 20 STEP -1
360 RESET(X,Y1-(Y2/X2)*(X-20))
370 SET(X,1+Y1-(Y2/X2)*(X-20)):NEXT X
375 REM REDRAW TOP PART OF LEFT BACK VERTICAL LINE 10
380 FOR Y=Y1 TO Y1-Y2+2 STEP -1
390 RESET(20+X2,Y):NEXT Y
400 FOR Y=Y1 TO Y1-Y2+1 STEP -2
410 SET(20+X2,Y):NEXT Y
415 REM ROUTINE TO PRINT RATIOS IF DESIRED
420 FOR Z=1 TO 300
430 IF INKEY$="" THEN 450
440 PRINT@950,(40-Y1)/(40*N);SQR(X2[2+Y2[2)/(40-Y1)/1.9
    5;
450 NEXT Z
460 Y1=Y1+1
470 IF Y1<40-Y2 THEN 280 ELSE STOP
```

Program Listing 2. The Ellipse

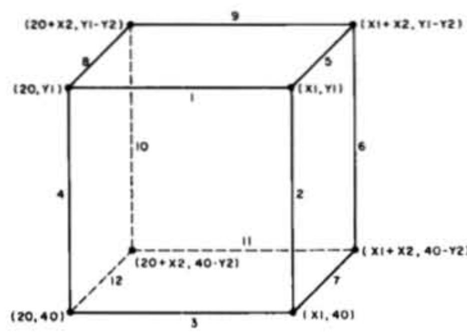


Fig. 1

```
0 ON ERROR GOTO 200
10 PRINT"CHOOSE SCALE FACTOR BETWEEN 0 AND 1":INPUT N
20 W=10*N:A=W:B=W
30 PI=3.14159:WW=10
40 PRINT"INPUT H FOR CONSTANT LENGTH MAJOR AXIS TO BE H
    ORIZONTAL,V FOR CONSTANT LENGTH VERTICAL MAJOR AXI
    S":INPUT D$
50 CLS
60 FOR I=0 TO 2*PI STEP 2*PI/90
70 IF D$="V" THEN A=W ELSE B=W
80 X1=A*COS(I):Y1=B*SIN(I)
90 X=65+5.4*X1
100 Y=23-2.3*Y1
110 SET(X,Y)
120 NEXT I
130 FOR P=1 TO 300
140 IF INKEY$="" THEN 160
150 PRINT@59,WW/10;
160 NEXT P
170 READ WW:W=WW*N:GOTO 60
180 GOTO 180
190 DATA 8.3,8,7.5,6.7,6.2,5.7,5,4
200 RESUME 180
```

Program Listing 3. The Box

The Micromatic 80, a step ahead for your TRS-80™ and other microprocessors.

The Micromatic 80 is a TTL based interface unit designed to integrate the popular Radio Shack TRS-80®, and many other processors with parallel Centronics™ compatible output, to the IBM manufactured Selectric™ terminal.

- Used IBM Selectric terminals are cleaned and functionally checked before shipping.
- 90 day warranty on interface only.
- No software required.
- $795. plus $25. for shipping and handling.
- TRS-80 to Centronics cable $29.00.
- Mail or phone orders accepted.
- Certified Check, Money Order.

Write or call for Free Brochure (317) 299-8614

The MICROMATIC Corporation

5747 W. 85th Street • Indianapolis, Indiana 46278

425

--frq:-- LEVEL II

SOUND STX

REAL TIME DISPLAY OF SOUND INPUT WITH RECORDER MIKE MACHINE LANGUAGE PROGRAM LISTING OF LESS THAN 100 BYTES FULL LOUDNESS DEPTH FROM UNMODIFIED '80 WITH CTR-41 RECORDER DISPLAY MOVES TO MUSIC OR VOICE FOR EXPERIMENTS OR LAZY VIEWING SEND 6.00 CK OR MO WITH STAMPED SELF ADDRESSED ENVELOPE TO

EDF 359
BOX 3054
BRANFORD CT.
06405

TRS 80

TRS-80 Model I

MODEL 3
call today!

699

Level II 16K. 26-1056

Model II

$3499

64K, 1-Disk TRS-80 Model II System

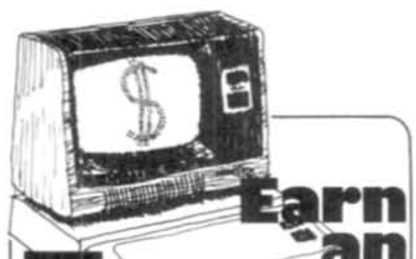
We accept check, money order or phone orders with Visa or Master Charge. (Shipping costs added to charge orders).

VISA CHARGE IT

Computers Unlimited 419-332-4881 Collect

1524 OAK HARBOR ROAD, FREMONT, OHIO 43420 321

TRS-80 is a Trademark of Tandy Corp.



Earn an EXTRA Income with your TRS-80*

Everyone is feeling the bite of inflation but some TRS-80 owners have found a way to ease the pain. Some are making a few hundred extra dollars and a few are making thousands of dollars in their spare time. Our booklet "Money Making Ideas for the TRS-80" is a collection of these money making methods, along with program sources, program hints, getting started suggestions and pit-falls to avoid.

Send $9.95 to:

J&R Electronics
P.O. Box 492 398
Crystal City, MO 63019

* TRS-80 is a trademark of the Radio Shack Division of Tandy Corporation

Write a moving story.

Walking Words

Hubert C. Borrmann
2840 South Circle Dr. #209
Colorado Springs, CO 80906

The subroutine described here moves several instructions, or a message, across the TRS-80 screen from right to left, and repeats until any key is pressed. It shortens messages to 58 positions and inserts one period in front of the message and two periods at the end to show messages as unbroken bands.

To use the subroutine, a message is placed in variable S1$. The variable is linked to the subroutine as follows:

```
100 S1$="TYPE 'SPACE' WHEN READY"
110 GOSUB 10000.
```

This will move your message on the bottom line from right to left and repeat it. Should you desire to have the message move on another line, place the beginning address of this line into variable SA and link to entry point 10100, as follows:

```
100 S1$="TYPE 'SPACE' WHEN READY"
110 SA = 512
120 GOSUB 10100.
```

The value of whatever key has been pressed will be returned to the main program in S6$; and S1$ will not be destroyed at exit.

Pressed for Space

There are several comments in the subroutine which you may leave out if pressed for space, but the comments are in fact very handy in any subroutine. The subroutine will shorten messages to 58 positions and will insert one period in front of the message, and two periods at the end. This will make it appear as an unbroken band. The time delay after each execution of the print instruction may be changed in line 10180.

Entry point 10000 will print the message on the bottom line, while entry point 10100 or 10110 bypasses loading the SA. These lines assume that SA was loaded in the main program with the starting address.

Line 10110 fills the "background" with periods. The next three lines adjust the length, determine the highest print position on the line for centering and insert the periods in the message variable.

Line 10150 resets several variables so that S4$ extracts the characters to be printed out of S1$ until S4$ is equal to S1$. On every pass the effective print position is advanced by one.

S3, which points to position 1 in S1$, remains one. S2 indicates the number of positions to be extracted and is increased by one on each pass.

After the complete message appears on the screen (S7 = S9), the length of S4$ is reduced by one on each pass, controlled by S2.

S3 is increased by one each pass, so that S4$ extracts the second to the last character out of S1$. Then the third to the last is extracted etc., until its length is zero. The effective print position remains unchanged.

The program then branches to 10150 and the process is repeated until the test of S6$ indicates a key or space is depressed. The print line is reset to spaces and the periods are removed from S1$. This allows a subsequent GOSUB without reloading S1$.

At this point the subroutine then returns to the main program. I find this routine quite handy, and you might, too. ■

```
10 CLEAR 200 : CLS
20 PRINT@ 128,"INSTRUCTIONS # 1"
30 S1$="PRESS 'SPACE' WHEN READY"
40 GOSUB 10000
50 CLS : PRINT@ 512,"INSTRUCTIONS # 2"
60  SA=640
70 GOSUB 10100
80 CLS : PRINT@ 0,"AND MORE INSTRUCTIONS"
90 S1$="TYPE 'SPACE'-BAR TO CONTINUE, 'E' TO END"
100 GOSUB 10000
110 IF S6$="E" THEN END ELSE 20
10000 SA=960:GOTO 10110 ' MOVING-BAND-SUBROUTINE
10010 ' AT ENTRY S1$ CONTAINS THE MESSAGE, UP TO 58 CHA
    RACTERS
10020 ' AT EXIT  S6$ CONTAINS VALUE OF DEPRESSED KEY
10030 ' OTHER     : S2  = LENGTH OF MESSAGE TO BE DISPL
    AYED
10040 ' VARIABLES : S3  = STARTING POSITION IN STRING
10050 '             S4$ = WORKING STRING
10060 '             S5  = HIGHEST PRINT ADRESS ON SCREE
    N
10070 '             S7  = WORK VARIABLE
10080 '             S8  =   "     "
10090 '             S9  = LENGTH OF S1$
10100 '             SA  = START ADRESS OF LINE ON SCREE
    N
10110 PRINT@ SA,STRING$(63,".");
10120 S2=LEN(S1$) : IF S2 > 58 THEN S2=58
10130 S5=INT(SA+(64-S2)/2)+S2
10140 S4$=MID$(S1$,1,S2) : S1$="."+S4$+".."
10150 S7=0 : S9=LEN(S1$) : S2=S9 : S3=1
10160 IF S2=> S7 THEN S2 = S7+1
10170 S4$=MID$(S1$,S3,S2)
10180 PRINT@S5-S7,S4$; : FOR S8=1 TO 15 : NEXT S8
10190 S6$=INKEY$ : IF S6$ <> "" THEN 10230
10200 IF S7=S9 THEN 10220
10210 S7=S7+1 : IF S7 < S9 THEN 10160
10220 S2=S2-1 : S3 = S3+1 : IF S2 <0 THEN 10150 ELSE 10
    160
10230 PRINT@ SA,STRING$(63," "); : S1$=MID$(S1$,2,LEN(S
    1$)-3)
10240 RETURN                ' END OF THIS SUBROUTINE
```

Program Listing

A supermarketing program for creatures of impulse.

Mind Your A's & P's

Lois L. Leonard
12733 Stone Canyon Rd.
Poway, CA 92064

Ah, the joys of grocery shopping! Your delight at finding the local social club is meeting in the aisles of your store. The rapture when you realize the next item on your list is on the other side of the market. The treasure hunt to locate where they hide capers.

If that's how you feel about grocery shopping, maybe this article is not for you. This program produces a grocery list that leads you past aisles that contain nothing you need—preventing your buying all those nifty impulse items—and gets you to the check-out counter quickly, saving both time, steps and money.

Groceries Are Data

The program, written on a 16K TRS-80, loads in 7K and runs in over 10K. It requires a printer and a light pen, but can be easily re-written for keyboard input. The light pen is a simple device that plugs into the tape recorder input port in the back of the keyboard. My pen is from Oasis System and the routine in this program is taken from their instruction manual.

The DATA statements are groceries listed by aisle number. Unless you have the same store layout, a trip to the store with pen and paper is in order. On the same trip study how you shop. Do you like to start at one end or the other?

List each aisle and beneath these headings list the items you normally use. Leave a blank for novelty foods and rarely purchased items.

If there are a large assortment of items normally purchased on one aisle, make each side a subcategory. For example, see Aisle 11 in the program.

The data for each heading is laid out beginning with an asterisk. The second and third characters are the number of items in the aisle. The remaining characters name the aisle. If an item is dropped or added, the number of items in the list must be changed. Also, an "end" must be used, and it must be counted

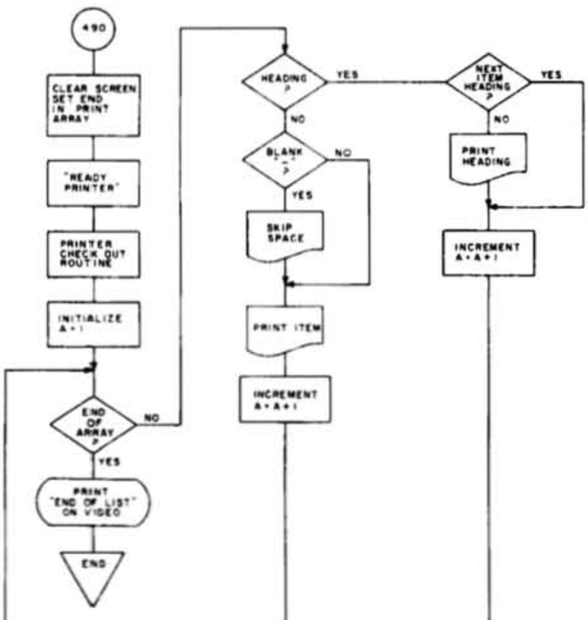


Fig. 1. Printer Routine

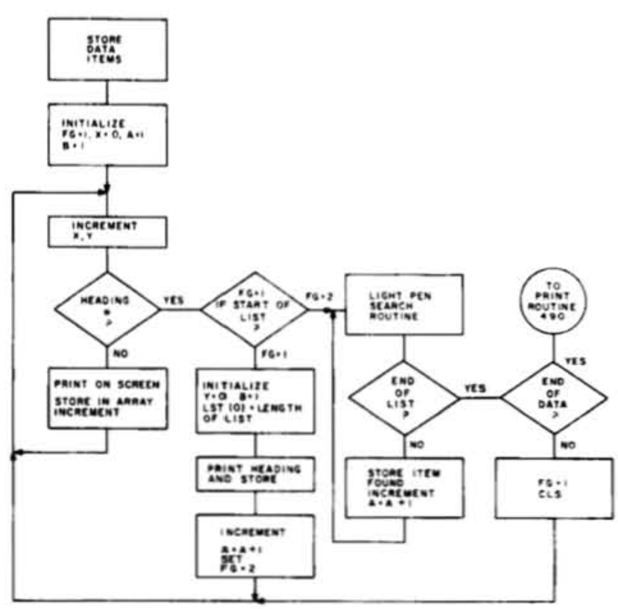


Fig. 2. Grocery List by Light Pen

as an item.

The program is set up with a maximum of 20 items per aisle. More than 20 slows down the light pen search. If the program is to be rewritten for keyboard input, items would be limited only by your screen space.

On items that need further explanation put "——" in front of them. Examples of this are "-------Jello", "-------spices", "——jelly". There are too many varieties of spices to list separately. When "------spice" is chosen, the printer skips a line before the item and leaves you room to write the name of the spice.

Some of the benefits of this program surprised me. On my first test trip, I was able to do my ordinary shopping minus one-half hour of time. After using it for a month, I found I was buying fewer impulse items. ■

Program Listing

```
10 REM                    GROCERY LIST  USING THE  LIGHT
     PEN
20 REM                    WRITTEN BY LOIS L. LEONARD
30 REM                    SEPTEMBER 1979
40 REM
50 DIM LT(25),LT$(350),ST$(350),LST(40),LST$(40)
55 REM
60 REM                              READ DATA STATEMENTS INTO
     ARRAYS
70 REM
75 CLS:PRINT@ 468,"LOADING DATA"
76 FOR Y = 1 TO 20:READ LT(Y):NEXT Y
80 FOR X = 1 TO 350
100 READ LT$(X)
105 IF LT$(X)="*END" THEN GOTO 130
110 NEXT X
130 X=0:A=1:FG=1:B=1
140 REM
150 REM                              PRINT ARRAYS AND TEST FOR
     HEADING
160 REM
165 CLS
170 X=X+1:Y=Y+1
180 IF LEFT$(LT$(X),1)="*" THEN GOTO 260
190 PRINT@ LT(Y)+4,LT$(X);
195 LST(B)=LT(Y):LST$(B)=LT$(X):B=B+1
200 GOTO 170
220 REM
230 REM                               HEADING ROUTINE
240 REM
260 IF FG=2 THEN GOTO 360
265 C=LEN(LT$(X)):Y=0
270 PRINT@ 20, RIGHT$(LT$(X),C-3)
280 J1$=MID$(LT$(X),2,2)
282 LST(0)=VAL(J1$)
285 B=1
290 ST$(A)=LT$(X)
300 A=A+1
310 FG=2
320 GOTO 170
330 REM
340 REM                                LIGHT PEN SEARCH
350 REM
360 GOSUB 9100
370 IF LST$(SCAN)<>"END"GOTO 430
375 IF LT$(X)="*END" THEN GOTO 490
380 FG=1
385 CLS
390 GOTO 180
400 REM
410 REM                                STORE ITEM FOUND
420 REM
430 ST$(A)=LST$(SCAN)
440 A=A+1
450 GOTO 360
460 REM
470 REM                                PRINT OUT GROCERY LIST
480 REM
490 CLS
495 ST$(A)="*END"
500 PRINT"READY PRINTER"
510 PRINT"HIT ENTER WHEN READY"
520 A$=INKEY$: IF A$="" THEN 520
530 A=PEEK(14312)                      :'PRINTER CHECK OUT ROU
     TINE
540 IF A=63 GOTO 600
550 IF A=255 PRINT"PRINTER IS TURNED OFF":GOTO500
560 IF A=223 PRINT"YOU HAVE A PAPER PROBLEM":GOTO 500
600 A=1
610 IF ST$(A)="*END" THEN PRINT "END OF LIST":END
620 IF LEFT$(ST$(A),1)="*" GOTO 660
625 IF LEFT$(ST$(A),1)="-" THEN LPRINT" ":'ALLOWS SPACE
     TO WRITE
630 LPRINT ST$(A)
640 A=A+1
650 GOTO 610
660 IF LEFT$(ST$(A+1),1)="*" GOTO 700
670 C=LEN(ST$(A))
680 LPRINT TAB(5) RIGHT$(ST$(A),C-3)
700 A=A+1
710 GOTO 610
9000 OUT 255,0
9010 FOR Z=0 TO 6:NEXT Z
9020 LP=(INP(255) AND 128)
9030 RETURN
9100 REM                                      MAGIC LIGHT PEN S
     UBROUTINE
9110 L=LST(0)
9120 C$=CHR$(140)
9130 FOR I=1 TO L
9140 PRINT@ LST(I),C$;
9150 NEXT I
9160 GOSUB 9000
9170 IF LP=0 GOTO 9160
9180 SCAN=1
9190 PRINT@ LST(SCAN)," ";
9200 GOSUB 9000
9210 IF LP=0 GOTO 9260
9220 PRINT@ LST(SCAN),C$;
9230 SCAN=SCAN+1
9240 IF SCAN<= L GOTO 9190
9250 GOTO 9160
9260 PRINT@ LST(SCAN),C$;
9270 GOSUB 9000
9280 IF LP=0 GOTO 9160
9290 CNT=2
9300 PRINT@ LST(SCAN), " ";
9310 GOSUB 9000
9320 PRINT@ LST(SCAN),C$;
9330 IF LP<> 0 GOTO 9180
9340 GOSUB 9000
9350 IF LP =0 GOTO9160
9360 CNT=CNT-1
9370 IF CNT<> 0 GOTO 9300
9380 PRINT@ LST(SCAN)-2,"=>";
9390 PRINT@ LST(SCAN)+1,"<=";
9400 GOSUB 9000
9410 IF LP<> 0GOTO9400
9420 RETURN
10000 REM              GROCERY STORE ITEMS
10010 REM       ALL ITEMS ARE IN GROUPS AS THEY APPEAR
     IN THE
10020 REM       STORE.  ALL HEADINGS HAVE A "*" BEFORE
     THE
10030 REM       HEADINGS.  THE NUMBER AFTER THE "*" WIL
     L
10040 REM       BE THE NUMBER OF ITEMS IN THE LIST.  IF
     YOU
10050 REM       ADD OR DELETE AN ITEM, REMEMBER TO CHAN
     GE THE
10060 REM       NUMBER.
10070 REM
10080 DATA 138,202,266,330,394,458,522,586,650,714
10090 DATA 168,232,296,360,424,488,552,616,680,744
10100 DATA *20AISLE 13 - PRODUCE
10110 DATA CANTALOUPES,MUSHROOMS,APPLES,CELERY
10120 DATA CARROTS,PEARS,LETTUCE,CUCUMBERS
10130 DATA TOMATOES,AVOCADOES,SQUASH
10140 DATA BANANAS,POTATOES,CORN,CABBAGE
10150 DATA ONIONS,GRAPEFRUIT,GRAPES,----------
10160 DATA END
10170 DATA *07  BAKERY - REAR OF STORE
10180 DATA BREAD,COFFEE CAKE,WIENER BUNS
10190 DATA HAMBURGER BUNS,BAGELS,----------
10195 DATA END
10200 DATA *11AISLE 13 - DRY FRUIT AND NUTS AND WINES
10210 DATA RAISINS,DATES,PEANUTS,PRUNES
10230 DATA BURGUNDY,CABERNET SAVIGNON,ROSE
10240 DATA LAMBRUSCO,CHABLIS,--------------
10250 DATA END
10260 DATA *18AISLE 12 -SOFT DRINKS - DRESSINGS AND SAU
     CES
10270 DATA DIET ROOT BEER,DIET COLA,GINGER ALE
10280 DATA 7-UP,DR. PEPPER
10300 DATA ITALIAN DRESSING,FRENCH DRESSING
10310 DATA  GOOD SEASON MIX,MUSTARD,MAYONNAISE
10315 DATA SPAGHETTI SAUCE,TOMATO SAUCE
10320 DATA TOMATO PASTE,CATSUP
10330 DATA BAR-B-Q SAUCE,A-1 STEAK SAUCE
10340 DATA ------------,END
10350 DATA *18AISLE 11 - GELATIN AND BAKING SUPPLIES
10360 DATA ----------- JELLO,----------PUDDING MIX
10370 DATA FLOUR,WHOLE WHEAT FLOUR,PANCAKE MIX
10380 DATA YEAST,CAKE MIX,FROSTING
10390 DATA DATE NUT BREAD MIX,BANANA BREAD MIX,MAPLE SY
     RUP
10400 DATA CORN STARCH,BAKING POWDER,BAKING SODA
10410 DATA CHOC. CHIPS,CORN SYRUP,------------
10420 DATA END
10430 DATA *13AISLE 11 - SALAD OIL  PASTAS  SPICES
10440 DATA STOVE TOP STUFFING MIX,RICE
10450 DATA MACARONI AND CHEESE,NOODLES,SPAGHETTI
10460 DATA  PARMESAN CHEESE,---------SPICE
10470 DATA SALAD OIL,SHAKE N BAKE,VINEGAR
```

Program continues

```
10480 DATA SALT,------------,END
10490 DATA *13AISLE 10 - TEA - COFFEE - SOUP - CEREALS
10500 DATA GROUND COFFEE,INSTANT COFFEE,TEA BAGS
10510 DATA INSTANT TEA,TOMATO SOUP,VEGETABLE SOUP
10520 DATA MUSHROOM SOUP,----------SOUP
10550 DATA RAISIN BRAN,SUGAR,BROWN SUGAR
10560 DATA ------------,END
10570 DATA *15AISLE 9 PAPER PRODUCTS
10580 DATA KOTEX,TOILET PAPER,TISSUES
10605 DATA PAPER TOWELS
10610 DATA PAPER NAPKINS,PAPER CUPS,PAPER PLATES
10620 DATA ALUMINUM FOIL,LARGE TRASH BAGS
10630 DATA SMALL TRASH BAGS,PLASTIC WRAP
10640 DATA SANDWICH BAGS,CIGARETTES,------------
10650 DATA END
10660 DATA *12AISLE 8 - TOILETRIES
10670 DATA TOOTH PASTE,BUFFERIN,RUBBING ALCOHOL
10680 DATA TOOTH BRUSH,HAND LOTION,SHAMPOO
10690 DATA CREAM RINSE,DEODORANT,SHAVING CREAM
10700 DATA AFTER SHAVE,------------,END
10710 DATA *13AISLE 8 - FROZEN VEGETABLES
10720 DATA SOY SAUCE,SQUASH,BROCCOLI
10730 DATA PEAS,CARROTS,CORN,SPINACH
10740 DATA BEANS,MIXED VEGETABLES,CUP OF NOODLES
10750 DATA MATZO BALL SOUP MIX,------------
10760 DATA END
10770 DATA *09BACK OF STORE - DAIRY
10780 DATA BEER,MILK,LOWFAT MILK,BUTTER MILK
10790 DATA YOGURT,SOUR CREAM,COTTAGE CHEESE
10800 DATA ------------,END
10810 DATA *15AISLE 7 - FROZEN FOOD & JUICES - PICKLES
10820 DATA TV DINNER,CHICKEN PIE,PIZZA
10830 DATA BEEF PIE,LASAGNA
10860 DATA ORANGE JUICE,LEMONADE,COOL WHIP
10870 DATA FROZEN WAFFLES,GREEN OLIVES,BLACK OLIVES
10880 DATA DILLS,GERKINS,------------,END
10890 DATA *16AISLE 6   FROZEN PIES  CANDY  PRESERVES
10900 DATA ICE CREAM,----------TOPPING,FROZEN PIE
10910 DATA SACCHARIN
10930 DATA POP CORN,CANNED MIXED NUTS,WALNUTS
10940 DATA ALMONDS,MARSHMALLOWS,PEANUT BUTTER
10950 DATA ----------JELLY,----------PRESERVE
10960 DATA BATTERIES,LIGHT BULBS,------------
10970 DATA END
10980 DATA *10BACK OF STORE - DELECATESSEN
10990 DATA EGGS,MARGARINE,BUTTER
11000 DATA ----------CHEESE,DINNER ROLLS,BISCUITS
11010 DATA CREAM CHEESE,SLICED AMERICAN CHEESE
11020 DATA ------------,END
11030 DATA *08AISLE 5 - PET FOOD AND HARDWARE
11040 DATA CANNED CAT FOOD,DRY CAT FOOD,KITTY LITTER
11050 DATA DOG BISCUITS,SULFODENE,COFFEE FILTERS
11060 DATA ------------,END
11070 DATA *13AISLE 4 - WAX - BLEACH
11080 DATA BLEACH,SOFTENER - SHEETS
11090 DATA SOFTENER - LIQUID,RAID,S.0.S PADS
11100 DATA LYSOL,PLEDGE,SILVER POLISH
11110 DATA LIQUID GOLD,-------SHOE POLISH
11120 DATA RUBBER GLOVES,------------,END
11130 DATA *11AISLE 4 - DETERGENT
11140 DATA HAND SOAP, DISH WASHING LIQUID
11150 DATA DISHWASHER DETERGENT,DETERGENT
11160 DATA FANTASTIC,TOILET BOWL CLEANER
11170 DATA COMET CLEANER,PINE SOL
11180 DATA POTATO CHIPS,------------,END
11190 DATA *11AISLE 3 - JUICE - CANNED MILK
11200 DATA NYLONS,TOMATO JUICE,TANG
11210 DATA APPLE JUICE,LEMON JUICE,LIME JUICE
11220 DATA CIDER
11240 DATA EVAPORATED MILK,DRY MILK
11250 DATA ------------,END
11260 DATA *13AISLE 2 - CANNED FRUIT - CANNED FISH
11270 DATA PINEAPPLE,PEACHES,PEARS
11280 DATA FRUIT COCKTAIL,GRAPES,APPLE SAUCE
11290 DATA PUMPKIN,TUNA,SARDINES,SALMON
11300 DATA CHILI CON CARNE,------------,END
11310 DATA *12AISLE 2 - CANNED VEGETABLES
11320 DATA WHOLE MUSHROOMS,SLICED MUSHROOMS
11330 DATA ----------TOMATOES,GREEN BEANS
11340 DATA PEAS,BEETS,CORN,KIDNEY BEANS
11350 DATA PORK & BEANS,PIMENTOES,------------
11360 DATA END
11370 DATA *17AISLE 1 - MEATS
11380 DATA HAM,TONGUE,CORNED BEEF,PORK CHOPS
11390 DATA LAMB CHOPS,PORK LINKS,ROAST,NY STEAK
11400 DATA RIB EYE STEAK,ROUND STEAK,BEEF CHUCK
11410 DATA ----------FISH,CHICKEN - WHOLE
11420 DATA CHICKEN - PARTS,GROUND BEEF,------------
11430 DATA END
11440 DATA *08AISLE 1 - CRACKERS
11450 DATA WHEAT THINS,BREAD CRUMBS,CROUTONS
11460 DATA RITZ CRACKERS,SALTINES,GRAHAM CRACKERS
11470 DATA ------------,END
11480 DATA *08FRONT OF STORE - COLD CUTS - FROZEN FISH
11490 DATA BACON,BOLOGNA,SLICED HAM
11500 DATA TURKEY,SHRIMP,---------FISH
11510 DATA ------------,END
20000 DATA *END


DONE
```

MANAGEMENT SCIENCE OPERATIONS RESEARCH

TRS-80™ MATHEMATICAL PROGRAMMING SYSTEM.

A collection of programs which implement the simplex algorithm, the transportation algorithm, and a network flow optimization algorithm. (Comes with documentation.) **$25.00**

TRS-80™ DECISION SYSTEM.

A collection of programs which implement break-even analysis, decision analysis, insurance analysis, element-ordering, and game theory. (Comes with documentation.) **$25.00**

TRS-80™ SIMULATION SYSTEM.

A program which facilitates the simulation of continuous dynamic systems described by differential or difference equations. Provides for integration, printing and plotting of the output on video or printer. (Comes with documentation.) **$25.00**

SOFTWARE ENGINEERING SYSTEMS INC. ✓238

TRS-80 is a trademark of the Radio Shack Division of Tandy Corporation

VISA

3214 75th Street
Lubbock, Texas 79423 (806) 792-9310

"Consultants in Software and Systems Engineering"

Computers & Gambling Magazine*

PRESENTS:

PROBABILITY HANDICAPPING DEVICE I

A 16K BASIC PROGRAM FOR:

HORSE RACE HANDICAPPING!

This amazing program was written by a professional software consultant to TRW Space Systems and is being introduced by the publishers of Computers and Gambling Magazine. "PHD-1" is a large complex basic program requiring a full 16K. It is carefully human factored for easy use. PHD-1 is a comprehensive horse racing system for spotting overlays in thoroughbred sprint races (less than 1 mile). You simply sit down with your computer and the Racing Form the night before the race and answer 5 or 6 questions about each horse's past performance. Your 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.

Statistics for thousands of horses were used to develop this handicapping system. The appendix of the manual contains a detailed tab run of a 100 consecutive race system workout showing an **amazing 45% positive return** (45¢ for each $1.00 wagered). A graph is also included showing PHD-1's close fit to the ideal predicted probability vs. actual win percentage curve.

This program features: □ Win probability and odds for each horse □ Verification display of each horse's parameters prior to entry for easy error correction □ Bubble-sort routine for final display □ Facility for line printer output □ Cassette ARCHIVE routine to store PHD-1's output for later analysis □ Complete users manual

The user's manual may be ordered seperately for your purusal for $7.95 and will be credited if you purchase PHD-1.

PHD-1 User's Manual and 16K Cassette for: Apple II Applesoft, Challenger (Specify Type), TRS-80† Level II, Pet **$29.95**

Make checks payable to: Ca. res. add 6%

✓193 **JOE COMPUTER**
22713 Ventura Blvd., Suite F, Woodland Hills, CA 91364

*BE A WINNER. Get on the Computers and Gambling Products Magazine mailing list for $3.00 and receive available back issues.

†TRS-80 is a registered trademark of Tandy Corporation

HERE'S HOW! 1. Input data to secretary 2. Into computer 3. Printout back to me.

PRESTO! Subdivision calcs in 1/2 the time of a programmable calculator.

OUR MICROCOMPUTER SOFTWARE DOES:

- TRAVERSE GEOMETRY
- COORDINATE GEOMETRY
- PROFILE GRADES
- RADIAL STAKEOUT
- CIRCULAR CURVE GEOMETRY
- CIRCULAR CURVE STAKEOUT

DEVELOPED AND SUPPORTED BY ANDREW MACHEN REGISTERED CIVIL ENGINEER.

Complete Program Package With Instruction Manual **$260**

MICRO COMP SOFTWARE SYSTEMS CIVIL ENGINEERING ✓95

Call or write for FREE INFORMATION PACKET (714) 755-4033
143c So. CEDROS, SOLANA BEACH CA. 92075

In the beginning ..... your computer was to make life easy. Then you found that you had to become a programmer to make your computer do the tasks you want it to do.

Now, you can relax! The ***CREATOR*** is here. This program will create ***BASIC,*** error free program text for you. All you need to do is answer the questions. The Creator even checks for errors.

The Creator will generate reports, perform calculations, and even interface with other program routines. No need to become a professional computer programmer. The Creator, under your command, will create custom programs for you.

The ***CREATOR*** is available on disk for the TRS-80 model One and Two. Also available for the Apple II and CP/M. Complete with an easy to read, well documented bound manual.

✓37

IJG COMPUTER SERVICES
369 N. Mountain Ave. • Suite B • Upland, CA 91786 U.S.A.

TRS-80 IS A REGISTERED TRADEMARK OF TANDY CORP.

A cryptic look at coding.

An Article Called Intrepid

Buzz Gorsky
2449 Derbyshire Rd.
Cleveland, OH 44106

If I got 55515255545663877991905642733645398979814471704262335640858792628258503766784158428988596451348266424560544648656883735059524456623893368354 as a message, I would probably say "Yes that's true." Of course I would have the benefit of knowing that the message decodes to "*80 Microcomputing* contains many interesting programming articles."

How did I know that?

In his book *A Man Called Intrepid* (Ballantine Books, New York, 1976), William Stevenson describes some of the effort during World War II by the British to decode German military transmissions. The German code changed with the transmission of each character of a message. Another complicating factor was that a single code machine might send different codes to different recipients.

I have written a series of three programs which provide the same type of coding and decoding. The first program will generate an original code; the second will encode a message, changing the code for each character of the message: and the third will decode an encoded message.

Simple Principles

The principles of the code are quite simple. A list called C(I) is maintained in memory. It has a code value for each ASCII value used by the TRS-80. This represents the initial code at the start of a message.

Each message consists of a string of numerals that may be considered as pairs, each pair representing an encoded ASCII value. The first six pairs are special. These six are chosen by selecting six single-digit numerals at random and using the ASCII representation of each.

Three of these six carry no significance and exist to confuse things. However, the value of the encoded digit corresponding to the second pair, the third pair and the fifth pair are used by the program.

The value obtained from the sixth pair tells the program how many times to adjust the code in a given direction. Direction implies that the code can be changed by adding or subtracting a fixed amount from each C(I). The value of the second pair is the amount which will be added to C(I). The value of the fifth pair is the amount which will be subtracted.

> *"The German Code changed with the transmission of each character of a message."*

As the message is generated values are either added or subtracted from C(I) each time a character is encoded. Thus the code changes with each letter. Further, the way the code changes depends on three of the first six data pairs sent.

In Program Listing 1, (lines 1-90), line 5 prints a big question mark on the screen as a symbol for the encoding program. In line 20 string space is set aside, variables are typed and C is dimensioned.

Line 30 generates a random number between 32 and 95, that is, between the limits for the ASCII set. The first value chosen is assigned to C(1), and each subsequent new value will be assigned to each subsequent C(I) until all values from 32 to 95 are represented.

In line 40, if a new random number (N) is the same as any of the C(I) already in memory, then that N is discarded and a new random number is picked. J counts the number of C(I) existing and when J equals 63 the task is done. Line 50 prints J on the screen so that you can follow the progress of the program.

In line 60 the C(I) are represented as a string Y. This string consists of the concatonated strings of the numerical values of each C(I). Then, in lines 70 and 80, Y is saved on tape. The C(I) could be saved on tape directly, but this method is much faster.

Saving to Tape

The second program, shown in Listing 2, (lines 100-410), acquires the message, encodes it and provides for saving the message on tape and saving the final code on tape. Line 110 takes care of the housekeeping. It retrieves the code from tape and separates Y into the C(I) again.

Line 120 tells the operator that the message can be typed and must not exceed 100 characters. This is a somewhat arbitrary figure, but the message will be stored as a single string and that cannot exceed 255 characters. Since each character of the message is coded as a two-character string representing an encoded ASCII value, the message length can actually reach 127 characters.

The operator is also told that hitting * will erase a character just typed and that hitting / will terminate the message. As the message is entered, the count will change, and the screen will show how many characters are left. The message will appear on the screen in its text and ASCII forms.

In line 130, I is set equal to 94 which is the number of characters remaining. Remember that the first six characters are generated by the program. Line 140 generates the start of the string. A value K is generated randomly as an integer between zero and nine, and a string is made. The string will include a space and a numeral, since the space was saved for the symbol K. Only the right character is taken to get a single character string.

The ASCII value of the string is found, and then a string of the ASCII value is formed. All of that is done by X = RIGHT$(STR$(ASC(RIGHT$(STR$(K),1))),2). It may seem like a lot of work, but this results in a two-character string that represents the ASCII value of K.

While each of the six Ks is randomly generated, Y is set equal to Y + X. Y is an ever growing string of all the Xs. In line 150 each character of the message is entered using the INKEY$ function.

If an * is hit, I is increased so that the previous character will be replaced by the next character entered. When / is hit, or when I is zero, the message is ended.

In line 180 the encoding process begins. Y is the uncoded message represented as a string of ASCII values. Since each element of Y is two characters long, J, which equals one-half the length of Y, indicates how many ASCII pairs there are.

In lines 190 to 225 the second, third and fifth ASCII pairs are evaluated to see what numerals they represent. I1, I2 and I3 are set equal to these numerals. I1 will eventually be added to C(I) values when the code is changed, and I2 will be subtracted. I3 will tell how many addition functions to do before switching to subtraction, or vice versa.

In line 230 the construction of Z begins. Z will be a character string representing the encoded ASCII values, and the final message in code. It begins as the first 12 characters of Y, then the rest is the encoded values obtained from the message.

In line 240, X is set equal to two-character string taken from the middle of Y. As I moves along, X will be set sequentially to be each ASCII pair in Y, beginning with the seventh pair, which is where the text of the message actually begins. K is set equal to the value of X minus 32, and then equal to C(K), which is the coded value corresponding to K. For example, X = 32, the value of X would be 32 and K would first be set equal to 0 and then to C(0). That would be the coded ASCII value for the space character.

Lines 270 to 350 check how many times the code has been shifted in a given direction and then make the appropriate shift. K0 is a variable which keeps track of whether the code is being shifted by addition or subtraction. When K0 is 0 the code is being shifted by addition, when it's 1, the code is being shifted by subtraction.

K1 keeps track of how many shifts have been made. This number is compared with I3 and when it equals 13 it is then reset to 0. K0 receives a new value and the opposite shift begins. In 290 to 320 the code is adjusted by adding I1 to each C(I). If any C(I) is then larger than 95 the value is adjusted by subtracting 64. In this way the complete set is maintained.

In lines 330 to 360 the code is adjusted by subtracting I2 from each C(I) value and adding 64 to any which would be less than 32 otherwise. In line 370, I is printed on the screen so that you can see the progress as each character of the code is generated. Following line 380 the coded message and the new code can be saved on tape.

Decoding

Program Listing 3 begins on line 500. This program reads in the original code and the coded message and then decodes the message. It is analogous to many parts of the second program. First it looks at the first six ASCII pairs and from the second, third and sixth pairs, obtains the values of I1, I2 and I3. It will know how the code is shifted during decoding.

Next it decodes each ASCII pair beginning with the seventh pair. This is done by finding which C(I) equals the value of the particular ASCII pair.

I = 32 is the decoded ASCII value and CHR$(K = 32) on line 650 finds the decoded character. The string Y is built up of these characters, so that Y will be the decoded message.

K0 and K1 are used, as in the encoding routine, to keep track of how the code is changed. It is changed with the decoding of each character, so that there is a correct code for decoding the next character.

Finally in line 770 the decoded message is displayed. The code is prepared and stored on tape as a single string again. ■

Program listing

```
1 REM 'ENIGMIZER'--CRYPTOGRAPHIC PROGRAM BY BUZZ GORSKY
2 REM FOR 80 MICROCOMPUTING
5 CLS:Y=1:FOR X=35 TO 80:SET(X,Y):NEXT:FOR Y=1TO20:SET(
     X,Y):NEXT:Y=20:FOR X=80 TO 48 STEP -1:SET(X,Y):NEX
     T:X=48:FOR Y=20 TO 35:SET(X,Y):NEXT:SET(X,Y+1):SET
     (X,Y+2):FOR I=1TO5000:NEXT
10 CLS:REM ROUTINE TO GENERATE ORIGINAL CODE
20 CLEAR450:DEFINT C,I,J:DEFSTR X,Y,Z:DIM C(63):RANDOM
30 N=RND(95):IF N<32 THEN 30
40 FOR I=0 TO J:IF N=C(I) THEN 30
50 NEXT I:C(J)=N:IF J<63 THEN J=J+1:PRINT@540,J:GOTO30
60 FOR I=0 TO 63:X=RIGHT$(STR$(C(I)),2):Y=Y+X:NEXT
70 INPUT"PREPARE TAPE TO SAVE CODE";Z
80 PRINT#-1,Y
90 END
100 REM ROUTINE TO ENCODE MESSAGE THEN SAVE MESSAGE, EN
     CODED MESSAGE, AND NEW CODE
110 CLEAR 1000:DEFINT C,I,J,K:DEFSTR X,Y,Z:DIM C(63):RA
     NDOM:INPUT"TAPE READY TO INPUT CODE";X0:INPUT#-1,Y
     :FOR I=0 TO 63:C(I)=VAL(MID$(Y,I*2+1,2)):NEXT:Y=""
120 CLS:PRINT "ENTER MESSAGE. DO NOT EXCEED 100 CHARACT
     ERS.":PRINT "END WITH'/'; USE '*' TO DELETE LAST L
     ETTER"
130 PRINT "CHARACTERS REMAINING--":PRINT:I=94
140 FOR J=1TO6:K=RND(9):X=RIGHT$(STR$(ASC(RIGHT$(STR$(K
     ),1))),2):Y=Y+X:NEXT J
145 PRINT@152,I:IF I=0 THEN 170
150 X=INKEY$:IF X="*" THEN 155 ELSE IF X="/" THEN 170 E
     LSE IF X="" THEN 150 ELSE 160
155 Y=LEFT$(Y,LEN(Y)-2):Z=LEFT$(Z,LEN(Z)-1):I=I+1:GOTO1
     45
160 Z=Z+X:Y=Y+RIGHT$(STR$(ASC(X)),2):PRINT@512,Y:I=I-1:
     PRINT@768,Z:GOTO145
170 CLS
180 J=LEN(Y)/2:FOR I=2 TO 5:IF I=4 THEN NEXT I
190 X=MID$(Y,2*I-1,2)
200 IF I=2 THEN I1=VAL(CHR$(VAL(X)))
210 IF I=3 THEN I2=VAL(CHR$(VAL(X)))
220 IF I=5 THEN I3=VAL(CHR$(VAL(X)))
225 NEXT I
230 Z=LEFT$(Y,12):FOR I=7 TO J
240 X=MID$(Y,2*I-1,2)
250 K=VAL(X)-32:K1=K1+1
```

Program Continued

```
260 K=C(K):Z=Z+RIGHT$(STR$(K),2)
270 IF K0<>0 THEN 280 ELSE IF K1<I3 THEN 290 ELSE K1=0:
     K0=1:GOTO330
280 IF K1<I3 THEN 330 ELSE K1=0:K0=0
290 FOR J1=0 TO 63
300 IF C(J1)+I1>95 THEN C(J1)=C(J1)-64
310 C(J1)=C(J1)+I1
320 NEXT J1:GOTO370
330 FOR J1=0 TO 63
340 IF C(J1)-I1<32 THEN C(J1)=C(J1)+64
350 C(J1)=C(J1)-I2
360 NEXT J1
370 PRINT@500,I:NEXT I
380 CLS:PRINT"ENCODING COMPLETE":INPUT"TAPE READY TO SA
     VE CODED MESSAGE";X0:PRINT#-1,Z:INPUT"LINEPRINT OF
      CODED MSG";X0:IF X0="Y" THEN LPRINT Z
390 FOR I=0 TO 63:Y0=Y0+RIGHT$(STR$(C(I)),2):NEXT
400 INPUT"TAPE READY TO SAVE NEW CODE";Z:PRINT#-1,Y0
410 END
500 REM ROUTINE TO DECODE MESSAGE AND SAVE NEW CODE
510 CLEAR 1000:DEFINT C,I,J,K:DEFSTR X,Y,Z:DIM C(63)
520 INPUT"TAPE READY TO ENTER CODED MESSAGE";X:INPUT#-1
     ,Z:INPUT"TAPE READY TO INPUT CODE";X0:INPUT#-1,Y0
530 FOR I=0 TO 63
540 C(I)=VAL(MID$(Y0,2*I+1,2)):NEXT
550 J=LEN(Z)/2:FOR I=2 TO 5:IF I=4 THEN NEXT I
560 X=MID$(Z,2*I-1,2)
570 IF I=2 THEN I1=VAL(CHR$(VAL(X)))
580 IF I=3 THEN I2=VAL(CHR$(VAL(X)))
590 IF I=5 THEN I3=VAL(CHR$(VAL(X)))
600 NEXT I
610 FOR I=7 TO J
620 X=MID$(Z,2*I-1,2)
630 K2=VAL(X):K1=K1+1
640 FOR K=0 TO 63:IF C(K)<>K2 THEN NEXT K
650 Y=Y+CHR$(K+32)
660 IF K0<>0 THEN 670 ELSE IF K1<I3 THEN 680 ELSE K1=0:
     K0=1:GOTO720
670 IF K1<I3 THEN 720 ELSE K1=0:K0=0
680 FOR J1=0TO63
690 IF C(J1)+I1>95 THEN C(J1)=C(J1)-64
700 C(J1)=C(J1)+I1
710 NEXT J1:GOTO 760
720 FOR J1=0 TO 63
730 IF C(J1)-I1<32 THEN C(J1)=C(J1)+64
740 C(J1)=C(J1)-I2
750 NEXT J1
760 PRINT@564,I:NEXT I
770 CLS:PRINT"DECODED MESSAGE:":PRINT:PRINTY
780 Y0="":FOR I=0TO63:Y0=Y0+RIGHT$(STR$(C(I)),2):NEXT
790 INPUT"IS TAPE READY TO RECORD NEW CODE";X0:PRINT#-1
     ,Y0


DONE
```

PRIAM Hard Disks Now Available from SIRIUS SYSTEMS!

PRIAM's high-performance, low-cost Winchester disc drives speed up throughput and expand data storage from 20 megabytes to 154 megabytes. And a single controller can be used to operate 14-inch-disc drives with capacities of 33, 66, or 154 megabytes or floppy-disc-size drives holding 20 and 34 megabytes. So it's easy to move up in capacity, or reduce package size, without changing important system elements or performance.

- *Fast, Linear Voice Coil Positioning*
- *10 ms track-to-track positioning*
- *Fully servoed head positioning*
- *Dedicated servo tracks*
- *DC Power required only!*
- *Simple, parallel interface*
- *Optional SMD interface*
- *50 ms Average Positioning time*
- *90 ms Maximum Positioning Time*
- *5.4 ms Average Latency*

THE PRIAM LINEUP

Model/Disc Size	Capacity	Size	Weight	Price
DISKOS 3350 (14")	33Mbytes	7" × 17" × 20"	33 lbs.	$2995
DISKOS 6650 (14")	66 Mbytes	7" × 17" × 20"	33 lbs.	$3749
DISKOS 15450 (14")	154 Mbytes	7" × 17" × 20"	33 lbs.	$4695
DISKOS 2050 (8")	20 Mbytes	4.62" × 8.55" × 14.25"	20 lbs.	$2995
DISKOS 3450 (8")	34 Mbytes	4.62" × 8.55" × 14.25"	20 lbs.	$3745
DISKOS 570	5.3 Mbytes	floppy-size	(low)	(low)
DISKOS 1070	10.6 Mbytes	floppy-size	(low)	(low)

All PRIAM DISKOS Drives have a Transfer Rate of 1.03 Mbytes/Sec.
Optional SMD interface available for $150.

SIRIUS SYTEMS offer cases and enclosures for all PRIAM Hard Disk Drives. All 14" Winchester Drives will mount in our 14" Standard Case. The 8" Winchesters have two alternatives: a single drive case and a dual drive case. All SIRIUS SYSTEMS Winchester drive cases include Power Supply, internal cabling, switches, fan, extra AC outlet (not switched, but fused) and possess very adequate ventilation. Drive addressing is done on the rear of the Case and not on the drive itself to provide ease of use during operation. All WINCHESTER DRIVE Cases are Warranted for a full year and come in our standard blue-black color scheme. Consult us for current availability and pricing.

Remex RFD 4000/4001 8" Floppy Disc Drives Double sided... Double density!!

$549.95

RFD 4001, $569.95

Offers quality and features found in drives costing much more! ■ Single or Double Density ■ Double-Sided Drive ■ Door Lock INCLUDED ■ Write-Protect INCLUDED ■ 180 Day Warranty ■ Compatible with Shugart 850/851 ■ Low Power Operation ensures LONGER LIFE!! ■ Model RFD 4001 offers Data and Sector Separator

RFD 4000/4001 Technical Manual	6.95
Connector Set #3 (AC, DC, Card Edge)	10.95
Connector Set #4 (AC and DC)	2.95
RFD 4000C/8 Cabinet (for use with Power Modules)	29.95

Remex 1000B ... If you've been looking for a less expensive floppy disc drive, but not wanting to sacrifice quality — this is it!

$419.95

You get both in the Remex 1000B! For only $419.95 look **at what you get:** ■ 8" Floppy Drive ■ Single or Double Density ■ **Hard or Soft Sectoring** ■ **Media** Protection Feature ■ Single Density **Data Separator** ■ 180 Day Factory Warranty

Door Lock Option	$19.95
Interface Adapter (REMEX-to-Shugart)	$14.95
Write Protect Option	$19.95
Connector Set #1 (AC, DC, & Card Edge)	$18.95
RFD 1000B Technical Manual	$5.95
RFD 1000B CASE (for use (with Power Modules)	$29.95

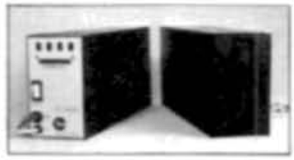
SIRIUS 8" DISK POWER MODULES

The Single and Dual Drive Power Modules are designed to provide DC and (switched) AC power for one (the Single Drive Power Module) or two (the Dual Drive Power Module— the DDPM will power three RFD 4000s or 4001s) 8" Floppy Disk Drives. Many features are included for safe and reliable operation and the Power Modules come with our standard 180 day WARRANTY (the Open Frame Power Supply warranty is for 2 years). All Power Modules will work with either the RFD 4000C/8 or RFD 1000B case (color schemes match also).

Dual Drive Power Module (DDPM)	$139.95
Single Drive Power Module (SDPM)	119.95

SIRIUS 80+

Perfect Add-Ons for Your Computer System!



The SIRIUS SYSTEMS 80+ Series of Floppy Disk add-ons are designed to provide unmatched versatility and performance for your computer. Consisting of four different add-ons, there is a 80+ Series Floppy Disk to meet your need. All 80+ Series Floppy Disk are compatible with the TRS-80* and come ready to plug in!

COMMON CHARACTERISTICS

- *5 ms track-to-track access time*
- *Auto-eject*
- *180 day WARRANTY*
- *Exceptional speed stability — 1½%*
- *Single density (FM) or double density (MFM/M²FM)*
- *Ultra high reliability*
- *2 year Power Supply Warranty*
- *Mix any or all 80+ Series on the same cable!*
- *Includes user accessible plugboard for drive reconfiguring*

SPECIFIC CHARACTERISTICS

The **SIRIUS 80+1** is a single sided, 40 track, highly reliable Floppy Disk add-on. Offering 5 more tracks than the Radio Shack model, it cost $140 less! Formatted data storage is 102K/204K bytes single/double density.

SIRIUS 80+1 .... **$359.95**

The **SIRIUS 80+2** is a dual sided, 70 track (35 per side), highly versatile Floppy Disk unit. It appears to the TRS-80* as TWO 35 track drives, yet COST LESS THAN HALF THE PRICE! Even greater savings result, since data is recorded on both sides of the media instead of only a single side. Using the plug board, it may be reconfigured for other computer systems! (The 80+2 operates as Drive 0 and any of the other three addresses (with the sandard Radio Shack Cable) or as any of four drives (with the SS Standad Cable).) Formatted data storage is 80.6K/161.2K bytes single/double density.

SIRIUS 80+2 .... **$449.95**

The **SIRIUS 80+3** is a single sided, 80 track, "Quad" density Floppy Disk unit. Offering 2½ times the storage of a Standard Radio Shack drive, the 80+3 greatly reduces the need for diskettes correspondingly. Additionally, because of the increased storage and faster track-to-track access time, the 80+3 allows tremendously increased throughput for disk based programs!!! The 80+3 INCLUDES SIRIUS's TRAKS-PATCH on Diskette. Formatted data storage is 204K/408K bytes single/double density.

SIRIUS 80+3 .... **$489.95**

The **SIRIUS 80+4** Floppy Disk add-on is a double sided, 160 track (80 per side), 5¼" monster! The ultimate in state-of-the-art 5¼" Floppy Disk technology, to 80+4 is seen by the TRS-80* as two single sided disk drives, each with 80 tracks. Thus, in terms of capacity one 80+4 is equivalent to 4½ standard Radio Shack drives — a savings of over 73% (not to mention diskettes!!!). (With a double density converter, the available memory is huge!) The 80+4 is similar to the 80+2 in that it arrives configured as Drive 0 and any of the other three addresses (with the standard Radio Shack Cable) or as any of four drives (with the SS Standard Cable). The 80+4 INCLUDES TRAKS-PATCH on Diskette. (The plug board is also included.) Formatted data storage is 408K single density or 816K bytes double density.

SIRIUS 80+4 .... **$624.95**

All 80+ Series Floppy Disk add-ons operate a 5 milliseconds track-to-track access time (eight times faster than the SA 400) but are Expansion Interface Limited to 12 milli-seconds for the TRS-80*.

*TRS-80® Tandy Corp.

MPI 51/52 ...

A Great Reliable Mini-Drive!

- *Fast! 5ms track to track access*
- *Exclusive Pulley-Band Design*
- *Unique Door/Ejector Mechanism*
- *Reliable 1½% Speed Stability*
- *Single/Double Density Operation*
- *Industry/ANSI Standard interface*

MPI 51
(Single Head, 40 tracks, 120K/240K bytes Single/Double Density**) **$259.95**

MPI 52
(Dual Head, 70 tracks, (35/side), 218.8K/437.5K Single/Double Density**) **$349.95**

MPI 91/92 ... NEW STATE-OF-THE-ART DISK DRIVE!

MPI 91
(Single Head, 80 tracks, 240K/480K Single/Double Density**) **$389.95**

MPI 92
(Single Head, 160 tracks (80/side), 480K/960K Single/Double Density**) **$499.95**

**Unformatted data storage

Introducing the Versatile, Low-Cost OMEGA Series Controller

As new technological advances bring down the cost of fast, reliable mass data storage, the need for an inexpensive, versatile controller have become greater and greater. To meet this need, SIRIUS SYSTEMS' OMEGA Series Controller was designed.

The SIRIUS OMEGA Series Controller Module utilizes an **on-board microprocessor** to mediate data transfer to a wide variety of peripherals from an equally wide variety of host computer systems. Up to four Winchester Hard Disks (8" or 14"), four 5¼" Floppy Disks Drives and/or up to eight 8" Floppy Disk Drives may be **in use at one time.** Host systems interfacing is accomplished via a parallel or a serial interface. With the addition of a Personality module, the OMEGA Series Controller Module is directly **compatible with many popular computer systems** (among them the TRS-80*, Apple, Heath, and others). Provision is made for the addition of a streaming tape drive, also.

SPECIFIC HARDWARE FEATURES INCLUDE:

- *Control of up to twelve Floppy Disk Drives (eight 8" and/or four 5¼")*
  - *8" and/or 5¼" Disk Drive Utilization*
  - *Single (FM) or Double (MFM) density data storage*
  - *Hard or Soft sectored diskette usage*
  - *Utilization of "Quad" density (96 tpi) 8" or 5¼" Disk Drives*
- *Control of up to four WINCHESTER type PRIAM DISKOS Disk Drives*
  - *8" or 14" may intermix on the same cable*
  - *Accommodates 8" and/or 14" drives of 5.3Mbytes to 154Mbytes*
  - *Ultra-Fast data transfers*
- *Extremely flexible host-controller interfacing*

SPECIFIC SOFTWARE FEATURES INCLUDE:

- Dynamic format modifications via command words
- Extremely flexible format acceptance for unusual data storage formats
- Easily interfaces to standard operating systems (TRS-DOS*, CP/M***, etc)
- Operates in either get/put sector mode or data string mode
- Performance parameters may be changed by EPROM replacement or Dynamic Reprogramming

Dedicated systems cards are also available on a limited basis for the STD-BUS and the S-100. These cards feature shared memory also (again, software selectable) in addition to the regular OMEGA Series Controller Module features. Consult SIRIUS SYTEMS for current price and availability for the entire line of OMEGA Series Memory Units and Controllers. **Dealer inquiries are invited.**

SIRIUS SYSTEMS
7528 Oak Ridge Highway
Knoxville, Tennessee 37921 ✓67

TO ORDER CALL (615) 693-6583

Phone Orders Accepted 9AM-7PM (ESDT)

We accept **MC, VISA, AE, COD** (requires Certified Check, Cashier's Check or Cash) and **Checks** (personal checks require 14 days to clear). **SHIPPING AND HANDLING:** $7.00 per Floppy Disk Drive or 80+ Module ■ 5% for other items (any excess will be refunded) ■ **Foreign Orders** add 10% for Shipping & Handling. Payment in U.S. currency ■ Tennessee residents add 6% Sales Tax ■ **VOLUME DISCOUNTS AVAILABLE**

✓Reader Service—see page 226

Bach to BASIC!

Music Note Recognition

J. David McClung
P.O. Box 1590
Dallas, TX 75221

My wife and I bought our TRS-80 partly to help with our children's educations. We didn't realize then how very little educational software was available. I turned, therefore, to writing my own.

Music Note Recognition is a program I wrote to teach my two oldest children how to read music. The program displays either a treble clef or a bass clef with a single whole note. The child is asked to name the note within a previously selected time. After 20 notes, the responses are evaluated and the incorrect ones reviewed.

Six Steps

I follow six steps when debugging a program. Step one clearly defines the lesson. Children learn more quickly when lessons are simple. For example, in Music Note Recognition I could have included sharps, flats or fractional notes. Instead, I limited the lesson to whole natural notes.

Step two presents the lesson facts. Most educational programs I have seen fail by asking questions before the student has a chance to study the material. My children quickly lose interest in a program, if it asks them several questions in a row that they cannot answer. Programs are more effective if the lesson is presented before testing begins.

In Music Note Recognition, line 240 lets the child review the lesson. In the subroutine beginning at line 1400, each note is presented with its name. The child can study as long as necessary before beginning the test.

Step three tests the child's knowledge. The test is the heart of any educational program. The student should be allowed to demonstrate his knowledge of the material.

I try to make the test questions active, varied and interesting. My children lose interest, if the program repeats the same questions in the same sequence each time the program is run. I use the TRS-80 random generator to vary the order of the questions. I also include protection to prevent repeating the same question too often.

In Music Note Recognition, lines 420–440 select the sequency of questions. Each question is compared to previously selected questions to prevent duplication. Thus, 20 of the 33 possible questions are presented each time the program is run, but in different sequences.

Children respond best to programs that have a lot of action on the CRT. For example, this music program redraws the entire staff for each test question rather than merely moving the note. The extra action helps hold the child's attention.

Include Challenges

If the program is going to be useful for more than a few runs, the test must become more difficult as the student masters the lesson. You can challenge the student by limiting the time allowed for each answer.

The value of H selected in line 320 determines the time allowed. In my computer, I must multiply the number of seconds selected by 40. If the program is used without a keyboard debounce routine, the 40 must be increased. The correct value can be established by experimentation.

The program responds to each answer immediately, correcting a child's responses. This feature makes the computer especially valuable as a teaching tool. Lines 560–580 confirm the answer given or tell the child what the correct answer was.

Step four evaluates the results. An evaluation should encourage the student to study. An effective educational program will evaluate how well the child has learned the lesson and encourage the child to perform better. I try to present the evaluation in a positive manner.

For my older children, I use their school's grading scale. They are shown the percentage correct and a letter grade. For younger children, I use "Great," "Good," "Fair" or "Need more

the electric pencil II™

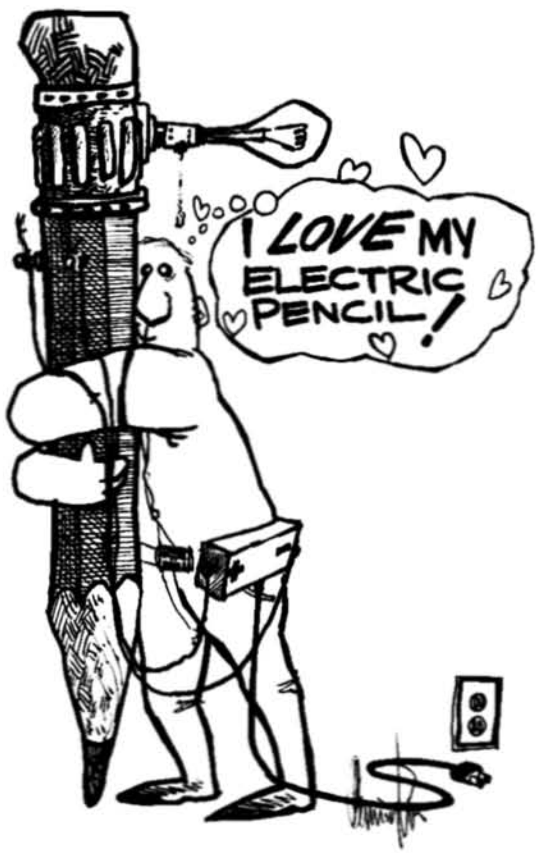
© 1980 Michael Shrayer

for the TRS-80 Model II* Computer

The Electric Pencil is a Character Oriented Word Processing System. This means that text is entered as a continuous string of characters and is manipulated as such. This allows the user enormous freedom and ease in the movement and handling of text. Since lines are not delineated, any number of characters, words, lines or paragraphs may be inserted or deleted anywhere in the text. The entirety of the text shifts and opens up or closes as needed in full view of the user. Carriage returns as well as word hyphenation are not required since each line of text is formatted automatically.

As text is typed and the end of a screen line is reached, a partially completed word is shifted to the beginning of the following line. Whenever text is inserted or deleted, existing text is pushed down or pulled up in a wrap around fashion. Everything appears on the video display screen as it occurs thereby eliminating any guesswork. Text may be reviewed at will by variable speed or page-at-a-time scrolling both in the forward and reverse directions. By using the search or the search and replace function, any string of characters may be located and/or replaced with any other string of characters as desired. Specific sets of characters within encoded strings may also be located.

When text is printed, The Electric Pencil automatically inserts carriage returns where they are needed. Numerous combinations of Line Length, Page Length, Character Spacing, Line Spacing and Page Spacing allow for any form to be handled. Right justification gives right-hand margins that are even. Pages may be numbered as well as titled.



the electric pencil

—a Proven Word Processing System

The TRSDOS versions of The Electric Pencil II are our best ever! You can now type as fast as you like without losing any characters. New TRSDOS features include word left, word right, word delete, bottom of page numbering as well as extended cursor controls for greater user flexibility. BASIC files may also be written and simply edited without additional software.

Our CP/M versions are the same as we have been distributing for several years and allow the CP/M user to edit CP/M files with the addition of our CONVERT utility for an additional $35.00. CONVERT is not required if only quick and easy word processing is required. A keyboard buffer permits fast typing without character loss.

	CP/M	TRSDOS
Serial Diablo, NEC, Qume	$ 300.00	$ 350.00
All other printers	$ 275.00	$ 325.00

The Electric Pencil I is still available for TRS-80 Model I users. Although not as sophisticated as Electric Pencil II, it is still an extremely easy to use and powerful word processing system. The software has been designed to be used with both Level I (16K system) and Level II models of the TRS-80. Two versions, one for use with cassette, and one for use with disk, are available on cassette. The TRS-80 disk version is easily transferred to disk and is fully interactive with the READ, WRITE, DIR, and KILL routines of TRSDOS.

TRC	Cassette	$ 100.00
TRD	Disk	$ 150.00

✓255

Features

TRSDOS or CP/M Compatible * Supports Four Disk Drives * Dynamic Print Formatting * Diablo, NEC & Qume Print Packages * Multi-Column Printing * Print Value Chaining * Page-at-a-time Scrolling * Bidirectional Multispeed Scrolling * Subsystem with Print Value Scoreboard * Automatic Word & Record Number Tally * Global Search & Replace * Full Margin Control * End of Page Control * Non Printing Text Commenting * Line & Paragraph Indentation * Centering * Underlining * Boldface

VISA master charge

*TRS-80 is a registered trade mark of Radio Shack, a division of Tandy Corp.

mss

MICHAEL SHRAYER SOFTWARE, INC.
1198 Los Robles Dr.
Palm Springs, CA. 92262
(714) 323-1400

practice." For pre-schoolers I use a smiling face or a sad face.

Reviews and Invitations

Step five reviews incorrect answers. When the child answers a test question incorrectly, the computer provides the correct one. In addition, the questions answered incorrectly are reviewed following the evaluation. Line 800 repeats each missed question, until answered correctly.

Step six concludes with an invitation to try again. I try to end every educational program this way. The student should be praised for good work, but encouraged to achieve higher scores or shorter response times. Such challenges keep the program interesting.

My children have enjoyed learning by computer so much that my 12-year-old daughter, LeEtta, and my 10-year-old son, Denny, are beginning to write their own simple programs. The educational usefulness of the TRS-80 is limited only by the availability of software. With a little practice any computer owner can write programs that challenge and teach the student. ■

Program Listing

```
40 CLEAR 150:GOTO 210: 'SKIP INPUT ROUTINE
100 'INPUT ROUTINE AND TIMING LOOP
110 FOR L=1 TOH:I$=INKEY$
120 IF I$>="A" AND I$<="G"PRINT I$;:RETURN
130 IF I$<>"" PRINT @0,"ANSWER A,B,C,D,E,F OR G
    "
140 NEXT :RETURN
200 'BEGIN PROGRAM
210 CLS:M$=STRING$(63,"-"):DIMA(20):DIME(20):R=0:W=0:K$
    =STRING$(3,CHR$(191))
220 CLS:PRINT@20,"MUSIC NOTE RECOGNITION":PRINT:PRINT@7
    4,"(C) 1980 BY DAVE MCCLUNG, RICHARDSON, TX"
230 PRINT:PRINT"THIS PROGRAM IS DESIGNED TO TEACH MUSIC
     STUDENTS THE NAMES":PRINT"OF EACH LINE AND SPACE.
    ":PRINT:PRINT"THE QUIZ WILL CONSIST OF 20 QUESTION
    S.":PRINT:PRINT
240 PRINT:INPUT "DO YOU WANT TO REVIEW ";Z$
250 IF LEFT$(Z$,1)="Y" GOTO 1400
300 'GET INFORMATION
310 CLS:INPUT "WHAT IS YOUR NAME ";N$
320 PRINT:INPUT "HOW MANY SECONDS FOR EACH QUESTION ";H
    :H=H*40
330 '59 SECONDS MAXIMUM
340 IF H>2360 PRINT"YOU CAN'T HAVE THAT LONG":PRINT:GOT
    O 320
400 'SELECT TEST QUESTIONS -- NO REPEATS
410 PRINT:PRINT"STANDBY -- THE COMPUTER IS SELECTING TH
    E QUESTIONS"
420 FOR X=1 TO 20
430 A(X)=RND(23):FOR Y=0 TO (X-1):IF A(Y)=A(X) GOTO 430
     ELSE NEXT Y
440 NEXT X
500 'TEST
510 FOR G=1 TO 20:N=A(G)
520 GOSUB 1000:GOSUB 1110
530 IF C=1 GOSUB 1210 ELSE GOSUB 1310
540 IF S=924 OR S=156 PRINT@S,"--"+K$+"--"; ELSE PRINT@
    S,K$;
550 PRINT@0,"#";G;"  ";"WHAT IS THE NAME OF THIS NOTE";
    :GOSUB 110
560 PRINT@0,STRING$(63," "):IF I$=Q$ R=R+1:PRINT@0,"COR
    RECT":GOTO 590
570 IF I$<>Q$ AND I$<>""PRINT@0,"WRONG!  THE NOTE IS ";
    Q$;:P=P+1:E(P)=A(G)
580 IF I$="" PRINT@0,"YOUR TIME IS UP!  THE NOTE IS ";Q
    $;:P=P+1:E(P)=A(G)
590 GOSUB 1600:NEXT
600 'EVALUATE TEST RESULTS
610 CLS:PRINTCHR$(23):PRINT"TEST RESULT":PRINT:PRINTN$;
     " ,YOUR TEST RESULT IS:"
620 PRINT:PRINT"NUMBER CORRECT = ";R
630 W=20-R
640 PRINT"NUMBER MISSED  = ";W
650 PRINT"SCORE          = ";:U=((R/20)*100):PRINTUSING
     "###%";U
660 PRINT"GRADE          = ";
670 IF U<60PRINT"F"
680 IF U>59 AND U<70 PRINT"D"
690 IF U >69 AND U<80 PRINT"C"
700 IF U>79 AND U<90 PRINT"B"
710 IF U>89 PRINT"A"
720 PRINT:IF U=100 PRINT"YOU DID GREAT!":GOTO 900
800 'REVIEW THE QUESTIONS MISSED
810 PRINT:PRINT"TO RECHECK THE ONES YOU MISSED -- HIT E
    NTER";:INPUT Z$
820 CLS:FOR O=1 TO (20-R)
830 N=E(O):GOSUB 1000:GOSUB 1110
840 IF C=1 GOSUB 1210 ELSE GOSUB 1310
850 IF S=924 OR S=156 PRINT@S,"--"+K$+"--"ELSEPRINT@S,K
    $;
860 PRINT@0,"TRY THIS ONE AGAIN ";:GOSUB 110:PRINT@0,ST
    RING$(63," "):IF I$=Q$ PRINT@0,"CORRECT! ":GOTO 88
    0
870 PRINT@0,"WRONG!":GOSUB1600:GOTO 860
880 GOSUB1600:NEXT O
890 CLS:PRINTCHR$(23):PRINT"CONGRATULATIONS, ";N$:PRINT
    :PRINT:PRINT"YOU GOT THEM ALL"!GOSUB 1600::PRINT:P
    RINT"FINALLY":PRINT:GOSUB 1600:PRINT"AFTER MUCH EF
    FORT"
900 PRINT:PRINT"TO TAKE THE TEST AGAIN -- HIT ENTER":IN
    PUT Z$:RUN
1000 'SELECT NOTE
1010 ON N GOTO 1020,1021,1022,1023,1024,1025,1026,1027,
     1028,1029,1030,1031,1032,1033,1034,1035,1036,1037,
     1038,1039,1040,1041,1042,1043
1020 Q$="G":S=222:C=1:RETURN
1021 Q$="F":S=286:C=1:RETURN
1022 Q$="E":S=350:C=1:RETURN
1023 Q$="D":S=414:C=1:RETURN
1024 Q$="C":S=478:C=1:RETURN
1025 Q$="B":S=542:C=1:RETURN
1026 Q$="A":S=606:C=1:RETURN
1027 Q$="G":S=670:C=1:RETURN
1028 Q$="F":S=734:C=1:RETURN
1029 Q$="E":S=798:C=1:RETURN
1030 Q$="D":S=862:C=1:RETURN
1031 Q$="C":S=924:C=1:RETURN
1032 Q$="C":S=156:C=2:RETURN
1033 Q$="B":S=222:C=2:RETURN
1034 Q$="A":S=286:C=2:RETURN
1035 Q$="G":S=350:C=2:RETURN
1036 Q$="F":S=414:C=2:RETURN
1037 Q$="E":S=478:C=2:RETURN
1038 Q$="D":S=542:C=2:RETURN
1039 Q$="C":S=606:C=2:RETURN
1040 Q$="B":S=670:C=2:RETURN
1041 Q$="A":S=734:C=2:RETURN
1042 Q$="G":S=798:C=2:RETURN
1043 Q$="F":S=862:C=2:RETURN
1100 'PRINT STAFF
1110 CLS:PRINT@256,M$:PRINT:PRINTM$:PRINT:PRINTM$:PRINT
     :PRINTM$:PRINT:PRINTM$
1120 RETURN
1200 'DRAW TREBLE CLEF
1210 FOR D=9 TO 40:SET(21,D):NEXT D
1220 RESTORE
1230 FOR D=1 TO 61:READ A,B:SET(A,B):NEXT D
1240 DATA 10,30,10,31,11,29,10,32,10,33,11,34,12,28,12,
     35,13,27,13,36,14,26,14,36,15,25,15,31,15,32,15,37
     ,16,24,16,30,16,37,17,23,17,29,17,37,18,22,18,28,1
     8,37,19,21,19,27,19,37,20,20,20,26,20,37
1250 DATA 22,9,22,19,22,26,22,37,23,10,23,18,23,26,23,3
     6,23,37,24,11,24,17,24,27,24,35,24,36,25,12,25,16,
     25,27,25,28,25,34,25,35,26,13,26,14,26,15,26,28,26
     ,29,27,30,27,31,27,32,26,33,26,34
1260 RETURN
1300 'BASS CLEF
1310 RESTORE:FOR X=1TO61:READA,B:NEXTX:FOR X=1 TO 92:RE
     AD V,K:SET(V,K):NEXT X:RESTORE:RETURN
1320 DATA9,18,9,19,10,17,10,18,10,19,11,16,11,17,11,19,
     11,20,12,15,12,16,12,19,12,20,13,15,13,19,13,20,14
     ,14,15,13,15,14,15,38,15,39,16,13,16,37,16,38,17,1
     3,17,36,17,37,18,13,18,35,18,36
1330 DATA19,13,19,35,19,34,20,13,20,34,20,33,21,13,21,3
     3,21,32,22,14,22,32,22,31,23,14,23,15,23,31,23,30,
     24,15,24,16,24,30,24,29,25,16,25,17,25,28,25,29,26
     ,17,26,18,26,27,26,28,27,18,27,19,27,26,27,27
1340 DATA28,19,28,20,28,21,28,22,28,23,28,24,28,25,28,2
     6,29,20,29,21,29,22,29,23,29,24,29,25,30,21,30,22,
     30,23,30,24,32,15,32,16,32,17,32,21,32,22,32,23,33
     ,15,33,16,33,17,33,21,33,22,33,23
1400 'REVIEW TREBLE
1410 CLS:PRINT@0,"TREBLE CLEF"
1420 GOSUB 1110
1430 GOSUB 1210
1440 FOR N=1 TO 12:GOSUB 1000:
1450 IF N/2=INT(N/2) S=S+10
1460 IF S=934 OR S=166 PRINT@S,"--"+K$+"--"+" "+Q$; ELS
     E PRINT @S,K$+" "+Q$;
1470 NEXT N: PRINT@988," TO REVIEW BASS CLEF - HIT ENTE
     R ";:INPUT Z$
1480 CLS
1490 GOSUB 1110
1500 GOSUB 1310
1510 PRINT@0,"BASS CLEF"
1520 FOR N=13 TO 24:GOSUB 1000
1530 IF N/2=INT(N/2) S=S+10
1540 IF S=924 OR S=156 PRINT @S,"--"+K$+"--"+" "+Q$; EL
     SE PRINT @S,K$+" "+Q$;
1550 NEXT N:PRINT@988,"TO PROCEED WITH QUIZ - HIT ENTER
      ";:INPUT Z$
1560 GOTO 310
1600 'PAUSE
1610 FOR Y=1 TO 1000:NEXT:RETURN
1700 END
```

KRELL SOFTWARE

presents for the TRS-80*
PET, Apple II, and Apple II Plus

ELECTORAL COLLEGE 1980

The Tool for forecasting the outcome of the 1980 presidential Election. Will it be CARTER? REAGAN? ANDERSON? or will the election be forced into the House of Representatives? This program, developed by a professor of Political Science is built to be used in two ways:

1. During the political campaign prior to the election and,
2. On Election Night, as the partial returns roll in on network news.

Using the state by state data on previous elections that the program provides, simulated elections are run and the probability of outcomes calculated. $14.95

COLLEGE BOARDS

The best way to sharpen your skills for the College Board SAT Exams is to work on actual examinations. Each of these 4 programs confronts the user with a virtually limitless series of questions and answers. Each program is based on past SAT exams and presents material of the same level of difficulty and in the same form as used in the verbal and mathematical portions of the College Board Examinations. Scoring on each exam is provided in accordance with the formula used by College Boards.

COLLEGE BOARD – VOCABULARY	19.95
COLLEGE BOARD – WORD RELATIONSHIPS	19.95
COLLEGE BOARD – MATH PART A	19.95
COLLEGE BOARD – MATH PART B	19.95
COMPLETE SET	59.95

TIME TRAVELER

The best of the adventure games. Confronts player with complex decision situations and at times, the demand for real time action. Using the time machine, players must face a challenging series of environments that include: the Athens of Pericles, Imperial Rome, Nebuchadnezzar's Babylon, Ikhnaton's Egypt, Jerusalem at the time of the crucifixion, the Crusades, Machiavelli's Italy, the French Revolution, the American Revolution and the English Civil War. Deal with Hitler's Third Reich, the Vikings, etc. Involve yourself with historical military and government operations, markets, etc. in fascinating game situations. Each game is unique! $24.95

THE SWORD OF ZEDEK

Fight to overthrow Ra, the Master of Evil. In this incredible adventure game you must confront a host of creatures, natural and supernatural. To liberate the Kingdom, alliances must be forged and treasures sought. Treachery, deceit and witchcraft must be faced in your struggle as you encounter wolves, dwarves, elves, dragons, bears, owl, orcs, giant bats, trolls, etc. Each game is unique in this spectacular and complex world of fantasy. $24.95

REALTIME SUPER STAR BASEBALL

Performance is based on the interaction of actual batting and pitching data. Players select rosters and lineups and exercise strategic choices including base stealing, pinch hitting, bunting, intentional walks, hit and run plays, etc. Games include double plays, wild pitches, infield errors, hit batsmen and pick offs. $14.00.
ALSO AVAILABLE ALL TIME SUPER STAR BASEBALL $14.95.

PRIME TIME

Players compete as network executives. Each selects T.V. shows for competing time slots. Choose from a range of programs including sit-coms, dramas, soaps, westerns, sci-fi, news and documentary shows, etc. Up to three players compete for ratings and advertising revenue. Program simulates fan loyalties and industry events including FCC rulings and criticism from civic groups. Exciting and realistic. In game package #3.

HOSTAGE

Negotiate and/or stage military raids in this contest between the Authorities and the Terrorists.

Terrorists select their target; choosing to seize hostages at foreign embassies, the U.N. Building, Airliners, Hospitals, School Buses, or even Nuclear reactors.

As in real life, public opinion counts and shapes the players actions. Players have a dramatic and realistically wide range of tactical options. This game accurately reflects the intricacies of threat, promise and all facets of negotiations. In game package #1.

GAME PACKAGE #1
HOSTAGE, BLACK GOLD, PRIMARY FIGHT — 19.95

GAME PACKAGE #2
BANZAI, BANZAI II, SUPER BANZAI THE BLACK DEATH — 19.95

GAME PACKAGE #3
PRIME TIME, STAR CLIPPER, BULLS AND BEARS — 19.95

GAME PACKAGE #4
WORDSWORTH, HARDSCRABBLE, BIBLE QUODE SHAKESPEARE QUODE, BILL OF RIGHTS QUODE — 19.95

**All programs require 16K • TRS-80 Programs requires Level II Basic • Apple programs require Microsoft Basic*

* TRS-80 is a trademark of the Radio Shack Division of Tandy Corporation.

Send check or money order to **Krell Software** ✓375
21 Millbrook Drive, Stony Brook, NY 11790 (516) 751-5139

BY THE TIME YOU READ THIS AD,

dial (617) 373-1599 and order Insort-80, your TRS-80* microcomputer would have sorted 1,200 names and addresses, 10,000 numbers, or any other Random Disk File from your Basic program... *for only $49.95!*

"Let us sort things out for you!"

S&M SYSTEMS, INC.
P.O. Box 1225
2 Washington Street
Haverhill, Massachusetts 01830

Approximate Sort Time for Insort-80 is eight minutes.
TRS-80 is a registered trademark of the Tandy Corporation.

TSHORT +™ from WEB Associates

THE MOST POWERFUL BASIC SHORTHAND EVER!

TRS-80 Keyboard with decals installed

- 11 "SAVEABLE" KUSTOM keys. Ten 10-character and one 64-character
- 41 preprogrammed LV II and DOS statement keys.
- Complete decal set (see picture) included for both LV II and DOS systems.
- Includes DEBOUNCE and AUTO REPEAT.
- Hold "SHIFT or CLEAR" and press desired key — entire statement is typed on screen.
- Includes LOWER CASE DRIVER. "@" key substitutes for "SHIFT."
- Less than 1K bytes machine language, relocatable in hi or low memory.
- Features self-entering commands. I.E. CONT; GOTO10; KUSTOM.
- Includes comprehensive instruction manual.

TSHORT + (one cassette includes Level II & single drive DOS) .... **$ 19.95**

TSHORT + on formatted diskette (DOS only, requires 2 or more drives) .... **$ 24.95**

Send Check or Money Order to: Phone Orders
(714) 494-2869
Dealer Inquiries Invited
UPS/COD — Add $3.00

WEB ASSOCIATES ✓27
New Mail Order Address
P.O. Box 96QI
Corona Del Mar CA 92625
(Calif. Residents add 6% tax)

SEND FOR FULL LINE CATALOG

Deep in the heart of ROM.

BINAX KIBUFF

John T. Blair
122 Dumont Ave.
Norfork VA 23505

In an attempt to break Microsoft's BASIC for the TRS-80, several of our club members have spent many sleepless nights tearing through the listings of the Level II ROM's. What follows is some of the fruits of their labors and we hope that some of the newer machine language programmers can save some time by calling on the following information.

DELAY is used for time delays. Load the BC registers with a number between one and FFFF Hex (H). When DELAY is called, a return will occur approximately

(28•(N − 1) + 31)•563.67 Nanoseconds

later. This routine is located at 0060H.

```
LD     BC,XXXX  ;where XXX = delay time
CALL   0060H
```

CLS is used to clear the screen and home the cursor. It is located at 01C9H.

KIBUFF reads the keyboard into a buffer until a carriage return (CR) is entered, then it returns to the calling routine. The starting address of the buffer is contained in 40A7H, noted by programmers as (40A7H) and is initialized to 41E8H. This can be changed by the user if desired. This routine resides at 0361H.

```
CALL   0361H
```

BINAX is used to convert a binary number in the HL register to its ASCII equivalent. The ASCII value is in a buffer beginning at 4130H. The contents of 4130H, (4130H), is always equal to 20H.

The contents of 4131H to 4135H will have the decimal ASCII characters. (4136H) will always be 00H.

```
CALL   0FAFH
```

AXBIN converts from ASCII to binary. Load the HL register pair with the address of the first ASCII character to be converted. On return the DE register contains the binary equivalent. The maximum number which can be converted is 65529. This routine is located at 1E5AH.

```
LD     HL,XXXH   ;XXXX is address of
                 char. to be converted
CALL   IE5AH     ;call convert routine
LD     A,(DE)    ;Acc. contains the
                 ;binary value
```

OUTSTR will output a string of characters from memory to the CRT. Load the HL register with the address of the first character. Characters are outputted until either a OOH or 22H is found. OUTSTR is located at 28A7H. (Note: Since the addition of a disk I have found this routine may or may not work. It will work if the system is brought up in BASIC2, non disk based BASIC.)

VIDEO displays any character in the 'A' on the CRT. You must push the DE register onto the stack before calling this routine. It is located at 0033H.

```
PUSH   DE       ;save DE
CALL   0033H    ;display character
POP    DE       ;restore DE
```

PRNTR, the printer driver routine, located at 058DH, is CALLed from 003BH. This routine locates the printer at 37E8H, and checks its ready status by reading 37E8H for a 30H.

This routine is very poor because it is not useful with many printers other than Centronic's. (Radio Shack has assumed that a smart printer will be used.)

You can capture the routine and use your own by changing the address stored at 4026H and 4027H to the address of your printer driver. (Note: Be sure to PROTECT your printer driver program by answering the MEMORY SIZE question).

SCANKE scans the keyboard for an entry (key closure) and returns the ASCII value of the key in the 'A' register. SKANKE is located at 002BH, you must exchange registers before calling this routine.

```
EXX             ;save registers
CALL   002BH    ;scan the keyboard
EXX             ;restore registers
AND    7FH      ;mask out bit 8
```

Cursor control table is located from 0506H to 0540H.

1. Backspace and delete character—04CEH.
2. Delete last character—04CE. HL point to video memory of character.
3. Cursor on—04B8H
4. Cursor off—04BDH
5. Convert to expanded characters—04F6H
6. Backspace cursor—04DAH
6a. Advance cursor—04ECH
7. Down feed cursor—04E7H
8. Up feed cursor—04F1H
9. Cursor home—04C0H
10. Cursor to beginning of line—04A1H
11. Erase from cursor to end of line—0573H
12. Clear all after cursor—057CH

Relocated to RAM on power up:

1. Vectors for interrupts—06D2 to 06E6H
2. Keyboard device control block—06E7-06EE
3. Video DCB—06EF to 06F6H
4. Printer DCB—06F7 to 06FEH
5. 4080 to 191CH (uncertain)

Basic command table—1650 to 181FH

(Note: This is ASCII text, however, the first character in a word has the most significant bit (8) set to indicate new word.)

Jump table—1820H to 18C9H. These are two word address, i.e., AE 1D means that the routine is located at 1DAEH. This table is a one for correspondence for the BASIC command table.

Error messages—18C9 to 18F6H. The data here is ASCII code and two words each, i.e., 4E 46 is the ASCII for the NF code.

If any of you readers have uncovered any more of the secrets and would like to share them, then please send addresses of the routines and any other information you have.

One last note, Radio Shack's RS-232 card has several locations for its addressing. The control of the Baud rate, parity, number of bits to be sent, etc., are a combination of both hardware and software selection. Its data port is located at port EBH.

I wish to express my thanks to Bob Coble, Dwane Saylor and Geogre Higson for their assistance in locating and defining the routines.■

The intricacies of PRINT statements.

Variations On A Theme

George R. Bullitt
Route 82
East Haddam, CT 06423

Since becoming hooked on the TRS-80 in July 1978, it has been my singular objective to collect all the available reference material I could find on programming that wonderful machine. At best the collected works cover the more obvious routines, ignoring what I consider to be the fundamentals that can change a good program into a better one.

Printing a Line

One of those fundamentals is the printing of a line. Have you ever found in a book even one paragraph on the printing of a simple line?

Well, turn on your set, because here it is! Get yourself into BASIC, type and RUN the following:

```
10 CLEAR 100
20 PRINT STRING$(63,"-")
```

How about that!

Let's analyze the code. Line 10 clears string space.(Delete line 10 and see what happens.) Line 20 utilizes strings which print almost anything, as many times as you want. In this instance it printed 63 dashes (—) which is the maximum for the TRS-80 screen. Just remember that the number is placed left of the comma, and the character you want printed is on the right side, enclosed in quotes (except for control codes).

Type and RUN this mini program:

```
10 CLEAR 100
20 PRINT STRING$(21,"X");STRING$(21"
");STRING$(21,"*")
```

See what I mean about the STRING$ function?

In case you have forgotten, the semicolon between statements prints everything on the same line and eliminates the need to repeat the PRINT statement for each command. On the other hand, typing a colon between statements, followed by PRINT, prints each statement on a different line. If you are a skeptic try it by retyping line 20 to read:

```
20:PRINT STRING$(21,"X"):PRINT
STRING$(21," "):PRINT STRING$
(21,"*")
```

Okay, now that you know how to print a line, let's get more specific and print a line at the top of the screen then another one 1/2 inch below it. Type and RUN:

```
10 CLEAR 100
20 L$ = STRING$(63,"-")
30 PRINT L$
40 PRINT CHR$(138)
50 PRINT L$
```

In spite of those funny looking bar graph lines, it works! CHR$(138) is the ASCII code instructing the computer to print three blank lines. It also happens to be the graphics code to print that up and down bar on the screen. However, when that code is sent to a printer it prints only the three blank lines.

If you write a program strictly for the video display use the PRINT statement to print blank lines instead of CHR$(138). If you include a routine that also writes to a printer, then use it in that part of the program to save time and memory.

Notice in line 20 that STRING$(63,"-") is made equal to a smaller string character (L$) again saving time and memory.

It was stated earlier that the STRING$ function prints almost anything, so why not a control code such as CHR$(138)? Retype line 40 to read:

```
40 PRINT STRING$(1,CHR$(138))
```

Remember, since we are using a control code, the quotes are not used to the right of the comma. Run the program and you will find that it does the same job, printing three blank lines. Each additional increase in the number to the left of the comma will print one more line and add 5/32 of an inch between the printed lines.

Changing the number to two will print four lines or 5/8 of an inch and changing it to four will print six lines or approximately 1 inch between lines and so on.

More Sophistication

To expand the theme and make our program more sophisticated, type and run:

```
10 CLEAR 100
20 L$ = STRING$(63,"-")
30 INPUT "NO. OF LINES
TO PRINT";L
40 FOR X = 1 TO L
50 PRINT L$
60 NEXT X
```

Lines 40 through 60 set up the FOR-NEXT loop, ending it when the number of lines you entered have been printed. Entering 14 fills up the screen. Seven fills half the screen. This type of program is especially useful for making up ruled lines on a page with your printer. If

ΩMEGA
SALES
CO.

"WHOLESALE COMPUTER PRICES" DIRECT TO THE PUBLIC

12 Meeting St., Cumberland, R.I. 02864

PRODUCT SPECIAL OF THE MONTH!!


TRS-80
Model II - $3,500


Apple II
16K - $1049


INTERTEC SUPERBRAIN
32K RAM - $2449.00
64K RAM - $2649.00

Atari 800
$749.00

Products are NOW IN STOCK AT ΩMEGA Sales Co.

For TRS-80, Apple, CBM
(Interface Included)

Epson TX-80 - $745


NEC Spinwriter
5510-5530 - $2449


Soroc 120 - $699

CALL TOLL FREE FOR ΩMEGA'S PRICE!

ΩMEGA OFFERS THE BEST DELIVERY AND PRICE ON:
APPLE • ATARI • TRS-80 MODEL II • INTERTEC •
T.I. 810 • HEWLETT-PACKARD-85 • SOROC •
COMMODORE • NEC • QUME • CENTRONICS

ΩMEGA sells only factory fresh, top quality merchandise to our customers.
ΩMEGA will try to match any current advertised price with similar purchase conditions.
Before you buy anywhere else - be sure to call ΩMEGA Sales Co.
1-401-722-1027 or

ΩMEGA TOLL FREE

1-800-556-7586

ΩMEGA ships via UPS, truck, or air. COD's, VISA, Mastercharge accepted.

VISA master charge

you want to double space the lines change line 50 to read:

```
50 PRINT L$: PRINT
```

An easier way, however, is to increase the number in line 20 from 63 to 64. Increasing the number adds a dash (–), which pushes the invisible cursor around the corner to the head of the next line, completing the command. Then it moves to the line below, creating the double space, to wait for another command, which in this case is to print 64 dashes.

Now, how about printing a line at the top of the screen and then another one two inches below it with a message in between. Type and RUN:

```
10 CLEAR 100
20 L$ = STRING$(63,"-")
30 N$ = STRING$(4,CHR$
(138))
40 PRINT L$
50 PRINT N$
60 PRINT TAB
(22);"REPORT TO THE
BOARD
70 PRINT N$
80 PRINT L$
```

Not bad! But what if I want to vary the report heading? In that case, add the following:

```
35 INPUT "TYPE IN YOUR REPORT
HEADING";R$
```

Since the number of letters in the report heading will vary, a formula has to be built into the program so that the computer can figure the correct tab position, regardless of the number of letters.

The way we arrived at the tab position (22) in line 60 was to subtract the number of letters in our heading (19) from the total number of spaces (63) dividing the answer by two. Translating that into a formula the computer will understand is relatively easy. Simply stated the formula is: 63 minus the length (LEN) of the heading (R$) divided by two (/2).

In computerese it is written 63-LEN(R$)/2. Since the computer has to calculate the value of 63-LEN(R$), before it can divide the answer by two, it must be enclosed in brackets.

Now the formula looks like this: (63-LEN(R$)/2. It follows that the answer will be our final tab position. The formula itself will have to be enclosed in brackets. Retype line 60 and RUN:

```
60 PRINT TAB ((63-LEN(R$))/2);R$
```

For the finale, use the same formula to print a line across the screen with the report heading embedded in the middle. Type and RUN (clear the previous program by typing NEW and press ENTER):

```
10 CLEAR 100
20 INPUT "TYPE IN YOUR
REPORT HEADING";R$
30 L$ = STRING$((63-LEN
(R$))/2,"-")
40 PRINT L$;R$;L$
```

Simple but effective!

Note that the formula is placed to the left of the comma in line 30, giving the computer the same information as in line 60. However, instead of printing blank spaces to reach the tab position, it prints dashes (–).

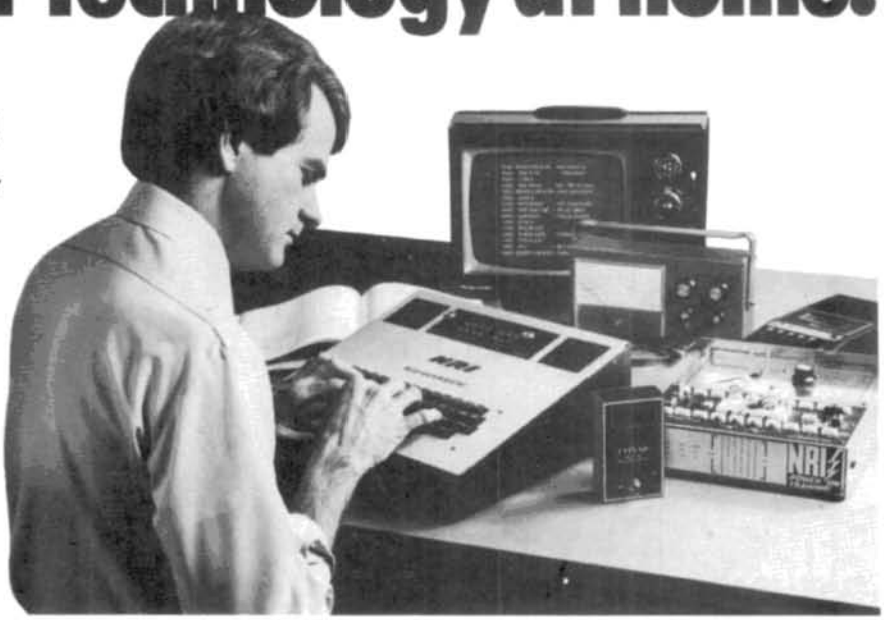
To leave a blank space on either side of the report heading change lines 30 and 40 adding line 35:

```
30 L$ = STRING$((63-LEN
(R$))/2)-1,"-")
35 B$ = STRING$(1," ")
40 PRINT L$;B$;R$;B$;L$
```

In order to leave room for a blank space, one dash is subtracted from the formula in line 30, and STRING$(1," ") is added to print that blank space on either side of the heading.

If you would like to do the exercises in this article on your printer change all PRINT statements to LPRINT. Also, you can change the number 63 to the number of characters (columns) your printer is capable of printing across the page. ■

Build your own microcomputer as you learn computer technology at home.



New from NRI! The First Interdisciplinary Home Study Course Ever Offered

As the microprocessor revolutionizes the computer world and microcomputers appear almost everywhere, NRI brings you a new, convenient, and effective way to keep up with this expanding technology. It's NRI's courses in Microcomputers and Microprocessors, created and designed exclusively for learning at home in your spare time.

Designed for the New Breed of Computer Technician

It's no longer enough to be just a programmer or technician. With microcomputers moving into the fabric of our lives as low-cost, easily available tools for business and home, both the programmer and technician must become total professionals. With practical knowledge of hardware, the programmer can design simpler, more effective programs. And with advanced programming skills, the technician can test and debug systems quickly and easily. The NRI course gives you simultaneous training in both skills...makes you one of this rare new breed.

Build Microcomputer, Test Instruments

NRI goes far beyond book learning to give you practical, "hands-on" experience. As you learn, you actually assemble NRI's designed-for-learning microcomputer. It performs like the finest of its kind, and features both assembly and basic language capabilities.

Every assembly step's a learning step. Using the NRI Discovery Lab® plus the NRI transistorized volt-ohm meter and CMOS digital frequency counter you also build, you perform meaningful experiments throughout your course...trace circuitry, interface components, introduce and correct problems, design your own programs, and more.

The Proven Way to Learn at Home

You don't have to worry with travel, classes, or time lost from work when you learn the NRI way. As they have for more than 60 years of teaching technical subjects, NRI brings the material to you. You study in your spare time, at your convenience, using "bite-size" lessons that program material into logical segments for easier assimilation. You perform experiments and build equipment using kits we supply. And your personal NRI instructor is always available for consultation should you have questions or problems. Over a million students have already shown the effectiveness of NRI training.

Choice of Courses

Several courses are available, depending upon your needs and background. NRI's Master Course in Microcomputers and Microprocessors starts with the fundamentals, explores basic electronics and digital theory, the total computer world, and the microcomputer. The Advanced Course, for students already versed in digital electronics, concentrates on software and the world of the microprocessor and microcomputer. In both courses, you build all instruments and your own computer.

Send for Free Catalog... No Salesman Will Call

Get the details on these exciting new courses in NRI's free, 100-page catalog. Shows all kits and equipment, lesson outlines, and full information, including facts on other electronics courses. Mail the coupon today and we'll rush your catalog. No salesman will ever call. Keep up with the latest technology as you learn on your own computer. If coupon has been removed, write to NRI Schools, Computer Department, 3939 Wisconsin Ave., Washington, D.C. 20016.

NRI Schools
McGraw-Hill Continuing Education Center
3939 Wisconsin Avenue
Washington, D.C. 20016

NO SALESMAN WILL CALL.
Please check for one free catalog only

All career courses approved under GI Bill.
☐ Check for details.

- ☐ Computer Electronics Including Microcomputers
- ☐ TV/Audio/Video Systems Servicing
- ☐ Complete Communications Electronics with CB • FCC Licenses • Aircraft, Mobile, Marine Electronics
- ☐ CB Specialists Course
- ☐ Amateur Radio • Basic and Advanced
- ☐ Digital Electronics • Electronic Technology • Basic Electronics
- ☐ Small Engine Repair
- ☐ Electrical Appliance Servicing
- ☐ Automotive Mechanics
- ☐ Auto Air Conditioning
- ☐ Air Conditioning, Refrigeration, & Heating Including Solar Technology

Name (Please Print) Age

Street

City/State/Zip

Accredited by the Accrediting Commission of the National Home Study Council

179-030

Reader Service—see page 226

A sort program with options plus.

Beyond Shell Metzner

Doug Walker
3485 Mock Orange Ct. S.
Salem, OR 97302

I needed a BASIC sort utility for my TRS-80, and not wanting to reinvent the wheel, I reviewed available literature in the hopes of finding a printed utility that met my business needs. I went looking—without success—for a utility that could perform user-specified major to minor field sorts in ascending or descending order on alphanumeric or numeric fields.

I did learn that the Shell-Metzner sort algorithm provides the fastest average BASIC sorting benchmarks.

Also, records can be rearranged by swapping or by using record pointers in a separate array or within a field of a multiple dimension array that is being sorted. In general, pointers allow faster BASIC sort speeds than record swapping, but more computer memory is needed to hold them and the expanded BASIC coding.

Both alphanumeric and numeric sorts can be readily accommodated, if a character string array holds the file to be sorted. The BASIC VAL function is used for numeric field sorts.

If the record swapping method is used, it is much faster to use the BASIC VARPTR function to swap character string field addresses than to exchange field values. This approach minimizes time consuming string space reallocations.

Develop Your Own Sort

Armed with these ideas, I developed a Level II, BASIC sort utility of my own.

Within a two-dimensional array, it allows you to specify an unlimited number of sort fields in major to minor sort sequence. You can also specify whether the sort on each field is to be ascending or descending, and whether numeric or alphanumeric.

The user specifies the sort parameters through INPUT and INKEY$ statements; but the program can be modified easily to assign the parameters for your business application program and then branch to the BASIC sort utility.

Program Listing 1 contains the BASIC program. Lines 140 through 250 are not part of the sort utility, but generate random character string data that is input for demonstration sorts.

Lines 260 through 480 prompt the user to input the sort parameters. Notice that the single dimension array A$( ) contains the data field names. Lines 500 through 660 contain the coding

```
ENTER AMOUNT OF STRING SPACE TO RESERVE? 2000
ENTER NUMBER OF RECORDS TO GENERATE? 5
BNBQMOQ         82          M           97055
NT              18          F           97052
DD              69          F           97054
TMOGGZS         51          M           97051
QTFMRZTX        61          F           97054
ENTER NAME SORT FIELD 1 ('NAME', 'AGE', 'SEX', OR 'ZIP CODE') OR
 ENTER 'STOP'? SEX
IS THE FIELD NUMERIC (Y/N)?N
ASCENDING OR DESCENDING SORT (A/D)?A
ENTER NAME SORT FIELD 2 ('NAME', 'AGE', 'SEX', OR 'ZIP CODE') OR
 ENTER 'STOP'? AGE
IS THE FIELD NUMERIC (Y/N)?Y
ASCENDING OR DESCENDING SORT (A/D)?D
ENTER NAME SORT FIELD 3 ('NAME', 'AGE', 'SEX', OR 'ZIP CODE') OR
 ENTER 'STOP'? STOP
DD              69          F           97054
QTFMRZTX        61          F           97054
NT              18          F           97052
BNBQMOQ         82          M           97055
TMOGGZS         51          M           97051
```

Example 1. A Sample Run

BRING YOUR TRS-80 KEYBOARD TO LIFE!

More than a year ago we used this headline to introduce AUTOK and QEDIT, the keyboard autorepeat and quick edit utilities for Level II BASIC. They've been very popular. But we couldn't resist working them over anyway, in response to customer suggestions. The result? **KEYEDIT**, a vastly improved AUTOK, QEDIT, and a few things more.

With **KEYEDIT** and your Level II or Disk BASIC system, you get:

- Debouncing. No need to use Radio Shack's KBFIX!
- Autorepeat on every key. Just hold a key down, and after a half second delay, the character repeats about eight per second.
- Single-keystroke keyword entry. Hold down SHIFT, hit a letter key, and an entire BASIC keyword is spelled out at once. Plus, you can assign any keyword to any key!
- Keyboard macro facility. Any frequently-typed pattern can be defined and later invoked in a single keystroke. You just fill in the blanks. Takes the drudgery out of repetitive keying.
- Screen oriented editing. **KEYEDIT**'s cursor moves anywhere in a displayed program listing for instant insertions and deletions. Plus, whole statements can be copied to other parts of the program or combined to form longer ones, without retyping the text. Makes BASIC's EDIT function obsolete!
- Easy loading. **KEYEDIT** loads from cassette using CLOAD (even though it's written in machine language), and may be saved on disk. Features can be deleted selectively just by deleting lines. Once RUN, **KEYEDIT** protects itself in memory and links into BASIC, where it unobtrusively awaits your command.
- Thorough documentation. Each feature is explained in detail along with instructions for user modifications.

KEYEDIT will save you hours of effort in BASIC program development. So why waste another minute? Bring your keyboard to life today with **KEYEDIT**!

KEYEDIT Level II cassette and instructions $19

DISCOVERY BAY SOFTWARE CO.
P.O. Box 464 Port Townsend, WA 98368

Bankcards welcome. Dealer inquiries invited.
Write us directly for more information.

CPT EXPANSION INTERFACE's

FOR TRS80's™ | FROM COMPUTEX

FEATURES

- PROVEN LNW SYSTEM EXPANSION BOARD
- RS232/20mA SERIAL INTERFACE
- MEMORY EXPANSION TO 32K BYTES
- CUSTOM FINISHED WOOD CABINET
- REMOVEABLE FRONT PANEL ALLOWS EASY DISK DRIVE ADDITIONS
- BUILT IN MUFFIN FAN AND DISK DRIVE POWER SUPPLIES
- 15 DIFFERENT CONFIGURATIONS FROM $249 KIT to $1300 for 2-DOUBLE DRIVE 32K ASSEMBLED SYSTEM

✓392

COMPUTEX (713) 332-4359
17710 Heritage Ct Webster Tx 77598

SUPERIOR SOFTWARE PACKAGES FOR THE DISK BASED **TRS-80***

SMARTTERM •$79.95

UNQUESTIONABLY THE BEST SMART TERMINAL PACKAGE FOR THE TRS-80

- True Break Key
- Auto Repeat (Typomatic) keys
- Programmable 'soft' keys
- Forward/Reverse Scrolling Multipage Display
- Transmit from Disk File, Screen or Buffer
- Receive to Disk File, Buffer or printer
- Multi Protocol Capability

SPOOL-80 • $39.95

A TRUE DISK-TO-PRINT DESPOOLER FOR THE TRS-80

- Print Disk Files While Running Other Programs
- Prints Compressed Basic Files
- Includes RS-232 Driver for Serial Printers

CALL US FOR YOUR CUSTOM SOFTWARE REQUIREMENTS

MICRON, INC.
10045 Waterford Drive
Ellicott City, MD 21043
(301) 461-2721

Model II Versions Available Soon

*TRS-80 is a Trademark of Tandy Corp.

— TRS-80 MODEL II USERS —

Preserve — Protect — Display

your equipment with

CRYSTAL CLEAR PLASTIC COVERS

- Keyboard and CRT ..... $24.95 ea
- Line Printer III ..... $14.95 ea

plus $1.50 postage each
(Indiana residents add 4% sales tax)

Introductory Offer: Buy both covers & save

Combination price $34.95 including postage

MODEL I covers also available.
DEALER INQUIRIES INVITED.

Crown Plastic Co. ✓119
3746 N. College 317-925-5566
Indianapolis, IN 46225

VISA

"HISPED"

TAPE OPERATION

2K Baud *PLUS* for the TRS-80™

- *Save, verify and load* **programs** up to 4 times faster than normal
- Save, verify and load array **data** up to 30 times faster than PRINT#
- User variable hardcopy formatting (3 output routines work with most printers)
- "HISPED" is a machine language program (not a hardware add-on)
- 2 copies plus a **free basic test program** supplied on high quality cassette—for level 2, 16K, 32K or 48K.

Write for full specifications or send $24.95 (ck or mo) + $1.00 P/H (Calif. residents add 6% sales tax)

24 Hour Ordering Line (415) 366-5340

VISA **PALOMAR SOFTWARE** master charge

170 S. Palomar Dr.
Redwood City, Ca. 94062

✓228

"TRS-80 is a registered trademark of TANDY CORP."

Turn your Micro into an electronic and security controller with our new...

TRS-80 INTERFACE

To Order, Send $59.95 plus $2.05 for shipping and handling
Illinois Residents add 6% tax)

TRS-80 Software (Specify Tape or Disk) $10 with purchase of Micro Commander

$59⁹⁵

90 Day Warranty
Assembled & Tested;

COMPUTER CONTROLLED - REMOTE CONTROL
Now an inexpensive and direct carrier current interface between the TSR-80 and the BSR X-10 remote control modules. The **MICRO COMMANDER**. X-10 modules and your computer can control your lights, appliances, motors, TV, stereo, heaters, alarms, fans, pumps, etc.

COMPUTER CONTROLLED - SECURITY
Add a new dimension to your security system. Place your home under control of your computer real time clock while you are on vacation. Add an input port to your computer and intelligence looms. With switches on doors or windows your computer can welcome guests or frighten intruders.

EASY TO USE - NO WIRES TO RUN
Total control of all X-10 modules. Utilize all 256 house and unit code combinations. Direct interface to AC power line. No command console to purchase. No sonic link. Plugs directly into TRS-80 Model 1 (Level II or Disk) cassette jack or any 1 bit input and 1 bit output TTL port.

MANUAL INCLUDES ALL SOFTWARE LISTINGS
The 14K basic 4 in 1 program does the following:
Loads machine language driver.
Simulates command console.
Provides sequential operation.
Utilizes real time clock for control.
Complete documentation includes 8080 source listing.

INTERFACE TECHNOLOGY ✓319
P.O.Box 383, Des Plaines, Il 60017
Phone (312) 297-2265

for the sorts.

The Shell-Metzner sort algorithm[1] has been used with a subroutine in lines 930 through 1010. Using the VARPTR, this subroutine compares sort field values (that switch record address pointers) in line 1030 through 1090.[2] Notice in the sort field value comparison that VAL is used to perform numeric sorts. The relational operators ">" and "<" are used for ascending and descending sorts.

Lines 680 through 690 check to see if there is another sort field. If not, the sort is completed and lines 850 through 870 display the sorted records.

If there is another sort field, lines 710 through 830 identify ranges of records in the prior sort which are equal to those in the second field. The beginning and ending record numbers of the previous sort field ranges, which equal the ranges in the second field, are input to the Shell-Metzner algorithm. These records are then re-sorted to the current field specifications.

Line 560 ignores record comparisons that are outside the current range of interest. This process is repeated until all records are processed.

A Sample Run

Example 1 is a sample run of five records generated by random field values. Next, a sort is specified in the order shown in Table 1.

Finally, the sorted records are displayed. Note that all "F" sex field records are ordered before all "M" sex field records (ascending sort sequence on major sort field). Within "F" and "M" ages are ordered from oldest to youngest (descending sort order on the minor sort field).

This BASIC sort has proved reliable and a real time-saver when writing business programs. Hopefully, you will be able to put it to good use with your system. ■

Sort Sequence	Field Name	Sort Order	Data Type
Major	Sex	Ascending	Alphanumeric
Minor	Age	Descending	Numeric

Program Listing 1

1. Thomas E. Doyle, *Kilobaud Microcomputing*, "5 Minutes Or 5 Hours—Sorting Routines Compared," May 1978, pages 100-102.
2. T. R. Dettman, *80-US Journal*, "Super Sorting," November/December 1979, pages 28 and 62.

```
10 REM  *  USER DEFINED MULTIPLE FIELD SORTS  *
20 REM  *  *  *  ASCENDING OR DESCENDING  *
30 REM  *  *  *  ALPHANUMERIC OR NUMERIC  *
40 REM  *  *  *  MAJOR TO MINOR  *
50 REM              DOUG WALKER
60 REM              3485 MOCK ORANGE COURT SOUTH
70 REM              SALEM, OREGON   97303
80 REM              (503) 393-2685
90 REM  *  HOUSEKEEPING  *
100 CLS:DEFINT A-Z
110 INPUT"ENTER AMOUNT OF STRING SPACE TO RESERVE";D
120 CLEAR D
130 REM  *  GENERATE RANDOM SORT RECORDS  *
140 INPUT"ENTER NUMBER OF RECORDS TO GENERATE";A
150 DIM C$(A,4),A$(4):A$(1)="NAME":A$(2)="AGE":A$(3)="S
    EX":A$(4)="ZIP CODE"
160 FOR I=1 TO A
170 B=RND(8)
180 FOR C=1 TO B
190 C$(I,1)=C$(I,1)+CHR$(RND(26)+64)
200 NEXT C
210 E=RND(85):C$(I,2)=STR$(E)
220 IF RND(2)=1 THEN C$(I,3)="M" ELSE C$(I,3)="F"
230 E=97050 + RND(5):C$(I,4)=STR$(E)
240 PRINT C$(I,1),C$(I,2),C$(I,3),C$(I,4)
250 NEXT I
260 REM  *  USER SPECIFIES SORT PARAMETERS  *
270 FOR C=1 TO 4:C1(C)=-1:C2(C)=0:D(C)=0:NEXT C
280 R=1:J=A:K=1
290 PRINT"ENTER NAME SORT FIELD";R;:INPUT"('NAME', 'AGE
    ', 'SEX', OR 'ZIP CODE') OR ENTER 'STOP'";S$
300 IF S$="STOP" AND R=1 GOTO290
310 IF S$="STOP" GOTO480
320 FOR C=1 TO 4
330 IF S$=A$(C) GOTO360
340 NEXT C
350 GOTO290
360 C1(R)=C
370 PRINT"IS THE FIELD NUMERIC (Y/N)?";
380 GOSUB1100
390 S1$=CH$
400 IF S1$="Y" THEN C2(R)=1:GOTO420
410 C2(R)=2
420 PRINT"ASCENDING OR DESCENDING SORT (A/D)?";
430 K$=INKEY$:IF K$="" GOTO430 ELSE PRINT K$
440 IF K$="A" THEN D(R)=1:GOTO460
450 IF K$="D" THEN D(R)=2 ELSE GOTO420
460 IF R=4 GOTO480
470 R=R+1:GOTO290
480 Z6=1
490 REM  *  SORTS USING SHELL METZNER ALGORITHM "*
500 Z3=J
510 Z3=INT(Z3/2)
520 IF Z3=0 THEN 650
530 Z4=K:Z5=J-Z3
540 Z7=Z4
550 Z8=Z7+Z3
560 IF Z6>1 AND (Z7<K OR Z7>J OR Z8<K OR Z8>J) THEN 620
570 GOSUB920
580 IF F1=2 GOTO620 ELSE GOSUB1020
590 Z7=Z7-Z3
600 IF Z7<1 THEN620
610 GOTO550
620 Z4=Z4+1
630 IF Z4>Z5 THEN 510
640 GOTO540
650 IF Z6=1 THEN 680
660 IF Z6>1 AND M=A THEN 680 ELSE 720
670 REM  *  SEE IF THERE IS ANOTHER FIELD TO SORT  *
680 Z6=Z6+1
690 IF C1(Z6)=-1 THEN 840
700 REM  *  SORT SEGMENT COMPUTATION ON INTERMEDIATE TH
    RU MINOR FIELD SORTS  *
710 M=1
720 Z7=M:Z8=M+1:J=1:L=0
730 IF C2(Z6-1)=1 THEN 760
740 IF C$(Z7,C1(Z6-1))=C$(Z8,C1(Z6-1)) THEN J=J+1 ELSE
    L=1
750 GOTO770
760 IF VAL(C$(Z7,C1(Z6-1)))=VAL(C$(Z8,C1(Z6-1))) THEN J
    =J+1 ELSE L=1
770 IF L=0 THEN 800
780 IF L=1 AND J>1 THEN K=M:J=Z7:M=Z8:GOTO500
790 L=0:M=Z8
800 Z7=Z7+1:Z8=Z8+1
810 IF Z7<A THEN 730
820 IF L=0 THEN K=M:M=Z7:J=Z7:GOTO500
830 GOTO680
840 REM  *  DISPLAY SORTED RECORDS  *
850 FOR I=1 TO A
860 PRINT C$(I,1),C$(I,2),C$(I,3),C$(I,4)
870 NEXT I
880 REM  *  CONTINUATION CHECK  *
890 PRINT"DO YOU WANT TO SORT THESE RECORDS AGAIN (Y/N)
    ?";
900 GOSUB1100
910 IF CH$="Y" THEN 270 ELSE END
920 REM  *  RECORD ADDRESS POINTER SWITCH CHECK  *
930 IF D(Z6)=2 GOTO970
940 IF C2(Z6)=1 GOTO960
950 IF C$(Z7,C1(Z6))<C$(Z8,C1(Z6)) THEN 1000 ELSE 1010
960 IF VAL(C$(Z7,C1(Z6)))<VAL(C$(Z8,C1(Z6))) THEN 1000
    ELSE 1010
970 IF C2(Z6)=1 THEN 990
980 IF C$(Z7,C1(Z6))>C$(Z8,C1(Z6)) THEN 1000 ELSE 1010
990 IF VAL(C$(Z7,C1(Z6)))>VAL(C$(Z8,C1(Z6))) THEN 1000
    ELSE 1010
1000 F1=2:RETURN
1010 F1=1:RETURN
1020 REM  *  RECORD ADDRESS POINTER SWITCH  *
1030 FOR C=1 TO 4
1040 I1=PEEK(VARPTR(C$(Z7,C))):I2=PEEK(VARPTR(C$(Z7,C))
     +1):I3=PEEK(VARPTR(C$(Z7,C))+2)
1050 J1=PEEK(VARPTR(C$(Z8,C))):J2=PEEK(VARPTR(C$(Z8,C))
     +1):J3=PEEK(VARPTR(C$(Z8,C))+2)
1060 POKE(VARPTR(C$(Z7,C))),J1:POKE(VARPTR(C$(Z7,C))+1)
     ,J2:POKE(VARPTR(C$(Z7,C))+2),J3
1070 POKE(VARPTR(C$(Z8,C))),I1:POKE(VARPTR(C$(Z8,C))+1)
     ,I2:POKE(VARPTR(C$(Z8,C))+2),I3
1080 NEXT C
1090 RETURN
1100 REM  *  Y/N INPUT CHECK  *
1110 CH$=INKEY$:IF CH$="" THEN 1110
1120 IF CH$="N" OR CH$="Y" THEN PRINT CH$:RETURN ELSE 1
     110
```

Table 1

T U B E

The Ultimate Buffered Editor for TRS-80 Disk Systems

TECO like, 21 edit commands, direct cursor control, block move, multiple input lines, file size limited only by available disk space. This is A FULL SCREEN EDITOR UTILIZING CURSOR CONTROL.

- **On TRS-80 Disk with Manual — $40.00**
  Also available for TRS-80

SBASIC Structured Basic Pre-Processor — $50.00

- **BASOPT Basic program optimization program — $20.00**

ULTIMATE COMPUTER SYSTEMS

313 Meadow Lane
Hastings, Michigan 49058
(616) 945-5334
(Dealer inquiries invited)

84

VISA & MASTERCHARGE ACCEPTED

**TRS-80 is a trademark of Tandy Corp.*

NEW INTERACTIVE DISASSEMBLER

INTER-DIS80

A Z-80 disassembler designed specifically for the TRS-80* Model I. With it you can obtain the original mnemonics from Z-80 machine code. You can also follow the logic of machine language programs like the routines in your ROM, and unravel the secrets of the BASIC interpreter.

Features:

—Enables you to follow jumps and subroutine calls.
—Continuous or one line at a time disassembly.
—Outputs to screen or to printer, at user's option.
—You can insert your own comments during disassembly.
—Completely relocatable. Relocator program included.

INTER—DIS80 requires a 16K machine with Level II BASIC. Disassembler and relocator supplied on cassette only.

Price: **$14.95**. Cheques or money orders accepted. (Quebec residents add 8% sales tax.)

OKTOLOGIK 426

874 NANTEL, LONGUEUIL
QUEBEC, CANADA J4L-1Z9

Dealer inquiries welcome.
***TRS-80 is a trademark of Tandy Corporation.**

INTELLIGENT TERMINAL

Use your TRS-80* as a dial-up terminal on any standard timeshare system. Includes control keys. Print command lists data on printer. Store command writes data to disk file. Write command reads disk file and outputs to RS232 interface $49.95

ADDITIONAL SOFTWARE AVAILABLE:

ACCOUNTS RECEIVABLE $495
Maximum 9000 customers with up to 3000 transactions per customer per billing period. Includes customer status reports, past due billings, mail list, etc. Reports may be run at any time as often as desired. High-quality, professional software.

GENERAL LEDGER $495
For medium sized business. Designed and proven impossible to unbalance books. IRS auditable. Up to 9 departments. High-quality, professional software.

PAYROLL $495
For up to 120 people per year. Some high-quality, professional software.

MAIL LIST $9.95
Basic, no-frills program. Easy to use. Lists all (or range of) addresses. Prints 4 lines by 30 chars on standard 3½" x 15/16" label.

UNILOGIC 423
P.O. Box 160
PARIS, KY 40361
(606) 987-2678
(606) 987-4310

(32K TRS-80* with disk drive required.)
*TRS-80 is a trade mark of Tandy Corp.

TRS-80™ SOFTWARE

MACHINE LANGUAGE SOFTWARE

MONITOR #3 .... $39.95
Disassembler; memory displays; memory move, search verify, and modify, read and write object tapes; hexadecimal arithmetic; object code relocator; unload programs for disk, symbolic output tapes; 41-page instruction manual.

MONITOR #4 .... $49.95
Same as Monitor #3 but adds: save and read disk files; direct input and output of disk sectors; send, receive, or talk to another computer via RS-232-C interface; symbolic disassembly on disk.

SMART TERMINAL $49.95
Enables your TRS-80 to be used as a remote terminal to a timesharing system. Supports lowercase and full-range of control keys. *Automatic transmission* between memory and host computer. Much more.

FASTSORT $9.95
Machine-language sorting program for use by Basic programs. Many times faster than other methods!

GAME OF LIFE $5.95
John Conway's game of "life" shows patterns evolving and changing swiftly before your eyes. A dazzling demonstration program!

BASIC SOFTWARE

MAILING LIST $69.95
Maintains mailing list files of over 1000 names per diskette. Add, delete, change, find name, machine language sort, print file.

SMALL BUSINESS ACCOUNTING $49.95
Based on Dome Bookkeeping Journal #612, keeps track of income, expenditures, and payroll for a small business of up to 16 employees. Daily, monthly, year-to-date summaries.

HOME BUDGET $49.95
Checkbook maintenance combined with records of income and monthly bills. Monthly and year-to-date summaries showing tax deductions.

DATABASE MANAGEMENT $29.95
Defines files of any description and maintain on cassette or disk. Add, change, delete, find, sort, justify, print, line print, total fields, write.

HOWE SOFTWARE 103
14 Lexington Road
New City, New York 10956

(*) TRS-80 is a registered trademark of Tandy corp.

COTTAGE SOFTWARE

FOR TRS-80™ Micro Computers

PACKER: Automatically edits all or part of your Basic program to ease editing, run faster, or save memory. Has 5 sections: UNPACK – unpacks multiple statement lines into single statements maintaining program logic; inserts spaces and renumbers lines for easier editing. SHORT — shortens your program by editing out all REM statements, unnecessary words and spaces. PACK – executes UNPACK and SHORT, then packs lines into multiple statement lines; maintains program logic. RENUM – renumbers program lines including all GOTO's, etc. You specify increment. MOVE – moves any line or block of lines to any new location in the program and renumbers lines. Written in machine language; supplied on tape in 3 versions for 16K, 32K, & 48K. For Level II or Disk Basic **$29.95**

DISASSEMBLER: Read, write, and copy system tapes. Display and modify memory contents. Disassemble ROM, DOS, and system tapes into Z-80 mnemonics. Search for strings in memory. Much more!! Includes 32 pages of documentation and information.
For 16K Level II **$19.95**

SYSTEM TAPE DUPLICATOR: Copy your system format tapes. Includes verify routine.
For any Level II **$14.95**

CHESDISK: Transfers your copy of Microchess to disk for quick and easy access.
For any Level II Disk system **$8.95**

CASSETTE LABEL MAKER: A mini-word processor to print cassette labels on a line printer. Includes manual and 50 peal-and-stick labels on tractor feed paper.
For 16K Level II and printer **$15.95**

INSTRUCTION MANUALS for any Cottage Software original programs available for 20% of program list price. Refundable when program purchased.

TRS-80* repairs and modifications. Call or write for info. MANY MORE items available. Call or write for catalog. DEALER inquiries invited.

Kansas residents add 3% sales tax. Foreign orders in US Currency only.

Call our 24-hour phone: 316-683-4811 or write

"TRS-80 is a registered trademark of TANDY CORP."

COTTAGE SOFTWARE
614 N. Harding 233
Wichita, KS 67208

SO YOU WANT TO USE YOUR TRS-80 TO:

Relearn French. Help your children with their reading. Teach your entire family to program. But Interaction Is So Awkward In BASIC.

BPILOT IS THE LANGUAGE FOR YOU

BPILOT is a version of PILOT written especially for the TRS-80 with either Level II or Disk BASIC. PILOT was developed by teachers for Computer Aided Instruction. PILOT allows you to concentrate on the instructional goals, not the computer.

BPILOT HAS 3 VERY IMPORTANT FEATURES:

1. BPILOT is a complete PILOT. It includes the eXecute Indirect command, a complete Match instruction, and long labels.
2. You create BPILOT programs using the TRS-80 commands you are familiar with (AUTO, EDIT, LIST, TRON . . .). There are no new (and confusing) system commands.
3. BPILOT allows you to use both PILOT and BASIC instructions in your programs. Besides increasing the power of PILOT, this makes it easy to learn PILOT if you know BASIC, and easy to teach your children BASIC once they know PILOT.

BPILOT is a concise assembly language program making extensive use of the Level II ROMs. On a 4K system, the Level II version of BPILOT leaves 2K for programs. On a 16K system, the Disk version leaves 4K for programs. At $24.95 for either version, including a reference manual and 3 demonstration programs, BPILOT is a true bargain. For more information or to order, contact: 386

Computer Aided and Managed Instruction
P.O. Box 2030
Goleta, CA 93018

Access forbidden memory locations in your Model II.

Deflower Your Debug

Donnal C. Walter, M.D., Ph.D.
Departments of
Pediatrics & Pharmacology
Univ. of Kansas Medical Center
Rainbow Boulevard
Kansas City, KS 66103

Unlike ROM-based operating systems, the TRS-80 Model II operating system (TRSDOS version 1.2) is loaded from disk into internal memory on power up. Since the supervisor program and input/output drivers always reside in the lower 7K of RAM, this software should be easily accessible for inspection or temporary modification.

The TRSDOS Debug Monitor, however, is programmed to conceal memory locations below hexadecimal 2800. It also protects RAM above hexadecimal F300, its own starting address. The Debug Monitor displays and modifies the user area between hexadecimal 2800 and F2FF only.

In order to gain access to the entire RAM, one must either write an independent debug program or disable the memory protection system of the TRSDOS Debug Monitor. The latter may be accomplished with the procedure described in this communication.

Memory Protection System

Five debug commands (ra(M), (B)rk, (J)ump, (L)oad and (F)ind share an address-entry subroutine from hexadecimal F9C0 to FA19 which contains a protection segment (Table 1). The simplest way to disable this segment is to load 00 (NOP) into hexadecimal addresses FA01 to FA06 and 80FF into FA09-FA0A.

Address	Object Code	Instruction	Comments
F9FF	AF	XOR A	CLEAR ACCUMULATOR
FAØØ	21 FF 27	LD HL,27FF	LOWER LIMIT OF USER AREA
FAØ3	ED 52	SBC HL,DE	DE CONTAINS THE SELECTED ADDR.
FAØ5	3Ø CD	JR NC,disp	IF DE< HL DO NOT CONTINUE
FAØ7	AF	XOR A	CLEAR ACCUMULATOR
FAØ8	21 FF F2	LD HL,F2FF	UPPER LIMIT OF USER AREA
FAØB	ED 52	SBC HL,DE	DE STILL CONTAINS SELECTED ADDR.
FAØD	38 C5	JR C,disp	IF DE>HL DO NOT CONTINUE

Table 1

Address	Object Code	Instruction	Comments
28ØØ	21 Ø1 28	LD HL,28Ø1	
28Ø3	E5	PUSH HL	LOAD STACK FOR RETURN
28Ø4	3E ØØ	LD A,Ø	CLEAR ACCUMULATOR
28Ø6	21 Ø1 FA	LD HL,FAØ1	ADDRESS OF PROTECT IN DEBUG
28Ø9	77	LD (HL),A	CLEAR MEMORY 'FAØ1'
28ØA	23	INC HL	
28ØB	77	LD (HL),A	CLEAR MEMORY 'FAØ2'
28ØC	23	INC HL	
28ØD	77	LD (HL),A	CLEAR MEMORY 'FAØ3'
28ØE	23	INC HL	
28ØF	77	LD (HL),A	CLEAR MEMORY 'FAØ4'
281Ø	23	INC HL	
2811	77	LD (HL),A	CLEAR MEMORY 'FAØ5'
2812	23	INC HL	
2813	77	LD (HL),A	CLEAR MEMORY 'FAØ6'
2814	23	INC HL	
2815	23	INC HL	
2816	23	INC HL	
2817	36 8Ø	LD (HL),8Ø	PUT "8Ø" IN 'FAØ9'
2819	23	INC HL	
281A	36 FF	LD (HL),FF	PUT "FF" IN "FAØA'
281C	21 A5 F6	LD HL,F6A5	STEPMODE PROTECTION ADDR.
281F	36 FF	LD (HL),FF	PUT "FF" IN 'F6A5'
2821	C3 77F3	JP F377	ENTER DEBUG MONITOR

Table 2

The step-mode loader (ra(M)) has a protection segment at hexadecimal F6A3 to F6AC which may be disabled by loading FF into address F6A5.

There are two protection segments which should not be disabled. These are at F4A0 for (L)oad (TOP) and F930 - F93D for (U)pload.

The program in Table 2 makes the proper changes to disable the protection segments. It is written in linear fashion for simplicity, but a loop may be substituted for the repetitive sequence. Implement it as follows:

1. In TRSDOS READY mode, type DEBUG ON and press ENTER.

2. TRSDOS READY will return. Then type DEBUG and press ENTER.

3. RAM from 2800 will be displayed. To enter ra(M) command, press M.

4. The prompt A = will appear. To modify RAM, press the F1 key.

5. Now enter the object code program from Table 2.

6. To effect this change in memory, press the F2 key.

7. To execute the program press C.

The monitor will then access

any area of RAM desired. The modified debug program may be saved on disk with the following steps.

8. To return to TRSDOS press S.

9. In TRSDOS READY mode type DUMP BUGGY {START-F300, END = FF80, TRA = 3000} and press ENTER.

Now the command BUGGY may be used in place of DEBUG after DEBUG ON. The command DEBUG ON sets the proper calling parameters in the supervisor area, and the command BUGGY loads the modified program in high RAM. BUGGY will not run if DEBUG is OFF.

With BUGGY the entire RAM is available for inspection. The user who wants to study the architecture of TRDOS should first become familiar with the Debug Monitor. The debug program is a good introduction to supervisor calls (SVCs) and interaction with the operating system. Important addresses are listed in Table 3.

TRSDOS consists of a set of supervisor functions which reside in RAM and an overlay system which loads auxiliary code into a specified area as needed. Many of the supervisor functions can be called from user programs using RST 8 (SVCs).

The SVC calling segment is located between hexadecimal 0145 and 018A. Because the Debug Monitor uses an SVC to set breakpoints, the monitor command (B)rk cannot be executed in the SVC calling routine itself. The address table for the SVCs starts at hexadecimal 01A2 (Table 4). The stack pointer starts at 1F00, and the overlay area is above 2000.

Conclusion

Model II TRSDOS (version 1.2) is a versatile and powerful disk operating system which appears to be relatively free of errors. Since the system is user-oriented, most users (such as businesses and schools) will not require direct access to the operating system itself.

Computer hobbyists and systems' programmers, however, will benefit from easy access to the TRSDOS machine code (especially the supervisor functions). Until independent debuggers and disassemblers become available for the Model II, the TRSDOS Debug Monitor can be used.■

START ADDR	FUNCTION or OPERATION	START ADDR	FUNCTION or OPERATION
F3ØØ	Initialization	F467	(L)oad Command Routine
F377	Program Entry Semgnet	F4B7	(P)rint Command Routine
F3B6	Main Command Mode Routine	F4D2	(E)mpty Command Routine
F422	(D)ecimal Command Routine	F4EE	(F)ind Command Routine
F429	he(x) Command Routine	F562	(B)rk Command Routine
F43Ø	(J)ump Command Routine	F5Df	(C)ontinue Command Routine
F43F	(O)ut Command Routine	F7C7	ra(M) Command Routine
F445	(H)elp Command Routine	F8A5	(R)egister Command Routine
F457	(S)ystem Command Routine	F8A5	(U)pload Command Routine

Table 3

SVC NAME	START ADDR	SVC NAME	START ADDR	SVC NAME	START ADDR	SVC NAME	START ADDR
Ø INITIO	Ø2A2	11 VDREAD	Ø84B	25 TIMER	Ø342	41 KILL	13EF
1 KBINIT	Ø37A	12 VIDKEY	Ø94D	26 CURSOR	Ø6ØD	42 CLOSE	13F5
2 SETUSR	Ø518	15 DISKID	ØEA3	27 SCROLL	Ø9AA	43 WRITNX	18Ø9
3 SETBRK	Ø4E6	17 PRINIT	ØEEA	33 LOCATE	1B96	44 DIRWR	182C
4 KBCHAR	Ø39A	18 PRCHAR	ØF44	34 READNX	172F	45 DATE	1CØ9
5 KBLINE	Ø3DB	19 PRLINE	ØF25	35 DIRRD	1746	48 PARSER	12Ø1
6 DELAY	ØBB7	2Ø RANDOM	Ø9E6	36 JP2DOS	1237	49 STSCAN	1CC6
7 VDINIT	Ø56Ø	21 BINDEC	ØA35	37 DOSCMD	1388	52 ERRMSG	139D
8 VDCHAR	Ø63D	22 STCMP	Ø9D3	38 RETCMD	138D	55 RS232C	14Ø8
9 VDLINE	Ø92C	23 MPYDIV	ØABD	39 ERROR	13A6		
1Ø VDGRAF	Ø799	24 BINHEX	ØBØ2	4Ø OPEN	13E9		

Table 4

New Releases for the TRS-80*

Oracle-80

Oracle-80 will provide you with business analysis and forecasting capabilities previously available only in large computer and time-sharing systems. Oracle-80 is a flexible, professional time series analysis and forecasting package that can be used in sales analysis and forecasting, product planning, business planning, etc. Investors can analyze stocks, company trends and growth rates. Financial managers and economists can analyze the general economic climate and investigate business cycles. Even families will find Oracle-80 useful in analyzing spending or energy consumption trends. Oracle-80 should be used by anyone who needs to analyze and forecast monthly, quarterly or annual data.

Even though this package uses advanced statistical analysis, you don't have to understand statistics to use it. Oracle-80 was designed to be used and understood by the typical business person. While Oracle-80 is designed for ease of use, its powerful analytical capabilities will satisfy even the professional forecaster. All input and output are written in plain English and the package documentation carefully explains all the functions of the program.

Adding, deleting and modifying your data is accomplished with a very flexible editor program, integral to Oracle-80. Automatic scaling of inputted numbers, the ability to choose from several output formats, a calendar format that identifies all your data by month/quarter/year, and the ability to add and subtract values while inputting from the keyboard provide added user flexibility. You can use moving average, rate of change, seasonal indices or cycle indices methods to analyze your data. The unique graphing capability of Oracle-80 lets you visualize your historic data or any of the modified data series you calculate. Additionally, you can direct any chart or graph to your printer.

Oracle-80 will forecast future data values using trend, moving average or seasonal methods. You may choose either a constant unit trend or a constant percentage growth trend forecast for even more flexibility.

Oracle-80 is one of the most powerful and useful business tools you will ever use. It puts the future in your hands.

This package requires the following minimum system:

1. A TRS-80 Level II with 16K RAM.
2. An Expansion Interface with 16K RAM.
3. One or more disk drives.
4. Any compatible Disk Operating System.
5. An optional line printer.

Order No. 0152RD (disk-based version) $99.95.

Order No. 0140R (cassette-based version) $75.00

Bowling League Secretary

Instant Software answers the prayers of harried bowling league scorekeepers everywhere by presenting the Bowling League Secretary package, a comprehensive system that lets you use your TRS-80 to maintain a computer-based statistics system for any bowling league.

The package is simple to operate and provides a dynamic reference to all the names of individual bowlers, their team numbers, scores, team names, league data, and all necessary statistics.

The system puts at your fingertips all individual weekly scores, team cumulative scores, bowler cumulative scores, and individual leaders in the following categories: high average, high series, high game and high points.

You can even start your own bowling league scorekeeping business. Bowling leagues will be happy to secure your services if you can provide a detailed printout of their league statistics. Speed, convenience and profitability are the bywords for the Bowling League Secretary package.

This package requires the following minimum system:

1. A TRS-80 with 16K of memory.
2. An Expansion Interface with 16K or more of memory.
3. One minidisk drive, two drives recommended.
4. Any compatible DOS.
5. A tractor-feed line printer.

Order No.0095RD disk-based $49.95.

Other Programs Available from Instant Software

TRS-80* LEVEL I & II

0001R	Basic and Intermediate Lunar Lander	$7.95
0002R	Space Trek II	$7.95
0004R	Beginner's Backgammon/Keno	$7.95
0007R	Ham Package I	$7.95
0008R	Electronics I	$7.95
0009R	Golf/Cross-Out	$7.95
0017R	Air Flight Simulation	$7.95
0019R	Business Package IV	$9.95
0023R	Oil Tycoon	$7.95
0033R	Bowling	$7.95
0043R	Santa Paravia and Fiumaccio	$7.95
0046R	Othello	$9.95
0050R	Grade Book	$9.95
0057R	Chessmate-80	$19.95
0099R	Typing Teacher	$9.95

TRS-80* LEVEL II

0028R	Ramrom Patrol/Tie Figher/Klingon Capture	$7.95
0034R	Space Trek IV	$7.95
0047R	Who-Dun-It?	$7.95
0049R	Demo II	$7.95
0051R	Ball Turret Gunner	$9.95
0055R	Demo III	$7.95
0056R	Bowling League Statistics System	$24.95
0058R	Programmer's Converter	$9.95
0063R	Cards	$7.95
0065R	Teacher	$9.95
0066R	Mimic	$7.95
0068R	Your Cribbage and Checkers Partner	$9.95
0069R	Household Accountant	$7.95
0070R	Skirmish-80	$9.95
0072R	Financial Assistant	$7.95
0076R	TRS-80* Utility II	$7.95
0077R	Enhanced BASIC	$24.95
0081R	TRS-80* Utility I	$7.95
0082R	Daredevil	$9.95
0084R	Music Master	$7.95
0089R	Energy Audit	$49.95
0092R	Archimede's Apprentice	$9.95
0100R	Video Speed-Reading Trainer	$9.95
0103R	Personal Bill Paying	$7.95
0106R	Airmail Pilot	$7.95
0111R	Wordwatch	$7.95
0117R	Night Flight	$9.95
0125R	Investor's Paradise	$9.95
0127R	Surveyor's Apprentice	$9.95
0129R	The Wordslinger	$29.95
0132R	Energy Consumption	$9.95
0135R	Executive Expense Report Generator	$9.95
0136R	Beginner's Russian	$9.95
0137R	Everyday Russian	$9.95
0156R	Money Madness	$9.95
0171R	Flight Path	$9.95
0250R	IRV	$24.95

TRS-80* DISKS

0052RD	Energy Audit	$75.00
0075RD	Accounts Payable/Receivable	$199.95
0095RD	Bowling League Secretary	$49.95
0123RD	The One-D Mailing List	$24.95
0139RD	Disk-Scope	$19.95
0147RD	Check Management System	$39.95
0180RD	Disk Editor	$39.95
0212RD	The Russian Disk	$24.95

PROGRAMS IN GERMAN:

The programs listed here can be purchased through:
MicroShop Bodensee
Markstrasse 3,
7778 Markdorf, West Germany

6004R	Beginner's Backgammon/Keno
6007R	Ham Package I
6008R	Electronics I
6009R	Golf/Cross-Out
6017R	Air Flight Simulation
6028R	RamRom Patrol
6031R	Space Trek III
6034R	Space Trek IV
6043R	Santa Paravia and Fiumaccio
6065R	Teacher
6069R	Household Accountant
6072R	Financial Assistant
6076R	Utility II
6081R	Utility I

TO ORDER: Look for these programs at the dealer nearest you (see list of dealers on page 199). If your store doesn't stock Instant Software send your order with payment to:

Instant Software
Order Dept.
Peterborough, N.H. 03458

(Add $1.00 for handling) or call toll-free 1-800-258-5473 (VISA, MC and AMEX accepted).

*A trademark of Tandy Corporation

Instant Software™

PETERBOROUGH, N.H. 03458
603-924-7296

✓2

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
Professional Data Systems
4506-A N. 16th St., Phoenix

Millets TV & Radio
621 East Broadway, Mesa

California
AMCO Elect. Supply
635 E. Arrow Hwy., Azusa

Byte Shop
6038 Clairmont Mesa Blvd.,
San Diego

Byte Shop
123 E. Yorba Linda, Placentia

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

Computer Mart of California
315 Diamond Bar Blvd., Diamond Bar

Hobbi-tronics
1378 So. Bascom Ave., San Jose

Hobby World
19511 Business Ctr. Dr., Unit 6
Northridge

Huntington Computing
2020 Charles St., Corcoran

I.C.E. House Inc.
398 North E. St., San Bernardino

Jade Computer Products
4901 W. Rosecrans, Hawthorne

Malibu Microcomputing
23910A Deville Way, Malibu

Marfam Co.
6351 Almaden Rd., San Jose

Opamp/Technical Books
1033 N. Sycamore Ave., Los Angeles

PC Computers
10166 San Pablo Ave., El Cerrito

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
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

Computerland
60 Skiff St., Hamden

Computer Works
1439 Post Rd. E., Liberty Plaza,
Westport

Instructional Systems Computers
807 Hartford Rd., Manchester

D.C.
The Program Store
4200 Wisconsin Ave., N.W.
Washington, D.C.

Florida
AI Personal Computer
178 Oxford Rd., Fern Park

AMF Electronics
11146 N. 30th St., Tampa

Boyd-Ebert Corporation
1328 West 15th St., Panama City

Computer Center
6578 Central Ave., St. Petersburg

Computer Junction
5450 So. State Rd. 7, Ft. Lauderdale

Computerland
7374 S. Tamiami Trail, Sarasota

Computerland of Ft. Lauderdale
3963 N. Federal Hwy., Ft. Lauderdale

Computerland of Jacksonville
2777-6 University Blvd. W.
Jacksonville

Computerland of Tampa
1520 E. Fowler Ave., Tampa

Computerland of West Palm Beach
4275 Okeechobee Blvd., West Palm
Beach

Computer Shack
3336 Beach Blvd., Jacksonville

Computer System Resources Inc.
3222 S.W. 35th Blvd., Gainesville

Curtis Waters Enterprises
236 Talbot Ave., Melbourne

Heath Kit Electronic
4705 W. 16th Ave. Center, Hialeah

HIS Computermation
1295 Cypress Ave., Melbourne

Ukatan Computer Store
Airport Rd., Destin

Williams Radio & TV Inc.
2062 Liberty St., Jacksonville

Your Basic Computer Store
971 Seaway Dr., Ft. Pierce

Georgia
Atlanta Computer Mart
Atlanta

Computerland of Atlanta
2423 Cobb Parkway, Smyrna

Micro Computer Systems
3104 E. Shadowlawn N.E., Atlanta

Hawaii
Computerland of Hawaii
567 N. Federal Hwy., Honolulu

Radio Shack Assoc. Store
1712 S. King St., Honolulu

Idaho
Electronic Specialists
8411 Fairview Ave., Boise

Illinois
Computerland
4507 North Sterling, Peoria

Computerland
9511 N. Milwaukee Ave., Niles

Computer Station
3659 Nameoki Rd., Granite City

Garcia & Associates
203 No. Wabash Ave., Suite 1510, Chicago

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
1721 Grant St., Bettendorf

Kansas
Central Kansas Computers
6 S. Broadway, Herington

Maine
Main Computronics
Intown Plaza, Bangor

Radio Shack
315 Main Mall Rd., So. Portland

Maryland
Jack Fives Electronics
4608 Debilen 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
342 Boston Turnpike, Shrewsbury

Lighthouse Computer Software
14 Fall River Ave., Rehoboth

Mark Gordon Computers
15 Kenwood St., Cambridge

New England Electronics Co.
679 Highland Ave., Needham

Small Business System Group
Main St., Dunstable

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 Southfield
29673 Northwestern Hwy., Southfield

Computer Mart
560 W. 14 Mile Rd., Clawson

Computer Room
455 E. Michigan Ave., Kalamazoo

Computronix Corp.
423 S. Saginaw Rd., Midland

Hobby House
1035 W. Territorial Rd., Battle Creek

Main Systems Inc.
1161 No. Ballenger Hwy., Flint

The Alternate Source
1806 Ada, Lansing

The Eight Bit Corner
722 Evanston Ave., Muskegon

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

Softwarehouse
816 Foley St., Jackson

W. Vernon Foster Inc.
816 Foley St., Jackson

Missouri
Century Next Computers
1001 E. Walnut, Columbia

Comp-U-Trs Software Center
51 Florissant Oaks Shopping Center
Florissant

Software Shack
16501 Greenwald Court, Belton

Montana
Intermountain Computer
529 So. 9th St., Livingston

Personal Computer
121 Red Oak Dr., Carl Junction

The Computer Store
1216 16th St. W. #35, Billings

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

Scottsbluff Typewriters Inc.
1824 Broadway, Scottsbluff

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 Corner of NJ
439 Rte. #23, Pompton Plains

Computer Encounter
2 Nassau St., Princeton

Computerland
35 Plaza Rte. #4, W. Paramus

Computer Mart of NJ
501 Rte. 27, Iselin

Crowley's
Rd. #3, Whitehouse Station

Dave's Electronics
Pennsville Shopping Ctr., Pennsville

GHB Enterprises Inc.
Rte. 38, Rudderaw Ave., Mapleshade

Lashen Electronics Inc.
21 Broadway, Denville

Personal Computing Inc.
51 Central Sq., Linwood

Radio Shack/J&J Electronic
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
Autel Electronics Co.
232 Wisconsin N.E., Albuquerque

South West Computer Center
121 Wyatt Drive, Suite 7, Las Cruces

Thomas E. Carr Jeweler
1300A Tenth St., Alamogordo

New York
Aristo Craft
314 Fifth Ave., NYC

Berliner Computer Center
102 Jericho Turnpk., New Hyde Park

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

Computerland of New York City
58 W. 44th St., New York

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

Digibyte Systems Corp.
31E. 31st St., New York

80-Microcomputer Services
118 Masten Ave., Cohoes

Home Computer Center
671 Monroe Ave., Rochester

Mr. Computer
Imp. Plaza, Rte. 9, Wappingers Falls

Softron 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

Sound Mill
Slocum Shopping Ctr., Havelock

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

21st Century Shop
16 Convention Way, Cincinnati

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

TRS-80 Products Ltd.
3520 S.E. Vineyard Rd., Portland

Pennsylvania
Artco Elect.
302 Wyoming Ave., Kingston

Artco Elect.
Back Mountain Shop. Ctr.
Shavertown

Audio Mart
518 Fifth Ave., New Brighton

Computer Workshoppe
3848 William Penn Hwy., Monroeville

Computerland of Harrisburg
4644 Carlisle Pike, Mechanicsburg

Computerland of Pittsburgh
5499 William Flynn Hwy., Gibsonia

Erie Computer Co.
2127 West 8th St., Erie

Mighty Byte Computer Center
537 Easton Rd., Horsham

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

Digital World, Inc.
329 Bald Hill Rd., Warwick

South Dakota
CB Radio Shack
21st and Broadway, Yankton

Tennessee
ACS
1100 8th Ave. So., Nashville

Computerlab
671 S. Menden Hall Rd., Memphis

H & H Electronics Inc.
509 N. Jackson St., Tullahoma

Texas
Computer Port
2142 N. Collins, Arlington

Houston Computer Tech
5313 Bissonet, Bellarie

Interactive Computer
7620 Dashwood, Houston

K.A. Elect.
9090 Stemmons Frwy., Dallas

Pan American Elect. Inc.
1117 Conway, Mission

Radio Shack Dealer
21969 Katy Freeway, Katy

The Compute Shop
6353 Camp Bowie Blvd., Ft. Worth

Waghalter Books Inc.
3 Greenway Plaza E., Houston

Utah
DC Computer Co.
1911 West 70 South, Provo

Quality Technology
470 E. 2nd So., Salt Lake City

Virginia
Computer Works
Rte. 6, Box 65A, Harrisonburg

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

Computer Connection Inc.
3100 NW Bucklin Hill Rd., Silverdale

Computerland of South King Co.
1500 S. 336 St., Suite 12
Federal Way

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

Computerworld
3015 W. Wisconsin Ave., Appleton

Magic Lantern Computed
3313 University Ave., Madison

Petted Microsystems
4265 W. Loomis Rd., Milwaukee

Wyoming
Computer Concepts
1104 Logan Ave., Cheyenne

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

✓Reader Service—see page 226

YOU'VE LOVED OUR WAR GAMES AND OUR FANTASY GAMES, ...AND NOW YOU'LL GO *CRAZY* FOR OUR

5 new COMPUTER GAMES

AVALON HILL'S new Microcomputer Games Division lets you square off against the computer as your opponent in five new cassette games, each with a complete set of instructions and software for use with TRS-80 Level II, 16K; Apple II APPLESOFT Basic, 16K; or PET, 16K.

MIDWAY CAMPAIGN—You command the US Naval forces as the computer commands the invading Japanese.
NORTH ATLANTIC CONVOY RAIDER—Can you cripple the British Home Fleet and force a surrender, or will the computer sink the Bismarck?
NUKEWAR—Strategic conflict from 'cold war' to 'hot'...and your foe is controlled by a cold and calculating computer.
B-1 NUCLEAR BOMBER—Be the Captain of a B-1 bomber in this tactical-level operation, as the computer springs Soviet defenses on your mission.
PLANET MINERS—Four giant corporations engage in intensive exploration and political maneuvering to exploit the resources of space. For one to four players, with the computer playing the part of the other competitors.
All these titles are available for only $15.00 each, plus $1.00 for shipping and our complete listing of computer games.

OVER 35 TITLES AVAILABLE
ASK ABOUT OUR 180-DAY GUARANTEED-SALES POLICY
DEALER & DISTRIBUTOR INFORMATION AVAILABLE ON REQUEST
MAIL ORDERS WELCOME!

✓355
ZOCCHI DISTRIBUTORS
01956 Pass Road
Gulfport, MS. 39531
(601) 896-8600

QUALITY TRS-80 SOFTWARE

KEYWORD Indexing System

A series of programs that will create a data file on disc, build an index of all occurrences of "Keywords" in the text of the data file and allow inquiries or searches into the file using the indexed keywords. The system features
- flexible record lengths with location pointers
- deletion of non keywords from index by system
- "and" "or" "not" logic for inquiries
- interface for user written inquiries

KEYWORD INDEX—2 disc 32K DOS system **$39.95**

SORTS for HOME and BUSINESS

No computer user should be without a versatile, easy to use sort program. The Northeast Microware in memory sort programs are written in Level II BASIC and have the following features
- Sort ALPHA or NUMERIC data
- Sort on up to 5 fields simultaneously
- In ascending or descending sequence
- Supports kb, video or tape I/O
- Supports seq. disk and printer I/O (SORT IID)
- Supports user I/O routines
- User exits (SORT IID only)

SORT II—16K Level II in memory sort **$19.95**
SORT IID—32K DOS in memory sort **$29.95**

FOR the SERIOUS GAMBLER
BLACKJACK SIMULATOR. Allows you to simulate the playing of thousands of hands of BJ and analyze the results on tape in Level II BASIC. **$19.95**

Manuals for all programs available for $3.00 ea.
(price deductible on purchase of program)

"TRS-80 is a registered trademark of TANDY CORP."

Northeast MICROWARE
BOX 2133, ✓74
BOSTON, MA. 02106

BEST SELLING INFORMATION MANAGEMENT SYSTEM IN MICROS TODAY!!

SELECTOR III-C2

- Powerful
- Creates and Maintains Multi- key data bases.
- Prints **FORMATED, SORTED REPORTS** with numerical summaries.
- Source code supplied
- Prints **MAILING LABELS** — and more!
- Comes with **APPLICATIONS PROGRAMS** including:

Sales Activity	Expense Register
Inventory	Checks Register
Payables	Client/Patient Record
Receivables	Appointments
NAD	Library

File management and report writing modules contain linkage to user subroutines to add virtually any special purpose application.

STATE OF THE ART in information management systems!

NEW — "Ready-to-run" version for the TRS-80® Model I, only from Business Microproducts. Also available for Model II.

Requires CP/M operating system or derivative and CBASIC2.

Offered on 5¼" or 8" all versions . . **$295.***
CBASIC2 with Selector Purchase **$75.***

TRSDOS→CP/M

FILE/TRAN

"BRIDGES THE GAP"

- Machine language **COM FILE** directly compatible with your **CP/M** system
- Automated Terminal Configurator
- Memory displayed in both **HEX** and **ASCII**
- Any disk Sector Selected and displayed in both **HEX** and **ASCII**
- Transfers both data and program files by file name byte by byte
- Newly created files scanned for potential errors between level **II BASIC & MBASIC 5.0** or later
- **CP/M** files scanned for any selected string
- Searches any program for all occurences of any string
- Generates a variable cross reference. Invaluable feature for any system level conversion and debugging
- Displays both **CP/M & TRSDOS** directories
- And more

From CP/M: TRSDOS now available for TRS-80 Model I
Both directions $149.00

FILETRAN Disk and Manual **$99**
Manual alone (manual price credited to system) **$20**

BUSINESS MICROPRODUCTS ✓382

A DIVISION OF **THE READY CORPORATION**
LIVERMORE FINANCIAL CENTER
1838 Catalina Court • Livermore, CA 94550
(415) 449-4412

VISA____________________M/C

• NEW •

NEVADA COBOL

- A **POWERFUL** subset of ANSI-74
- A **PRICE** that's **UNBEATABLE**—$99
- **EXTENDED** arithmetic & I/O features
- **FAST** compilation and execution
- **EASY** to use — Generates small executable object modules
- **UNIQUE** — Easily understood error messages
- **CP/M** compatible
- Also available **ON TRS-80**
- **REQUIRES** only **16K-RAM**
- Designed for **PORTABILITY**

STANDARD FEATURES

- Random access file structure
- Sequential files — Fixed and variable length
- Debugging capability
- Copy statement
- Data types & character string, 16 Bit Binary and packed decimal (Comp-3)
- 18-Digit accuracy
- Hexidecimal non-numeric literals
- Powerful editing
- Interactive accept/display

Offered on both 5¼" and 8" diskette, all versions **$99***

**CA residents add 6% sales tax continental shipping $3.00. Allow 2 weeks delivery.*

TRS-80 is a trademark of the Tandy Corp
CP/M is a trademark of Digital Research

✓Reader Service—see page 226

A video driver mod for those whose hands are no quicker than their eyes.

Slow Scroll

Peter A. Lewis
5850 Belt Line Road #1811
Dallas, TX 75240

Having converted from Level I to Level II, I am extremely pleased with all the new features of the more advanced language. One wrinkle that I don't appreciate, however, is the way the program files by on the screen when you list it. In this regard, Level I has a superior system by stopping the list when the screen is almost full and allowing you to hit the up arrow key to move the display up the screen.

I know you can press Shift @ to freeze the display, but I am invariably fumble-fingered and find that the part of the program that I wanted to see has somehow whisked by before I could stop it.

Another thing that bothers me about automatic scrolling is that whenever I write a program that displays more than one screen full of data, I need a "Press Enter to Continue" routine to stop the display.

There are three things I don't like about that procedure. First, you have to keep track of the lines that your program is displaying so you know where to insert the pauses. Second, you cannot use the bottom line of the screen because it is needed for the "PRESS ENTER..." message. Third, if the user has the option to output to the printer, your program has to bypass the pause messages in that case.

A small modification to the Level II video driver solves all of the above. With this modification installed, any line that ends with a new line character (ASCII 13), normally causing the screen to scroll, now has the following effect:

- The display freezes after that line is printed.
- Pressing the up arrow allows normal printing to continue until the next new line character (just like Level I LIST).
- Pressing CLEAR clears the screen and a new screen of data is displayed.
- Pressing Break stops the program or the list and the ready message is displayed.
- Any other key is ignored.

The modification is not active when the cursor is turned on. This allows the screen to scroll normally when you are inputting a long program. You can also use this feature to temporarily turn off the modification within a program. By executing a PRINT CHR$(14) the cursor is turned on and normal scrolling is in effect until you turn it off with a PRINT CHR$(15).

The cursor will also be turned off after an INPUT statement or if the program is restarted from Ready.

Three Methods

You can load the modification in three ways. With any method, the program is completely relocatable. My version ends at the top of a 16K machine, you may want to load it at a different location to accommodate a larger memory size or other machine language routines.

If you have the Editor/Assembler, enter the source code shown in Program Listing 1, assemble it and create a SYSTEM tape. Then initialize the system by entering SYSTEM followed by /0 and set memory size to 32678. Load the SYSTEM tape and enter a / to execute it. Enter CLEAR, and you're in business.

Program Listing 1. Source Code

```
                 ; SCREEN CONTROL - 11/05/79 - PETER A. LEWIS
                 ;
                 ; SET UP DCB DRIVER ADDRESS
                 ;
7FA7                     ORG     32679
7FA7 CD0B00      INIT    CALL    000BH    ;PUT LOCN IN HL
7FAA 110A00              LD      DE,SCREEN-$
7FAD 19                  ADD     HL,DE
7FAE 221E40              LD      (401EH),HL ;ENTRY ADDR TO DCB
7FB1 C3CC06              JP      06CCH    ;RETURN TO READY
                 ;
                 ; CHECK FOR FLAG CHARACTER (00H - OFF, 01H - ON)
                 ;
7FB4 F5          SCREEN  PUSH    AF       ;SAVE FLAGS
7FB5 CD0B00              CALL    000BH    ;PUT LOCN IN HL
7FB8 1801                JR      BYPFLG   ;BYPASS FLAG
7FBA 00                  DEFB    0        ;ON/OFF FLAG
7FBB 23          BYPFLG  INC     HL       ;HL POINTS TO FLAG
7FBC 23                  INC     HL
7FBD 79                  LD      A,C      ;CHARACTER TO A
7FBE E6FE                AND     0FEH     ;LOW BIT OFF
7FC0 2003                JR      NZ,NOFLAG ;NOT A FLAG
                 ;
                 ; SAVE NEW FLAG
                 ;
7FC2 71                  LD      (HL),C   ;STORE NEW FLAG
7FC3 182D                JR      BYPDRV   ;BYPASS DRIVER
                 ;
                 ; TEST FLAG
                 ;
7FC5 7E          NOFLAG  LD      A,(HL)   ;FLAG TO A
7FC6 B7                  OR      A        ;TEST FOR ZERO
7FC7 282B                JR      Z,RSTRA  ;ZERO - BYPASS ROUTINE
                 ;
```

Program continues.

```
                00780 ; TRAP NEW LINE CHARACTER
                00800 ;
7FC9 3A2240     00820         LD      A,(4022H) ;CURSOR CHAR
7FCC B7         00840         OR      A         ;IS CURSOR ON?
7FCD 2025       00860         JR      NZ,RSTRA ;YES - GO TO DRIVER
7FCF 79         00880         LD      A,C       ;CHARACTER TO A
7FD0 FE0D       00900         CP      0DH       ;NEW LINE?
7FD2 2020       00920         JR      NZ,RSTRA ;NO - TO DRIVER
7FD4 2A2040     00940         LD      HL,(4020H) ;CURSOR ADDR
7FD7 114000     00960         LD      DE,40H    ;LINE SIZE
7FDA 19         00980         ADD     HL,DE     ;ADD 1 LINE
7FDB 7C         01000         LD      A,H       ;NEW MSB
7FDC FE40       01020         CP      40H       ;SCREEN OVERFLOW?
7FDE 2014       01040         JR      NZ,RSTRA ;NO - TO DRIVER
                01060 ;
                01080 ; WAIT FOR KEYBOARD ENTRY
                01100 ;
7FE0 CD4900     01120 KBDIN   CALL    49H       ;WAIT FOR KEYBD
7FE3 FE01       01140         CP      01H       ;BREAK?
7FE5 2811       01160         JR      Z,READY ;YES - SEND READY
7FE7 FE5B       01180         CP      5BH       ;UP ARROW?
7FE9 2809       01200         JR      Z,RSTRA ;YES - TO DRIVER
7FEB FE1F       01220         CP      1FH       ;CLEAR?
7FED 20F1       01240         JR      NZ,KBDIN ;NO - READ KEYBD AGAIN
                01260 ;
                01280 ; PROCESS CLEAR KEY
                01300 ;
7FEF CDC901     01320         CALL    01C9H     ;CLEAR SCREEN
                01340 ;
                01360 ; BYPASS DRIVER
                01380 ;
7FF2 F1         01400 BYPDRV  POP     AF        ;RESTORE FLAGS
7FF3 C9         01420         RET               ;BYPASS DRIVER
                01440 ;
                01460 ; RETURN TO DRIVER
                01480 ;
7FF4 F1         01500 RSTRA   POP     AF        ;RESTORE FLAGS
7FF5 C35804     01520         JP      0458H     ;TO DRIVER
                01540 ;
                01560 ; RETURN TO READY
                01580 ;
7FF8 3E0E       01600 READY   LD      A,0EH     ;TURN ON CURSOR
7FFA 322240     01620         LD      (4022H),A
7FFD C3191A     01640         JP      1A19H     ;TO READY MSG
7FA7            01660         END     INIT
00000 TOTAL ERRORS

BYPDRV  7FF2 01400    00620
BYPFLG  7FBB 00440    00400
INIT    7FA7 00200    01660
KBDIN   7FE0 01120    01240
NOFLAG  7FC5 00700    00520
READY   7FF8 01600    01160
RSTRA   7FF4 01500    00740 00860 00920 01040 01200
SCREEN  7FB4 00360    00220
```

```
100 REM - SCREEN CONTROL - 11/8/79 - PETER A. LEWIS
120 DEFINT A-Z
140 CLS:INPUT "ENTER STARTING ADDRESS FOR LOAD";A
160 IF A=0 THEN A=32679
180 FOR X=0 TO 88
200 READ D
220 POKE A+X,D
240 NEXT X
260 PRINT "TO ACTIVATE, ENTER:"
280 PRINT:PRINT,"SYSTEM":PRINT,"/";A:PRINT ,"CLEAR":PRI
    NT
300 PRINT "PRINT CHR$(1) TO TURN ON, PRINT CHR$(0) TO T
    URN OFF"
320 PRINT
340 END
1000 DATA 205,11,0,17,10,0,25,34,30,64,195,25,26,245
1020 DATA 205,11,0,24,1,0,35,35,121,230,254,32,3,113
1040 DATA 24,45,126,183,40,43,58,34,64,183,32,37,121
1060 DATA 254,13,32,32,42,32,64,17,64,0,25,124,254,64
1080 DATA 32,20,205,73,0,254,1,40,17,254,91,40,9,254
1100 DATA 31,32,241,205,201,1,241,201,241,195,88,4,62
1120 DATA 14,50,34,64,195,25,26
```

Program Listing 2. BASIC

If you want the program to be loaded at a different location in memory, just change the ORG statement. Remember to set memory size to one less than that address.

If you have T-BUG, you can enter the object code shown on the left side of Listing 1 by using the M command. Then use the P command to create a SYSTEM tape.

The third method of loading is the BASIC program shown in Program Listing 2. This routine allows you to specify the load address. (Entering 0 loads the program at the top of 16K.)

The modification can be turned on or off by executing a PRINT CHR$(1) or PRINT CHR$(0), respectively.■

✓Reader Service—see page 226

Halve track access time with a software patch.

QWIKDISK

Bruce Nazarian, WD8DRK
GNOME Computerworks
9918 Lauder
Detroit, MI 48227

While doing my homework on the various Disk Operating Systems available for a new pair of mini-floppy disk drives I just acquired, I ran across Apparat's NEWDOS and NEWDOS+. Each looked like an improvement over the TRS-DOS 2.2 I was using, so I scraped up some money and took the plunge.

NEWDOS made disk operations simpler and more reliable. But this was not to be its only benefit.

While researching NEW-DOS+, I read an article by Lance Micklus describing the track-to-track access time used in TRSDOS and NEWDOS (40 ms) and how this time could be improved by a factor of two or four. After reading further, it turned out the changes involved POKEing new values into the DOS from BASIC. The article mentioned that a patch to incorporate these changes would be available as soon as the author had time to work on it.

The Interim

In the interim, I tried everything to get QWIKDISK, as I called it, to operate efficiently.

I wrote a short BASIC program to POKE the values, but that meant I had to load BASIC, fix the access times and return to DOS.

I then tried converting the POKEs into Z-80 machine language and executing the code automatically each time the DOS booted up. This was much faster than BASIC, but still not very elegant.

Then I decided to attack the problem from a different angle. Why wait? After all, the DOS is just a Z-80 machine language program, like any other machine language program. I had done much machine language programming, on 6502 as well as Z-80, so I figured, if the guy who found the fixes could write a patch, maybe I could, too.

First of all, I used a machine language monitor program to dump portions of the DOS around the fixes. That gave me the hex machine code for the program at those points. I also dumped those same locations in ASCII for convenience.

Next, using Superzap (part of NEWDOS+) I examined the system programs on a DOS disk. Since Superzap dumps each disk sector in both hex and ASCII, I used the ASCII I had dumped beforehand to search for a character string that was identical.

Examining sector after sector, I finally hit paydirt in Sector 7. This sector is part of the module called SYS0/SYS. There it was, the identical ASCII string! Looking at the hex dump of that section, I compared it byte-for-byte to be sure I had the identical code. Then, using Superzap, I carefully replaced that one byte with its new value. Superzap wrote the sector back to disk with the new value in place.

With that sector done, I began searching for the ASCII string identical to the second one I dumped. I found it in Sector 9. Checking against my hex dump once more, I ascertained that this was indeed the second POKE point. Again, I used Superzap to correct the value in that sector to the new one and write the sector back to disk.

Now for the acid test. Would the fixed DOS boot up correctly from disk? With some uncertainty I pressed RESET. To my delight the disk bootstrapped at normal speed, then took off like a buzzsaw when the modified program was loaded.

How to Do It

Before I explain in detail how and where to modify your DOS diskettes, a word of warning: Superzap is a very powerful utility and, if used improperly, can easily destroy the system information on your diskette. Be sure you know how to use it correctly.

In addition, this modification is for ••SYSTEM•• diskettes, i.e. DOS diskettes only. Ordinary formatted data diskettes will not have the system programs

```
000700  A847 C356 47C3 0047 C337 47C3 5F47 3E0B .G.VG..G.7G..G>.
000710  F5CD 0046 F1CD 5846 CDF3 45E6 813D C269 ...F..XF..E..=..
000720  463A 0943 32E1 3718 EF01 1467 44C3 CF44 F:.C2.7.....D..D
000730  C3DF 44C3 B74C C3D2 4C3E D3EF 3EE3 EF01 ..D..L..L>..>...
000740  7180 444E 414D 454E 414D 452F 4558 542E ..DNAMENAME/EXT.

000700  A847 C356 47C3 0047 C337 47C3 5F47 3E0A .G.VG..G.7G..G>.
000710  F5CD 0046 F1CD 5846 CDF3 45E6 813D C269 ...F..XF..E..=..
000720  463A 0943 32E1 3718 EF01 1467 44C3 CF44 F:.C2.7.....D..D
000730  C3DF 44C3 B74C C3D2 4C3E D3EF 3EE3 EF01 ..D..L..L>..>...
000740  7180 444E 414D 454E 414D 452F 4558 542E ..DNAMENAME/EXT.
```

Fig. 1. Partial dump of NEWDOS, Track 0, Sector 7 showing (top) byte #OF before modification and (bottom) the same byte after modification to 20 ms access time. Columns at left indicate Drive #(0), Track #(00), Sector #(7) and starting byte number of each line (00, 10, 20, etc.)

```
000900  0046 CD69 463A ED37 BAED 53EE 37C8 3E1B .F..F:.7..S.7.>.
000910  F5CD 6946 F132 EC37 C907 3005 3ED0 32EC ...F.2.7..0.>.2.
000920  37CD F345 CB47 C818 F032 9446 E3C5 4623 7..E.G...2.F..F0
000930  7E23 32C9 467E 2366 6F22 A846 E1E3 CD47 .02.F.0..".F...G
000940  46CD 5544 C5D5 E521 EC37 F336 0011 EF37 F.UD...!.7.6...7

000900  0046 CD69 463A ED37 BAED 53EE 37C8 3E1A .F..F:.7..S.7.>.
000910  F5CD 6946 F132 EC37 C907 3005 3ED0 32EC ...F.2.7..0.>.2.
000920  37CD F345 CB47 C818 F032 9446 E3C5 4623 7..E.G...2.F..F0
000930  7E23 32C9 467E 2366 6F22 A846 E1E3 CD47 .02.F.0..".F...G
000940  46CD 5544 C5D5 E521 EC37 F336 0011 EF37 F.UD...!.7.6...7
```

Fig. 2. Partial dump of NEWDOS, Track 0, Sector 9 showing (top) unmodified and (bottom) modified second byte of Qwikdisk modification. Same modification may be made to either 35 or 40 track versions.

```
000700  B920 042C 7E18 1871 2643 6FCD 6946 3AED ...,....&C...F:.
000710  3777 796F 7E32 ED37 79CD 3946 3209 436F 7....2.7..9F2.C.
000720  3AEC 3707 7D32 E137 3008 2100 002B 7CB5 :.7..2.70.!..+..
000730  20FB E1C9 E607 0707 07F6 C732 4546 AFCB ............2EF..
000740  C7C9 CD00 46CD 6946 3AED 37BA ED53 EE37 ....F..F:.7..S.7
000750  C83E 1FF5 CD69 46F1 32EC 37C9 CD58 46F5 .>....F.2.7..XF.
000760  F100 0000 CDAC 43CB 47C8 18F8 3293 46E3 ......C.G...2.F.
000770  C546 237E 2332 D346 7E23 666F 22A7 46E1 .F0.#2.F.0..".F.
000780  E3CD 4746 CD69 46C5 D5E5 21EC 3736 0011 ..GF..F...!.76..
000790  EF37 C1C5 C1C5 F318 030F 300C 7ECB 4F28 .7.........0...O(
0007A0  F8F3 0000 03C3 A146 FB7E E67C 36D0 E1D1 ........F....6...
0007B0  C128 23CB 5720 13FE 2028 11F5 3EFF 3208 .(0.W....(..>.2.
0007C0  43CD 0046 3E0B CD58 46F1 10B5 473E 00CB C..F>..XF...G>..
0007D0  0838 033C 18F9 C1C9 3E08 CD71 460A 011A .8.<....>...F...
0007E0  023E A8CD 7146 0509 0A12 3EA9 18F5 3E08 .>...F....>...>.
0007F0  CD71 4605 011A 0001 0542 44C3 0047 0139 ..F......BD..G.9

000700  B920 042C 7E18 1871 2643 6FCD 6946 3AED ...,....&C...F:.
000710  3777 796F 7E32 ED37 79CD 3946 3209 436F 7....2.7..9F2.C.
000720  3AEC 3707 7D32 E137 3008 2100 002B 7CB5 :.7..2.70.!..+..
000730  20FB E1C9 E607 0707 07F6 C732 4546 AFCB ............2EF..
000740  C7C9 CD00 46CD 6946 3AED 37BA ED53 EE37 ....F..F:.7..S.7
000750  C83E 1AF5 CD69 46F1 32EC 37C9 CD58 46F5 .>....F.2.7..XF.
000760  F100 0000 CDAC 43CB 47C8 18F8 3293 46E3 ......C.G...2.F.
000770  C546 237E 2332 D346 7E23 666F 22A7 46E1 .F0.#2.F.0..".F.
000780  E3CD 4746 CD69 46C5 D5E5 21EC 3736 0011 ..GF..F...!.76..
000790  EF37 C1C5 C1C5 F318 030F 300C 7ECB 4F28 .7.........0...O(
0007A0  F8F3 0000 03C3 A146 FB7E E67C 36D0 E1D1 ........F....6...
0007B0  C128 23CB 5720 13FE 2028 11F5 3EFF 3208 .(0.W....(..>.2.
0007C0  43CD 0046 3E0A CD58 46F1 10B5 473E 00CB C..F>..XF...G>..
0007D0  0838 033C 18F9 C1C9 3E08 CD71 460A 011A .8.<....>...F...
0007E0  023E A8CD 7146 0509 0A12 3EA9 18F5 3E08 .>...F....>...>.
0007F0  CD71 4605 011A 0001 0542 44C3 0047 0139 ..F......BD..G.9
```

Fig. 3. Full sector dump of TRSDOS 2.2, Track 0, Sector 7. Both bytes to be modified are present in this sector and the figure shows (top) unchanged and (bottom) changed information.

on them.

This modification further assumes that the SYS0/SYS module will be at the same location on each system diskette. If you find that the byte at the specified location is not what I have indicated, *don't change it.* First, make sure you have the right track and sector before you alter any program bytes.

For NEWDOS or NEWDOS+, the bytes to change are located in two different sectors, as described in the story above. First, modify Track #0, Sector 7, relative byte # 0FH (Fig. 1).

The original value should be 0BH. This gives 40 ms track access time. For 20 ms change this byte to 0AH, and for 10 ms change it to 09H.

The second location is Track #0, Sector 9, relative byte # 0FH (Fig. 2). Normally, this byte would be 1BH. For 20 ms access, change it to 1AH, and for 10 ms make it 19H.

Modifying TRSDOS 2.2

If you have TRSDOS 2.2, you will need the Superzap utility included in NEWDOS+. But, if you can acquire it your disk changes are easy.

Both of the TRSDOS changes are made in one sector. First: Track 0, Sector 7, relative byte # 52H (Fig. 3). Its 40 ms value is 1FH. Change this to 1AH for 20 ms or 19H for 10 ms. Then change relative byte #C5H. Its 40 ms value is 0BH. Change it to 0AH for 20 ms, or 09H for 10 ms. That's all there is to it. (Fig. 3).

It will save a lot of time and effort if you try both of these access times out first. Some drives (Shugart, for example) are not capable of 10 ms access times, but can take 20 ms. Other drives (Percom, M.P.I., etc.) are capable of 5 ms access times.

Try the different times with your drives, and then Superzap the appropriate values onto your DOS diskettes. ■

```
000700  B920 042C 7E18 1871 2643 6FCD 6946 3AED ...,....&C...F:.
000710  3777 796F 7E32 ED37 79CD 3946 3209 436F 7....2.7..9F2.C.
000720  3AEC 3707 7D32 E137 3008 2100 002B 7CB5 :.7..2.70.!..+..
000730  20FB E1C9 E607 0707 07F6 C732 4546 AFCB ............2EF..
000740  C7C9 CD00 46CD 6946 3AED 37BA ED53 EE37 ....F..F:.7..S.7
000750  C83E 1BF5 CD69 46F1 32EC 37C9 CD58 46F5 .>....F.2.7..XF.
000760  F100 0000 3AEC 37CB 47C8 18F8 3293 46E3 ....:.7.G...2.F.
000770  C546 237E 2332 D346 7E23 666F 22A7 46E1 .F0.#2.F.0..".F.
000780  E3CD 4746 CD69 46C5 D5E5 21EC 3736 0011 ..GF..F...!.76..
000790  EF37 C1C5 C1C5 0018 030F 300C 7ECB 4F28 .7.........0...O(
0007A0  F8F3 0000 03C3 A146 FB7E E67C 36D0 E1D1 ........F....6...
0007B0  C128 23FE 2028 15CB 5720 0FF5 3EFF 3208 .(0..(..W...>.2.
0007C0  43CD 0046 3E0B CD58 46F1 10B5 473E 00CB C..F>..XF...G>..
0007D0  0838 033C 18F9 C1C9 3E08 CD71 460A 011A .8.<....>...F...
0007E0  023E A8CD 7146 0509 0A12 3EA9 18F5 3E08 .>...F....>...>.
0007F0  CD71 4605 011A 0001 0542 44C3 0047 0139 ..F......BD..G.9
```

Fig. 4. Full sector dump of TRSDOS 2.1, Track 0, Sector 7. Similar to TRSDOS 2.2, relative byte #52 has original value of 1BH rather than 1FH as in TRSDOS 2.2. The remaining byte is the same in both TRSDOS's.

```
000000  00FE 11F3 31FC 4121 E242 CD9A 423E 0132 ....1.A!.B..B>.2
000010  E137 3A02 4257 1E04 0100 4DCD AA42 2070 .7:.BW....M..B..
000020  3A00 4DE6 1021 E542 2869 D92A 164D 557C :.M..!.B(..*.MU.
000030  0707 07E6 0767 0707 845F 01FF 4DD9 CD75 ..............M...
000040  423D 2017 CD75 4247 CD75 426F 05CD 7542 B=....BG..B....B
000050  6705 28EA CD75 4277 2318 F63D 280B CD75 ..(...B.0..=(...
000060  4247 CD75 4210 FB18 D5CD 7542 CD75 426F BG..B.......B..B.
000070  CD75 4267 E9D9 0C20 14C5 3E01 32E1 37CD ..B.......>.2.7.
000080  AA42 200C C11C 7BD6 0A20 025F 140A D9C9 .B..............
000090  21F1 42CD 9A42 CD40 0076 E57E FE03 2008 !.B..B.@.......(.
0000A0  CD33 0023 FE0D 20F3 E1C9 C5CD B242 E1C8 .3.0.........B..
0000B0  444D ED53 EE37 21EC 3736 1BF5 F1F5 F17E DM.S.7!.76......
0000C0  0F38 FC36 08D5 11EF 37C5 C118 0B0F 300A .8.6....7.....0.
0000D0  7ECB 4F28 F81A 0203 18F6 7EE6 5CD1 C836 ..O(...........6
0000E0  D0C9 1C1F 0317 E84E 4F20 5359 5354 454D ........NO.SYSTEM
0000F0  0D17 E844 4953 4B20 4552 524F 520D EB5F ...DISK.ERROR...
```

Fig. 5. Full sector dump of BOOT/SYS, Track 0, Sector 0. Relative byte BAH is underlined. See footnote for more information on this modification.

Footnotes

* Recent information from APPARAT also mentions a third modification that is necessary for this fix. It involves a change to 1 byte in the BOOT/SYS module. This will make the BOOTSTRAP loader work at the higher step times also. This change may be made to TRSDOS 2.1, 2.2 or NEWDOS. In Track 0, Sector 0, change the relative byte BAH as follows: Retain original value (1FH in TRSDOS 2.2; 1BH in 2.1 and NEWDOS) for 40 ms step time. For 20 ms, change to 1AH. For 10 ms, change to 19H.

** Further information also indicates values for 5 ms step times (which may not work on other than MPI drives). For bytes that are changed in the 1BH, 1AH, 19H, sequence, 18H will give 5 ms step times. For bytes that use the 0BH, 0AH, 09H sequence, 08H gives 5 ms step time. Both values should correlate: If one byte is 1BH, the other should be 0BH. If one byte is 1AH, the other should be 0AH. Likewise for 19H; 09H, and 18H; 08H.

The solution to a loopy problem.

The "Next" Trap

Hubert C. Borrmann
2840 South Circle Drive #209
Colorado Springs, CO 80906

I fell into the "NEXT" trap when I tried to run the big ROM and RAM test in appendix C of the Level II manual.

Have you fallen into this sneaky trap yet? If not, you surely will sooner or later. What astounds me is, that I have so far seen no reference to this, not even in the *Learning Level II* book by D. A. Lien. ?NF ERROR was looking at me from the screen.

If you want to fix this program, insert the following line:

```
280 X = 99:NEXT X
```

Level II is a good interpreter, but reminds me of one of my teachers many many years ago who had an astounding memory for things that should have been forgotten. To explain this, type Listing 1, RUN and type 10.

OK, this is the way it should work. Now RUN it and type in a 5.

```
NF ERROR
```

Unfinished Business

This is a simple program, containing three FOR-NEXT loops. The first one in line 30 and 40,

The program then enters two nested loops and everything works fine until the Variable B is checked in line 110; where B should be changed from one or two for the second pass through the B-loop. But the interpreter won't let you do this and out executes nine times unless your entered number makes you jump out of the loop to line 50, and the loop will not run its course, leaving behind "unfinished business." (This is what the interpreter remembers, but should have forgotten.)

```
10 CLEAR 200 : CLS
20 PRINT@ 128,"INSTRUCTIONS # 1"
30 S1$="PRESS 'SPACE' WHEN READY"
40 GOSUB 10000
50 CLS : PRINT@ 512,"INSTRUCTIONS # 2"
60  SA=640
70 GOSUB 10100
80 CLS : PRINT@ 0,"AND MORE INSTRUCTIONS"
90 S1$="TYPE 'SPACE'-BAR TO CONTINUE, 'E' TO END"
100 GOSUB 10000
110 IF S6$="E" THEN END ELSE 20
```

Program Listing

comes the message ?NF ERROR IN 110.

You see, the interpreter remembers that unfinished business with Variable A and won't accept NEXT B until A is cleared. You know that a NEXT statement will let the program fall through if the variable involved is larger than the TO figure given in the FOR-TO statement. The NEXT statement adjusts the variable and then makes this test. So, fortunately our fix is easy for situations where you jump out of FOR-NEXT loops. Insert this line in our little program, and we are in business:

```
55 A = 999:NEXT A
```

Now RUN and type a 5.

Don't be misled if you enter the number 9! It still jumps out of the loop! I hope that this may clear up some unexplained phenomena and help you avoid the NEXT trap. ■

« MISOSYS »

diskmod Turn your Editor Assembler into a disk package. This patch modifies EDTASM 1.1 & 1.2 under 32K TRSDOS, NEWDOS, or VTOS! Features? Add full disk I/O, block move, global change, printer pagination with optional prompting, sorted symbol table, display memory utilization, correct DEFM expansion, protect memory, and recover after BOOT. From within the EDTASM you will have DIR, KILL, & FREE. This package is a must for assembler programmers! Priced at $20.

dsmblr ii Complement your assembly language tools with this Z80 disassembler which produces screen, printer, or disk file output. A 2-pass process provides SYMBOLS for 16-bit address and 8-bit relative references. EQUates & ORG are generated. New version reads SYSTEM programs & displays load addresses. Priced at $20 for disk systems. A 16K Tape Version (DSMBLR I) is available for $15.

cmdfile Now you can append 2 or more CMD files and/or SYSTEM tapes. Perform transfer to & from disk/tape of SYSTEM/CMD modules with offset capabilities. Provides PATCH, TAPEDISK, & LMOFFSET capabilities as well as adds a few of its own. Get CMDFILE today! 16K req'd. $20.

dutil All-purpose utility to examine, clear, initialize, move, and modify data in memory. Load, punch, verify SYSTEM tapes. Disk sector I/O. More! $20 (TUTIL, for the non-disk user available at $15).

MISOSYS Department K ✓221
5904 Edgehill Drive
Alexandria, Va. 22303
703-960-2998

master charge VISA

T H E B Ø Ø K

THE BØØK must be a part of your Z-80 language tools. Volume I gives you access to all math operations in your Level II ROM including ASCII-binary conversions. A fully-commented listing of math routines provides detailed explanation of how they work. Included is a symbol table of the entire machine noting over 500 addresses. THE BØØK will save you hours of assembler program development time. Don't start programming without THE BØØK. Order your copy for $14.95+$1.50 S&H.

IF YOUR CUSTOMER MAILING LIST HAS YOU NAILED...

behind a desk for hours at a time because of constant updates and changes, **dial (617) 373-1599** and we'll explain our Customer Control Mail List System that has 670 customers per diskette; will store name and address information, reference code, plus variable selection codes for each customer; will display customer information of a video screen, print reports and mailing labels; will sort 670 names in less than 30 seconds; select, sort, and print 670 mailing labels in just over 30 minutes; has Partial Key Lookup and instantaneous display; has select/sort options by record sequence, last name, city, state, zip code or reference code; ***and has a low price of only $99.00!***

S&M SYSTEMS, INC.

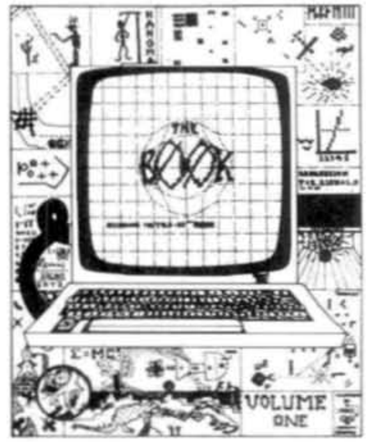
P.O. Box 1225 • 2 Washington Street
Haverhill, Massachusetts 01830

BACK TO BASIC

THE BØØK

ACCESSING THE TRS-80* ROM

If you ever do Assembly language programming, or you just want to know more about your TRS-80 ROM, "THE BØØK" is for you.



Volume I will give you access to over fifty machine language subroutines in the Radio Shack Level II BASIC. It includes information on the numeric data formats and a commented listing of the ROM routines.

"THE BØØK, Volume I", encompasses all arithmetic functions and mathematical operations. There are separate routines for integers, single precision, and double precision numbers and the data format for each of these number types is explained. The routines that perform ASCII to binary and binary to ASCII conversion are identified and explained to provide you a means of data I/O.

A fully commented listing provides the details on the step-by-step execution of these ROM routines. Although a complete disassembly is not provided in order to avoid copyright infringement, you can obtain a complete disassembly using the disassembler program listed in "THE BØØK." Volume I also includes a complete, detailed memory map of the entire machine and a symbol table noting over 500 addresses.

"THE BØØK" will save you hour upon hour of assembler program development time. Don't start programming without it.

Order your copy of "THE BØØK", today!

DEALER INQUIRIES INVITED

**Insiders Software Consultants, Inc.
P.O. Box 2441, Dept. M 1
Springfield, VA 22152** ✓305

*TRS-80 is a trademark of Tandy Corp.

☐ **Please send me Volume I of THE BØØK at $14.95 plus $1.50 for postage.**

NAME: ______________________
ADDRESS: ______________________
CITY, STATE ______________ & ZIP CODE: ________

☐ Check payable to Insiders Software Consultants, Inc.
☐ MASTER CHARGE MC Bank Code:
☐ VISA Exp. Date: Card Number:

Signature: ______________________

Get it with an Inkey$.

The Competition's Cursor

R. Daniel Bishop
Department of Chemistry
The School of the Ozarks
Point Lookout, MO 65726

Surely you know how it is when you select one gadget or machine from among a variety of competing models, wishing all the while that the model you chose had some of those dandy options offered by the competition. Buying a microcomputer requires choices and compromises, and even though I felt the TRS-80 had enough of the options I wanted in a microcomputer, I still found myself taking sidelong glances at that infernal, insistent flashing cursor that the competition offered.

Well, those feelings are a thing of the past, because now I can write my own flashing cursor into TRS-80 Level II BASIC programs.

INKEY$ Command

The key to the flashing cursor is the INKEY$ command and the graphics character printed out by the CHR$(143) command. While writing the main program, you need to determine the display position (0 to 1023) where the blip is supposed to appear each time it is used. Assign the value of this position to the variable CR and then use a GOSUB 1000 to transfer to the Flashing Cursor Subroutine (Listing 1).

Each character in the input string is stored sequentially in the N$(j) array while the counter J keeps track of how many characters have been typed. Once the enter key is depressed, the subroutine is transferred to the loop (lines 1012 to 1015) which adds these characters one at a time onto the string variable N$, so that the complete input information can be used later in the program.

CR is incremented by one after each keyboard entry except when the backspace or the enter keys are depressed (having ASCII values of 8 and 13, respectively). The flashing cursor moves one more space, awaiting the next input.

A simple program, using this routine, follows:

```
1 CLEAR 500
2 DIM N$(30)
5 CLS
10 PRINT"WHAT IS YOUR
NAME?"
15 CR=20
20 J=O
25 GOSUB 1000
30 PRINT
35 PRINT"I AM VERY GLAD TO
MEET YOU. ";N$;"."
40 END
```

```
998 END
999 J=J-1:IF J>=0 THEN CR=CR-1:N$(J)=""ELSEJ=J+1
1000 PRINT@CR,CHR$(143)
1001 FOR I=1TO5:NEXTI
1002 PRINT@CR," "
1003 FOR I=1TO5:NEXTI
1004 R$=INKEY$
1005 IF LEN(R$)=0 THEN 1000
1006 IF ASC(R$)=13 THEN 1012
1007 IF ASC(R$)=8 THEN 999
1008 PRINT@CR,R$
1009 N$(J)=R$
1010 CR=CR+1
1011 J=J+1:GOTO1000
1012 N$=""
1013 FOR I=0 TO J-1
1014    N$=N$+N$(I)
1015 NEXTI
1016 RETURN
```

Program Listing 1

In addition to specifying the starting cursor position, CR, in the main program, the input counter J must be initialized, and the array N$(J) is DIMensioned. The four lines, 2, 15, 20 and 25, take the place of the usual INPUT N$ statement. Note that the input N$ is returned as a character string. However, since INKEY$ was used to construct this character string, commas and quotation marks can all be used as valid input. The enter key is the only string delimiter that is recognized. If the input is used as numeric data, one additional instruction, 27 N = VAL(N$), must be inserted.

If only a single letter or number is to serve as the total input, as in selecting an option from a menu, the flashing cursor subroutine can be streamlined using only lines 1000 through 1005 and adding 1006 RETURN.

The input character is returned in this case as R$ rather than N$. This requires little more memory than INKEY$ normally uses and provides a far superior visual addition to the program.

Natural Restrictions

There are some natural restrictions imposed by using this routine in place of the INPUT command. The first is that the routine slows down keyboard response. A rapid-fire touch typist would go crazy trying to slow down to the computer's speed.

A second limitation is the size of the N$(J) array. Obviously, if an input string exceeds the array size chosen, the program crashes. Also, although the same array may be used (without clearing) for all input in the program, it still consumes some memory which may be put to better use.

A way around this problem is to use the video monitor as the array and read these characters directly from the display, adding them one at a time to N$. The routine that performs this task is a simple loop using the PEEK function.

```
1012 N$="":CR=CR-J
1013 FOR I=15360+CR TO 15360
+CR+J
1014 N$=N$+CHR$(PEEK(I))
1015 NEXTI
1016 RETURN
```

This loop replaces lines 1012-1016 in Listing 1. Line 1009 can be deleted and :N$(J) = " " in line 999 should be removed. This allows lines 1 and 2 in the main program to also be removed. All other lines remain the same. ■

A stock and sales plan designed to maximize profit.

Down the Road

Bill Vick
3325 Landershire Ln.
Plano, TX 75023

As a regional sales manager with a major consumer products company, I am constantly challenged with finding new and creative ways of filling the product pipeline between my company and the consumer. This pipeline, or channel of distribution, consists of department stores.

The way a store projects its future sales and determines the stock needed to support those sales is a basic financial planning concept, applicable to large or small businesses.

Our company now furnishes a stock and sales plan designed to maximize profits, as well as a profitability and return on investment analysis. These reports are generated on my TRS-80, and this article covers the planning process and its application.

The Retail Season

Retailers periodically go through a planning process forecasting sales for a given period of time, called a season. Normally a season runs six months and is referred to as a "spring" or "fall" season. Whether the sales forecast is done with linear regression, exponential smoothing, S.O.T.P. (seat of the pants), or any other means, the end result is that merchandise must be purchased in advance of planned sales. The control and management of the flow of inventory through a retailer determines whether or not a profit is made.

Plans are normally drafted for both seasons and the fiscal year. A turnover ratio of stock to sales is predetermined, based on the needs of both the retailer and the vendors involved. In a simplified way, turnover is the number of times an average inventory will replace itself as it is sold.

As an example, if over a six month period, or season, sales were $10,000, with an average inventory of $8,000, the turnover ratio would be 1.25 for the season. The turnover ratio is critical. Too fast a turnover means in all probability sales were lost as a result of being out of stock, while too slow a turnover can eat into profit and increase your overhead. Generally, the faster the turnover, the greater the profit.

Two Seasons

In my business, we work on two seasons a year, which go from February to July and August to January. The average inventory turnover is 3.0 per year, or 1.4 turns in the spring and 1.6 in the fall.

The following program determines monthly stock levels and planned purchases needed to maintain stock levels. A sales

```
WHAT YEAR IS THE PLAN FOR (YY)      ? 80
WHAT IS THE CHAIN'S NAME            ? ANY STORE
WHAT IS THE BRANCH/DOOR'S NAME  ? MAIN ST.
WHICH SEASON WILL THIS PLAN COVER (FALL OR SPRING) SPRING

ENTER THE 79 SALES FOLLOWED BY A COMMA, THEN THE 80 SALES
 79/80 SALES FOR FEB? 11.1,12.2
 79/80 SALES FOR MAR? 10.9,13.1
 79/80 SALES FOR APR? 11.5,10.7
 79/80 SALES FOR MAY? 12.4,15.2
 79/80 SALES FOR JUN? 9.2,9.9
 79/80 SALES FOR JLY? 11.6,11.9

PLANNED SEASON TURN? 1.39
IF YOU HAVE AN ACTUAL BOM ENTER IT.
IF YOU DO NOT PRESS ENTER ?_
```

Example 1.

```
          — 1980 STOCK/SALES PLAN FOR ANY STORE — MAIN ST.
           FEB     MAR     APR     MAY     JUN     JLY     TOT
79 SALES   11.1    10.9    11.5    12.4    9.2     11.6    66.7
80 SALES   12.2    13.1    10.7    15.2    9.9     11.9    73.0

BOM        52.6
EOM        53.5    51.1    55.6    50.3    52.3    52.5

OTR        13.1    10.7    15.2    9.9     11.9    12.2

TURN 1.39       AVG INV 52.5        TOTAL SALES 73.0        + 9%
```

Example 2.

its...SOFTWARE

SE Search Entry

SE is a super fast, general purpose information retrieval program for the TRS 80*. The uses of **SE** are limitless. It can be a file system, a matching service, an inventory control, or a message center. Whenever fast searching of large amounts of data is needed, **SE** can be your program.

Written in Z80 machine language, **SE** compares tens of thousands of characters in a few seconds. Simple commands add, change, or remove data entries. To search for entries, up to 64 characters can be combined as targets for immediate retrieval. **SE** occupies only 4K; the rest is storage and can be saved on tape or disk.

The tape version for 16K level II, SE2.0, is $24.95, while the disk version for DOS up to 48K, SE3.0, is $49.95. The price includes full documentation.

Other TRS-80 its...Products

COMPU-DIET - Weight Loss System
Behavior Mod, Forecast, Database
MINIVENT - Minimal Inventory Control
1400 items, 16K Level II
BASICIO - Machine Language I/O for BASIC
Data and Programs to Tape or Disk

Please send

SE2.0(tape) @$24.95 — **COMPU-DIET 1.2** @$19.95
SE3.0(disk) @$49.95 — **MINIVENT 2.0** @$49.95
BASICIO 1.5 @$14.95 — Additional Information

Fla. Res. add 4% Sales Tax. Total $ ______
Check/Money Order   VISA   MASTERCHARGE
Card No ______ Exp. Date ______
Name ______ Bank No. (MC) ______
Address ______

✓300

its... Information Technology Systems
POB 2667 Sarasota FL 33578
(813) 366-0064

*TRS 80 is a trademark of Radio Shack, a Tandy Corporation

If you're serious about the stock market, you need Tickertec™

Watch 48 to 400 of your favorite stocks without a 15 minute delay.

Tickertec™ is a computer program that displays the NYSE or AMEX tickertape on your TRS-80™ Model I or both exchanges as an option on the Model II. You see every trade as it is reported by the exchange and track the last ten trades, tickertape reported volume, and high and low limits on the stocks you are watching. Tickertec program prices start at $1,000.00 with many optional features available including hard copy and portfolio management systems. Programs may be purchased for cash (i.e., hard dollars) or payment can be arranged in the form of discounted brokerage commissions (i.e., Soft Dollar Software™). Exchange fees are extra. Call for FREE brochure TOLL-FREE at **(800) 223-6642**; in New York call **(212) 687-0705**; or circle the reader service number.

MaxUle& Company Inc. ✓428
6 East 43rd Street, N.Y., N.Y. 10017

THE ALTERNATE SOURCE

The magazine of advanced applications and software for the TRS-80

✓138

If you keep this up, you'll not only have a winner, but you will also provide a valuable service to many, many TRS-80 computerists. HOUSTON, TX

I think your approach is terrific; a magazine devoted to someone other than the beginner has been sorely needed on the market for a long time. PENDLETON, IN

Along with the OCTUG Newsletter, yours is the best I've seen for important info on the TRS-80. ATLANTA, GA

I'd like to say that I'm very glad I took the gamble and subscribed to The Alternate Source News; it looks like I made the right choice. Keep up the good work! MAKAWAO, HI

Of the several TRS-80 publications I take, I like The Alternate Source News the best so far! STUDIO CITY, CA

I was so pleased at being approached as if I knew something about computers' and programming, that I not only read every word, but am also enclosing a check for nine dollars. CANOGA PARK, CA

I just received my copy of Issue #2 of TAS. I'm really impressed. You've done a great job! WAUKEGAN, IL

Okay, I'm hooked. Enclosed is my $9.00 check. FRANCONIA, VA

IF YOU'RE SERIOUS ABOUT PROGRAMMING, YOU SHOULD SERIOUSLY CONSIDER THE ALTERNATE SOURCE NEWS.

☐ Sounds good! Enclosed is $9.00 for six issues. Please start my subscription with issue number ______

☐ Yeah, it sounds good, but I'd like to see a copy before I subscribe. Enclosed is $2.00 for the current issue.

NAME ______
ADDDRESS ______
CITY, STATE, ZIP ______

Mail to: The Alternate Source, 1806 Ada Street, Lansing, MI 48910

BOWLING LEAGUE STATISTICS

Leaders, Games, Pins, Handicaps, Blind Scores, Teams, Players — High Game, High Series, Lanes, Substitutes, Points, Scratch, Averages

TRS-80*
32K — DOS — PRINTER

BOWLSTAT 1.1 will generate team standing sheets suitable for posting with full team and player statistics, and also print league scoring sheets. Printing format is optimized for the 779 printer, but a modification for any printer with 80 characters minimum per line is provided. A wide range of handicapping and scoring options meets almost any league's requirements. Entry of weekly scores is fast and simple with extensive prompting and error detection.

Supplied on Diskette with manual
$49.95

Manual Only
(deductable on purchase of diskette)
$5.95

Pa. residents add 6% tax

Datascore ✓193
P. O. Box 995 Sharon, Pa. 16146

*Trademark of Tandy Corp.

16K MEMORY

TMS 4116-30JDH

16K × 1 Dynamic RAMs for: TRS-80™, Apple, Exidy and others.

8 $59.00

8 $74.00

for Canadian customers

Add $3 to help cover postage and handling. Sask. residence add 5% sales tax.

TRS-80™ installation information is available on request.

TAITRONICS ENTERPRIZES
1263 Wallace St. ✓380
Regina Sask. Can.
S4N 3Z2 ph. (306) 527-2088

FINALLY!

PROFESSIONAL PROGRAMMING TOOLS FOR EVERY BASIC PROGRAMMER

Do you write Basic programs with your TRS-80?* Then you know there are times when you need to:

- **RENUMBER** part or all of your program.
- **DELETE** many lines with one command.
- **APPEND** common routines from tape.

Now, with **PROgrammer,** you can do **all** these things! And **COMPRESS** your programs to speed execution and reduce size. And **MOVE** blocks of lines within your program quickly and efficiently. Moreover, load **PROgrammer** just once, and *these commands are always available!*

PROgrammer: $25 on cassette with complete instructions. Level II required. Please indicate memory size with order. Calif. residents add 6% sales tax. *TRS-80 is a trademark of Tandy Corp.

RATIONAL SOFTWARE™
963 E. California Pasadena, CA 91106
✓241

history for the prior period is shown, along with the B.O.M. (beginning of month) inventory, the E.O.M. (end of month) inventory, and the allowed monthly receipts or O.T.R. (open to receive).

Additionally, the average inventory carried on hand and the percent of increase or decrease planned is also shown. It is an interactive program designed to be used by people not versed in either computerese or retailing.

The advantages of this type of planning are many. Not only can the retailer's stock and sales plans be done with speed and accuracy, but a number of "what if" situations can quickly be accomplished to determine viable alternatives.■

Program Listing

```
100 CLEAR 1000 : CLS
110 REM *  STOCK AND SALES PLANNING
120 REM *  BILL VICK  /  1/30/80
130 REM *  (214) 596-0533
140 PRINT TAB(20)"STOCK & SALES PLANNING"
150 PRINT TAB(25)"BY BILL VICK"
160 PRINT TAB(22)"ALL RIGHTS RESERVED"
170 FOR T=1 TO 500 : NEXT T
180 DIM M(13), M$(13), EOM(13), Z$(2), OTB(13), S(13)
190 RESTORE : CLS
200 INPUT"WHAT YEAR IS THE PLAN FOR (YY)  ";YEAR$
210 YEAR$=RIGHT$(YEAR$, 2) : YEAR=VAL(YEAR$) : LYEAR=YE
AR-1
220 INPUT"WHAT IS THE CHAINS NAME         ";CHAIN$
230 INPUT"WHAT IS THE BRANCH/DOOR'S NAME  ";BRANCH$
240 FOR M=0 TO 12 : READ M$(M) : NEXT : REM *  READ IN
MONTHS
250 DATA JAN, FEB, MAR, APR, MAY, JUN, JLY, AUG, SEP, O
CT, NOV, DEC, JAN
260 PRINT"WHICH SEASON WILL THIS PLAN COVER (FALL OR SP
RING)  ";
270 GOSUB 820    : REM *  INKEY SUBROUTE
280 IF KB$="S" THEN PRINT"SPRING" ELSE PRINT"FALL"
290 PRINT
300 IF KB$="S" THEN SB=1: SE=6: S=1: GOTO 330
310 IF KB$="F" THEN SB=7: SE=12: S=2: GOTO 330
320 IF KB$<>"S" OR KB$<>"F" THEN CLS: GOTO 260
330 PRINT"ENTER THE ";LYEAR;" SALES FOLLOWED BY A COMMA
, THEN THE ";YEAR;" SALES"
340 FOR T=SB TO SE
350 PRINT LYEAR;"/";YEAR;" SALES FOR  ";M$(T);
360 INPUT S(T), M(T)
370 ST=ST+S(T)    : REM *  TOTALS SALES THIS YEAR
380 TS=TS+M(T)    : REM *    TOTALS SALES LASY YEAR
390 NEXT
400 AV=( INT(TS-ST)/ST)*100   : REM *  PERCENT INCREASE
IN SALES
410 PRINT : INPUT"PLANNED SEASON TURN";TURN
420 PRINT"IF YOU HAVE AN ACTUAL BOM ENTER IT."
430 INPUT"IF YOU DO NOT  PRESS ENTER  ";BOM
440 AS=TS/6     : REM *  AVERAGE MO SALES=TOTAL SALES/SI
X MONTHS
450 AI=TS/TURN    : REM *  AVERAGE INVENTORY=TOTAL SALES
/PLANNED TURN
460 IF BOM<=0 THEN 470 ELSE 480
470 IF M(SB)>AS THEN BOM=(M(SB)-AS)+AI ELSE BOM=(AS-M(S
B))+AI
480 Z$(1)="          FEB     MAR      APR     MAY     J
UN     JLY    TOT"
490 Z$(2)="          AUG     SEP      OCT     NOV     D
EC     JAN    TOT"
500 N$=" EOM     ###.#   ###.#   ###.#   ###.#   ###.#
  ###.#   "
510 BY$=" OTR     ###.#   ###.#   ###.#   ###.#   ###.
#   ###.#"
520 O$=" BOM     ###.#"
530 R$=" ## SALES ###.#   ###.#   ###.#   ###.#   ###.#
  ###.#   ###.#"
540 L$=" ## SALES ###.#   ###.#   ###.#   ###.#   ###.#
  ###.#   ###.#"
550 Q$=" TURN ##.##      AVG INV ###.#     TOTAL SALES
###.#     +####%"
560 FOR T=SB TO SE-1    : REM *   COMPUTES END OF MONTH
INVENTORY
570 EOM(T)=(M(T+1)-AS)+AI
580 EOM=EOM+EOM(T)     : REM *   TOTALS EOM FOR SEASON
590 NEXT
600 CN=(AI*7)-(BOM+EOM)    : REM * COMPUTES ENDING PERIO
D INVENTORY
610 EOM(SE)=CN
620 YEAR=YEAR+1900
630 CLS : PRINT YEAR;" STOCK/SALES PLAN FOR   ";CHAIN$;
"   -     ";BRANCH$
640 YEAR=YEAR-1900
650 PRINT
660 PRINT Z$(S)
670 PRINT
680 PRINT USING L$; LYEAR, S(SB), S(SB+1), S(SB+2), S(S
B+3), S(SB+4),S(SB+5), ST
690 PRINT USING R$; YEAR, M(SB), M(SB+1), M(SB+2), M(SB
+3), M(SB+4), M(SB+5), TS
700 PRINT : PRINT USING O$; BOM
710 PRINT USING N$; EOM(SB), EOM(SB+1), EOM(SB+2), EOM(
SB+3), EOM(SB+4), EOM(SB+5)
720 PRINT : PRINT USING BY$; M(SB+1), M(SB+2), M(SB+3),
 M(SB+4), M(SB+5), M(SB)
730 PRINT
740 AI=(EOM+EOM(SE)+BOM)/7    : REM *  AVERAGE INVENTORY

750 PRINT USING Q$; TURN, AI, TS, AV
760 PRINT : GOSUB 840
770 PRINT" DO YOU WANT TO TO PRINT THIS (Y/N)"
780 GOSUB 820     : REM *  INKEY SUBROUTE
790 IF LEFT$(KB$, 1)="Y" THEN GOSUB 850: GOTO 620  : RE
M * IF YES LPRINTS
800 IF KB$="N" THEN 190     : REM *  IF NO RERUNS PROGRAM

810 IF KB$<>"N" OR KB$<>"Y" THEN 770
820 KB$=INKEY$ : IF KB$="" THEN 820 ELSE 830    : REM *
 INKEY ROUTINE
830 KB=VAL(KB$) : RETURN
840 POKE 16414, 88 : POKE 16415, 4 : RETURN  : REM *  L
PRINT TO PRINT
850 POKE 16414, 141 : POKE 16415, 5 : RETURN    : REM *
 PRINT TO LPRINT
```

****SPECIAL**SPECIAL****

TRS-80 ADD ON DRIVES

IMMEDIATE DELIVERY

SINGLE SIDED $225.00
DOUBLE SIDED $345.00

COMPLETE SYSTEMS
SINGLE SIDED $365.00
DOUBLE SIDED $485.00

INCLUDES:
- MINI DISK DRIVE
- FUSED POWER SUPPLY
- VENTED CABINET
- CABLE
- 90 DAY WARRANTY
- FACTORY ASSEMBLED
- FACTORY TESTED

THESE ARE NEW 5" FD's

I 2 INTERFACE, INC ✓246
20932 CANTARA ST
CANOGA PARK, CA 91304
(213) 341-7914

VISA AND MASTER CHARGE ACCEPTED

CHALLENGING NEW—EXCITING

AND NOW A SPECIAL GAME OFFER

WAR GAME PACKAGE

- BLACK KNIGHT—A TRS-80 Level II 16K Basic cassette game with sound. A super dungeon game in a maze complete with traps, treasures and monsters. You will never be bored with over a MILLION randon dungeons. **(KB-1) $7.95**
- ACTION PACKAGE—One TRS-80 Level II Basic cassette with TWO new games:
  1) BOMBER—Taking off, dodging obstacles, bombing targest, and landing safely, will require your skill in the pilot's seat.
  2) TORPEDO—You'll have hours of fun trying to torpedo a fleet of assorted deadly enemy ships. **(AP-1) $8.95**

"*Both games with graphics & sound.*"

FAST DELIVERY

(PA residents—add 6% tax)

Send your check or money order now to:

JMS Corp. Box 18083 ✓249
Pittsburgh, Pa. 15236

THE ALTERNATE SOURCE

The magazine of advanced applications and software for the TRS-80 ✓138

Presents:

TRAKCESS!

by: Roxton Baker

- *Most powerful disk utility yet!*
- *"Smart" DUPLICATE command analyzes every track! (2 drives)*
- *Super Pencil-type Editing!*
- *Read/Write/Edit/Create—any—track or sector, standard or not!*
- *A great tool for learning about disk structures—including the formerly little documented info!*
- *Fingertip control of all disk functions on 1-4 drives.*
- *Recover damaged disks. Oddball tracks and sectors don't faze it. Feed it TRSDOS, CP/M, protected disks, almost anything!*
- *Use it to create special disks interactively!*

An experiment!

An experiment to make this incredible program available to all. Add 1 cent for each copy (up to 5) you want to make and resell for your friends. We license you. The idea is to make TRAKCESS available to everyone who needs it. Appreciate this effort? Make it a success—Order now! **$24.95** *on Disk. Trakcess is over 23K of code. It requires 48K to run. Handles any number of tracks.*

MC/VISA/COD—Add $1.25

Phone: 517/485-0344

TAS, 1806 Ada, Lansing, MI 48910

MOVING?

Let us know 8 weeks in advance so that you won't miss a single issue of 80 Microcomputing.

Attach old label where indicated and print new address in space provided. Also include your mailing label whenever you write concerning your subscription. It helps us serve you promptly.

☐ Address change only ☐ Extend subscription ☐ Enter new subscription
☐ 1 year $18.00 ☐ Payment enclosed ☐ Bill me later

If you have no label handy, print OLD address here — *print NEW address here*

AFFIX LABEL

Name__________ Name__________
Address__________ Address__________
City_____ State___ Zip___ City_____ State___ Zip___

80 MICROCOMPUTING P.O. Box 981 • Farmingdale NY 11737

---

from ELCOMPCO

DUAL DISK DRIVE CASE KIT

✓63 for Shugart, MPI, & other minis

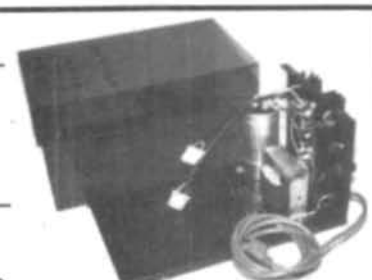
includes
- **CHASSIS**
- **PLEXIGLASS COVER**
- **ALL PARTS POWER SUPPLY**

(Disk Drives & Connecting Cables—extra)

Easy Assembly (1-2 hrs.)
Enough Power for 2 Mini Drives
use one now—one later

$125

Block off for Drive—$5 extra
DEALER INQUIRIES INVITED



COMPLETED UNIT

ELCOMPCO
Microcomputer Peripherals
Post Office Box 6133
Albany, CA 94706
C.O.D.

VISA

A return of serve from the bouncing dot.

Ping-Pong

Ronald Moehlis
3621 Clearview Dr.
Cedar Falls, IA 50613

This article contains five short programs to demonstrate the SET, RESET, POINT and INKEY$ features of the TRS-80 computer with Level II BASIC. The sequence of programs begins with a dot moving across the screen and each subsequent program adds additional instructions until you find yourself playing pong.

Program Listing 1 moves a dot horizontally across the screen. To plot a point with coordinates (X,Y) use SET(X,Y). The X value must range from 0 to 127 and the Y value from 0 to 47.

Statement 30 displays the dot before it is turned off by the RESET in statement 40. Statement 60 determines if the dot has reached the right edge of the screen and moves it back to the left edge.

Program Listing 2 adds two vertical "walls" in statement 6. When the dot reaches a wall it bounces in the opposite direction. Line 60 checks for this and line 68 changes the direction. Line 60 could be

```
IF POINT(X,25) THEN 68
```

which demonstrates the POINT feature. If the wall is directly ahead of the dot POINT(X,25) is TRUE; otherwise it is FALSE.

Program Listing 3 moves the dot diagonally and bounces it off walls in a rectangular box. Line 6 is modified and line 11 is added to provide the other walls. Since the dot now moves in two directions, the position variable Y and its increment variable K, is added to the program.

The paddles which can be moved to deflect the dot, are added in Program Listing 4. Also INKEY$ is added in statement 41 to allow real-time paddle action.

The O and L keys control the right paddle, while Q and A keys control the left. Statements 100 and 120 move the right paddle and statements 130 and 140 the left paddle.

Variable O is the Y value of the upper edge of the right paddle. L is the Y value of the lower edge. Q and A serve the same functions for the left paddle. Lines 55 and 63 determine if the dot has hit a paddle. If so, position variables X or Y change.

Program Listing 5 is the simple Pong game. Statements 53, 54, 60 and 61 determine if the dot is beyond the paddle. If so, a point is added to either score—variable RR or LL in lines 80 to 90. Now line 12 provides for a dot to appear at random from the top wall after a score.

Note that in the programs with the paddles, the movement of the dot is slower. This is because statements are executed to move the paddles.

More elaborate Pong programs can, of course, be written, but this set of programs demonstrates the idea to a beginner. ■

Note: Sound effects can be added to Pong using Babybeep by Dennis Kitsz. See *80 Microcomputing*, April, 1980.

```
1 REM PROGRAM 1
2 REM MOVEMENT OF A DOT
5 CLS
10 X=0
20 SET(X,25)
30 FOR I=1 TO 5:NEXT I
40 RESET(X,25)
50 X=X+1
60 IF X>127 THEN X=0
70 GOTO 20
```

Program Listing 1.

```
1 REM PROGRAM 2
2 REM DOT BOUNCING OFF WALLS
5 CLS
6 FOR Y=0 TO 47:SET(0,Y):SET(127,Y):NEXT Y
7 C=1
10 X=1
20 SET(X,25)
30 FOR I=1 TO 5:NEXT I
40 RESET(X,25)
50 X=X+C
60 IF X>126 OR X<1 THEN 68
65 GOTO 20
68 C=-C
70 GOTO 50
```

Program Listing 2.

```
1 REM PROGRAM 3
2 REM DOT BOUNCING IN A RECTANGULAR BOX
5 CLS
6 FOR Y=9 TO 39:SET(0,Y):SET(127,Y):NEXT Y
7 C=1
8 Y=10
9 K=1
11 FOR X=1 TO 126:SET(X,9):SET(X,39):NEXT X
12 X=1
20 SET(X,Y)
30 FOR I=1 TO 5:NEXT I
40 RESET(X,Y)
50 X=X+C
51 Y=Y+K
53 IF Y<39 AND Y>9 THEN 60
57 GOTO 66
60 IF X>126 OR X<1 THEN 68
65 GOTO 20
66 K=-K
67 GOTO 51
68 C=-C
70 GOTO 50
```

Program Listing 3.

```
1 REM PROGRAM 4
2 REM DOT BOUNCING IN A RECTANGULAR BOX WITH PADDLES
3 Q=20:A=25
4 L=25:O=20
5 CLS
6 FOR Y=9 TO 39:SET(0,Y):SET(127,Y):NEXT Y
7 C=1
8 Y=10
9 K=1
10 FOR J=20 TO 25:SET(7,J):SET(120,J):NEXT J
11 FOR X=1 TO 126:SET(X,9):SET(X,39):NEXT X
12 X=1
20 SET(X,Y)
30 FOR I=1 TO 5:NEXT I
40 RESET(X,Y)
41 A$=INKEY$
42 IF A$="" THEN 50
44 IF A$="O" THEN GOSUB 100
45 IF A$="L" THEN GOSUB 120
46 IF A$="Q" THEN GOSUB 130
47 IF A$="A" THEN GOSUB 140
50 X=X+C
51 Y=Y+K
53 IF Y<39 AND Y>9 THEN 60
55 IF  POINT(X,Y) THEN 66
60 IF X>126 OR X<1 THEN 68
63 IF  POINT(X,Y) THEN 68
65 GOTO 20
66 K=-K
67 GOTO 51
68 C=-C
70 GOTO 50
100 RESET(120,L):O=O-1:L=L-1:SET(120,O):RETURN
120 RESET(120,O):O=O+1:L=L+1:SET(120,L):RETURN
130 RESET(7,A):Q=Q-1:A=A-1:SET(7,Q):RETURN
140 RESET(7,Q):Q=Q+1:A=A+1:SET(7,A):RETURN
```

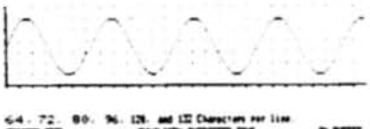
Program Listing 4.

```
1 REM PROGRAM 5
2 REM PONG
3 Q=20:A=25
4 L=25:O=20
5 CLS:LL=0:RR=0:PRINT@132,LL;:PRINT@186,RR
6 FOR Y=9 TO 39:SET(0,Y):SET(127,Y):NEXT Y
7 C=1
8 PRINT@158,"PONG";
9 K=1
10 FOR J=20 TO 25:SET(7,J):SET(120,J):NEXT J
11 FOR X=1 TO 126:SET(X,9):SET(X,39):NEXT X
12 Y=10:X=RND(100)+10
20 SET(X,Y)
30 FOR I=1 TO 5:NEXT I
40 RESET(X,Y)
41 A$=INKEY$
42 IF A$="" THEN 50
44 IF A$="O" THEN GOSUB 100
45 IF A$="L" THEN GOSUB 120
46 IF A$="Q" THEN GOSUB 130
47 IF A$="A" THEN GOSUB 140
50 X=X+C
51 Y=Y+K
52 IF Y<39 AND Y>9 THEN 60
53 IF X>122 THEN 90
54 IF X<6 THEN 80
55 IF  POINT(X,Y) THEN 66
60 IF X>122 THEN90
61 IF X<5 THEN 80
63 IF  POINT(X,Y) THEN 68
65 GOTO 20
66 K=-K
67 GOTO 51
68 C=-C
70 GOTO 50
80 RR=RR+1:IF RR>15 THEN 96
85 PRINT@186,RR;:GOTO12
90 LL=LL+1:IF LL>15 THEN 96
95 PRINT@132,LL;:GOTO12
96 PRINT@970,"":INPUT"PLAY AGAIN";B$
97 IF LEFT$(B$,1)="Y" THEN 5 ELSE END
100 RESET(120,L):O=O-1:L=L-1:SET(120,O):RETURN
120 RESET(120,O):O=O+1:L=L+1:SET(120,L):RETURN
130 RESET(7,A):Q=Q-1:A=A-1:SET(7,Q):RETURN
140 RESET(7,Q):Q=Q+1:A=A+1:SET(7,A):RETURN
```

Program Listing 5.

96 CHAR. PER LINE. ABCDEFGHIJKLMNOPQRSTUVWXYZ

BASE II for TRS-80
(The Graphic Line Printer)



64, 72, 80, 96, 120, and 132 Characters per line
TRACTOR FEED - AUXILIARY CHARACTER FONT - 2K BUFFER

Item	Expansion Interface With	Expansion Interface Without
Model [illegible]	[illegible]	[illegible]
Printer-Interface Cable	[illegible]	-
Modified RADIO SHACK Line Printer II Interface Cable	-	[illegible]
TOTAL	[illegible]	[illegible]
OUR PRICE	$710.00	$730.00

NOTE: THIS ENTIRE AD HAS BEEN CREATED ON THE BASE II LINE PRINTER.

AVS AUDIO/VIDEO SYSTEMS ✓148
2485 Autumnvale Dr.
San Jose, CA 95132

PH. 408-946-1265

64 CHAR. PER LINE. — 90 CHAR. PER LINE. — 132 CHAR. PER LINE.

We've got the FASTEST!!!

HIGH SPEED LIFE $9.95

GLIDERS & SPACESHIPS at 1800+ gen/min

J.H. Conway's famous Game of Life is animated !!!!

The most advanced Life program features: full screen 48 x 64 matrix • 32 preprogrammed patterns • special pattern generators • create, edit, and save your patterns on data tape • sound • single key functions - erase, save, single step, delay, scroll eight ways.

TRS-80* L2 16k REQUIRED

FOR CASSETTE AND INSTRUCTIONS SEND CHECK OR MONEY ORDER TO:

Synergistic Solar, Inc.
PO Box 560595, Miami, FL 33156

SPEED TABLE 1.77MHz	
POP. SIZE	[illegible]
5	2290
10	2070
50	1110
100	860
500	310
1000	190

For more information about our programs please mail us a postcard. *TRS-80 is a trademark of Tandy Corp.

---

accounts receivable

This program was developed for our family business. The goal was an orderly transition from hand-kept books to computerized bookkeeping. Accounts receivable including sales tax was the most time consuming, thus the beginning. Time spent on accounts receivable has been cut in half. More detailed statements sent earlier have decreased the outstanding receivables.

This program is designed for business people, not computer or accounting experts. All inputs are structured for fast and efficient operation. If a math check is required, extensions and totals are checked faster than with a calculator. If a math check is not required, only the amount of sale, less any tax, is entered. The program stores sales tax information on all customers and will display the proper tax, if any is required. Non-taxable customers making a taxable purchase, customers from a tax district other than the place of business, picking up at the place of business are easily handled by the program. Amount of tax and taxable sales are compiled and stored for up to 100 sales tax districts.

Every entry can be edited at any time. If an incorrect or missing transaction is found after statements are printed, the corrections can be made and the statement reprinted. Sales tax and monthly data in disk files are adjusted by this process.

Rate of service charge and customers to receive service charge are selected by the user.

Overview and Sample Reports.......$5.00
Documentation.......[illegible]
Program Disk & Documentation.......[illegible]

solutions 80
✓277 Route 55
La Grangeville, New York 12540

9.95 SOFTWARE

P.O. BOX 521 Lowell, MA 01853

1 • Memory based printer spooler, overlap processing and I/O.

2 • IBM based terminal driver, EBCD and correspondence code. Full ASCII character set with overstrikes. Scripsit ZAPS.

3 • Disk timing program. Meter Type numerical and statistical screen displays. Very easy calibration of all type drives.

4 • Cassette test programs. Writes test data to tape, then displays all errors on screen. Use to check all facets of cassette operation and duplication quality.

SEND FOR FREE FLYER

— The bottom line —
COST: $9.95 Each + .75 postage
MA Orders + 5% tax ✓235

TRS-80*

SAVE A BUNDLE

When you buy your TRS-80™ equipment!

Use our toll free number to check our price before you buy a TRS-80™ . . . anywhere!

TRS-80 is a trademark of the Radio Shack Division of Tandy Corporation

full Radio Shack warranty

TAB ✓148
SALES COMPANY
1412 WEST FAIRFIELD DR.
P.O. BOX 8098 PENSACOLA FL 32505
904/438-6507
nationwide 1-800-874-1551

THE WAIT IS OVER

Now for the first time, Cpaids, is making available its **Federal 1040 Tax** package for the TRS-80 Model II.

Featuring-1% and 3% medical limitation incorporated, auto checks for FICA over withholding, auto earned income credit, auto dividend exclusions, types tax preparer's Fed. ID and SSN; proformas all forms for subsequent years.

Schedules - A, B, C, D, E, F, G, R/RP, SE, TC, ES

Forms - 2106, 2119, 2210, 2441, 3468, 3903, 4625, 4972, 4726, 4797, 5695, 6521

Requires 64 K with dual disks using CP/M.

CALL TOLL FREE
800-321-2430

✓388

CPAIDS.

1640 Franklin Ave., Kent, Ohio 44240

PPI-80

PARALLEL I/O FOR THE TRS-80

The PPI-80 is a complete parallel I/O interface designed specifically for the TRS-80, consisting of 3 complete 8 bit I/O ports including such features as:

- switch selectable address decoding
- complete on board regulated power supply
- TTL compatible I/O lines conveniently available through 16 pin sockets
- +5 volts and ground at each socket
- 3 software selectable modes of operation
- handshaking
- plugs into keyboard or expansion interface
- on board kluge area for experimenting
- provisions for interfacing Sears-BSR-RS home controller

Possible applications include:

- bidirectional communication between microcomputers
- parallel printer interface
- wireless home control via BSR home controller
- direct control of lights, appliances, and motors
- interfaces to many popular boards including A/D-D/A converter and an EPROM Programmer

PPI-80 is available now and can be purchased in several forms

Completely assembled and tested.......$119.95
Complete kit with all parts.......89.95
Bare board drilled and etched with assembly manual.......25.95
Accessories
8 channel A/D - 2 channel D/A by Optimal Technology.......$115.00
EPROM Programmer Model EP-2A-79 by Optimal Technology.......155.00

To order, send payment plus $2.00 shipping and handling to:
QUANT SYSTEMS
P.O. BOX 628
CHARLESTON, S.C. 29402

S.C. residents add 4% sales tax
Overseas orders add $5 for shipping

✓289
Quant Systems
p.o. box 628
charleston sc
29402

80 microcomputing bookshelf

2 new books from the editors of KB & 80 Microcomputing

●**40 COMPUTER GAMES**—BK7381—At last! Here is an unbeatable game collection from the folks at your favorite computer magazine. Forty games in nine categories including; gambling, board games, space games, puzzles, card games and calculator games. These are the best game programs that have recently appeared in Kilobaud/Microcomputing magazine, and truly offer something for everyone.

Many versions of BASIC are represented including: North Star, Radio Shack, MITS and Tiny. Systems used are as diverse as Altair, H8, Tandy and KIM. Even the calculator set has something to cheer about with games for the HP-67, TI-59 and SR56.

A 'must have' edition for the serious computer gamesman. $7.95*

●**UNDERSTANDING AND PROGRAMMING MICROCOMPUTERS**—BK7382—This easy to follow text brings you the work of several, well known authors in the computer field. Understanding and Programming is divided into two parts. The first covers microcomputers and their associated software starting with the fundamentals of system architecture and ending with an introduction to programming languages. Section two addresses the thinking processes essential to good programming and gives detailed examples of well written software. The articles were selected from the 73 and Kilobaud Microcomputing library of fine articles and represent the best that has been written on the topic in recent years. A valuable addition to your computing library. $10.95*

Other Books

●**INSIDE LEVEL II**—BK1183—For machine language programmers! This is a comprehensive reference guide to the Level II ROMs, allowing easy utilization of the sophisticated routines they contain. It concisely explains set-ups, calling sequences, variable passage and I/O routines. Part II presents an entirely new composite program structure which unloads under the SYSTEM command and executes in both BASIC and machine code with the speed and efficiency of a compiler. Special consideration is given to disk systems. $15.95.*

NEW!

●**TRS-80 DISK AND OTHER MYSTERIES**—BK1181—by Harvard C. Pennington. This is the definitive work on the TRS-80 disk system. It is full of detailed "How to" information with examples, samples and in-depth explanations suitable for beginners and professionals alike. The recovery of one lost file is worth the price alone. $22.50.*

●**PROGRAMMING THE Z-80**—BK1122—by Rodnay Zaks. Here is assembly language programming for the Z-80 presented as a progressive, step-by-step course. This book is both an educational text *and* a self-contained reference book, useful to both the beginning and the experienced programmer who wish to learn about the Z-80. Exercises to test the reader are included. $14.95.*

●**LEARNING LEVEL II**—BK1175—by David Lien. Written especially for the TRS-80, this book concentrates on Level II BASIC, exploring every important BASIC language capability. Updates are included for those who have studied the Level I User's Manual. Sections include: how to use the Editor, dual cassette operation, printers and peripheral devices, and the conversion of Level I programs to Level II. $15.95.*

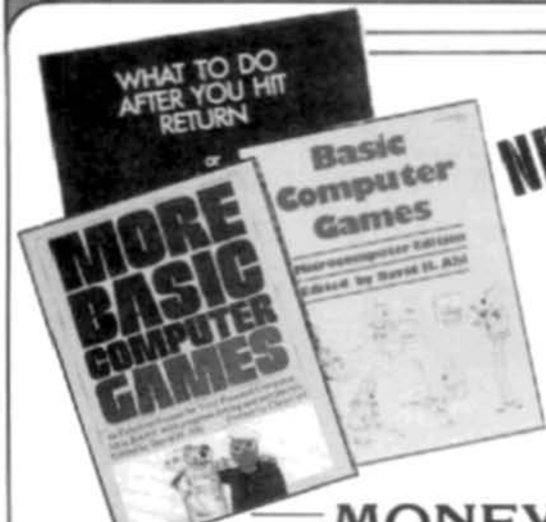
●**THE BASIC HANDBOOK**—BK1174—by David Lien. This book is unique. It is a virtual ENCYCLOPEDIA of BASIC. While not favoring one computer over another, it explains over 250 BASIC words, how to use them and alternate strategies. If a computer does not possess the capabilities of a needed or specified word, there are often ways to accomplish the same function by using another word or combination of words. That's where the HANDBOOK comes in. It helps you get the most from your computer, be it a "bottom-of-the-line" micro or an oversized monster. $14.95.*

*Use the order card in the back of this magazine or itemize your order on a separate piece of paper and mail to 80 Microcomputing Bookshelf ● Peterborough NH 03458. Be sure to include check or detailed credit card information. No C.O.D. orders accepted. All above add $1.00 handling. Please allow 4–6 weeks for delivery. Questions regarding your order? Please write Customer Service at the above address.

FOR TOLL FREE ORDERING CALL 1-800-258-5473

80 microcomputing bookshelf

GAMES



NEW!

• MORE BASIC COMPUTER GAMES – BK1182 – edited by David H. Ahl. More fun in BASIC! 84 new games from the people who brought you *BASIC Computer Games*. Includes such favorites as Minotaur (battle the mythical beast) and Eliza (unload your troubles on the doctor at bargain rates). Complete with game description, listing and sample run. $7.50.*

• WHAT TO DO AFTER YOU HIT RETURN – BK1071 – PCC's first book of computer games . . . 48 different computer games you can play in BASIC . . . programs, descriptions, many illustrations. Lunar Landing, Hammurabi, King, Civel 2, Qubic 5, Taxman, Star Trek, Crash, Market, etc. $10.95.*

• BASIC COMPUTER GAMES – BK1074 – Okay, so once you get your computer and are running in BASIC, then what? Then you need some programs in BASIC, that's what. This book has 101 games for you from very simple to real buggers. You get the games, a description of the games, the listing to put in your computer and a sample run to show you how they work. Fun. Any one game will be worth more than the price of the book for the fun you and your family will have with it. $7.50.*

MONEY-MAKING

• HOW TO MAKE MONEY WITH COMPUTERS – BK1003 – In 10 information-packed chapters, Jerry Felsen describes more than 30 computer-related, money-making, high profit, low capital investment opportunities. $15.00.*

• HOW TO SELL ANYTHING TO ANYBODY – BK7306 – According to *The Guinness Book of World Records*, the author, Joe Girard, is "the world's greatest salesman." This book reveals how he made a fortune – and how you can, too. $2.25.*

• FREELANCE SOFTWARE PUBLISHING – BK1179 – by B. J. Korites. "This book is about money and how to make it by writing and selling computer programs," (author's foreword). If you have the skills to write a saleable program, you now need to acquire the skills to sell that program. This compact book comprehensively covers the entire publishing process and many aspects of software salesmanship. $14.95.*

• THE INCREDIBLE SECRET MONEY MACHINE – BK1178 – by Don Lancaster. A different kind of "cookbook" from Don Lancaster. Want to slash taxes? Get free vacations? Win at investments? Make money from something that you *like* to do? You'll find this book essential to give you the key insider details of what is really involved in starting up your own money machine. $5.95.*

BUSINESS

• PAYROLL WITH COST ACCOUNTING – IN BASIC – BK1001 – by L. Poole & M. Borchers, includes program listings with remarks, descriptions, discussions of the principle behind each program, file layouts, and a complete user's manual with step-by-step instructions, flowcharts, and simple reports and CRT displays. Payroll and cost accounting features include separate payrolls for up to 10 companies, time-tested interactive data entry, easy correction of data entry errors, job costing (labor of distribution), check printing with full deduction and pay detail, and 16 different printed reports, including W-2 and 941 (in **CBASIC**). $20.00.*

• SOME COMMON BASIC PROGRAMS — BK1053 — published by Adam Osborne & Associates, Inc. Perfect for non-technical computerists requiring ready-to-use programs. Business programs, plus miscellaneous programs. Invaluable for the user who is not an experienced programmer. All will operate in the stand-alone mode. $12.50 paperback.*

• PIMS: PERSONAL INFORMATION MANAGEMENT SYSTEM – BK1009 – Learn how to unleash the power of a personal computer for your own benefit in this ready-to-use data-base management program. $11.95.*

*Use the order card in the back of this magazine or itemize your order on a separate piece of paper and mail to 80 Microcomputing Bookshelf • Peterborough NH 03458. Be sure to include check or detailed credit card information. No C.O.D. orders accepted. All above add $1.00 handling. Please allow 4–6 weeks for delivery. Questions regarding your order? Please write Customer Service at the above address.

FOR TOLL FREE ORDERING CALL 1-800-258-5473

HOBBYWORLD®

ELECTRONICS, INC.

19511 BUSINESS CTR. DR. DEPT V9
NORTHRIDGE, CALIFORNIA 91324

Send For Your Free Flyer Today!

74LS

Order by Cat No. 999 and Type

74LS00	$ .33	74LS74	$ .99	74LS161	$1.75	74LS257	$ .99
74LS01	$ .29	74LS75	$ .59	74LS162	$1.19	74LS258	$ .66
74LS02	$ .49	74LS76	$ .44	74LS163	$1.15	74LS259	$1.99
74LS03	$ .29	74LS83	$ .88	74LS164	$1.15	74LS260	$ .66
74LS04	$ .49	74LS85	$1.15	74LS165	$1.69	74LS261	$2.50
74LS05	$ .35	74LS86	$ .99	74LS166	$3.95	74LS266	$ .66
74LS08	$ .44	74LS90	$ .59	74LS168	$1.29	74LS273	$3.25
74LS09	$ .29	74LS92	$ .75	74LS169	$3.33	74LS275	$4.95
74LS10	$ .44	74LS93	$ .75	74LS170	$2.25	74LS279	$ .49
74LS11	$ .29	74LS95	$ .88	74LS173	$1.25	74LS283	$1.75
74LS12	$ .29	74LS107	$ .55	74LS174	$1.10	74LS293	$1.99
74LS13	$ .55	74LS109	$ .55	74LS175	$ .99	74LS295	$1.99
74LS14	$1.10	74LS112	$ .55	74LS181	$2.50	74LS298	$1.10
74LS15	$ .35	74LS113	$ .55	74LS190	$ .69	74LS324	$1.75
74LS20	$ .29	74LS114	$ .55	74LS191	$1.15	74LS365	$ .99
74LS21	$ .39	74LS122	$ .55	74LS192	$ .99	74LS366	$ .89
74LS22	$ .29	74LS123	$1.15	74LS193	$1.15	74LS367	$ .99
74LS26	$ .77	74LS124	$1.55	74LS194	$1.15	74LS368	$ .99
74LS27	$ .55	74LS125	$ .88	74LS195	$1.15	74LS373	$3.25
74LS28	$ .44	74LS126	$ .88	74LS196	$ .99	74LS374	$4.50
74LS30	$ .39	74LS132	$ .88	74LS197	$1.99	74LS377	$3.25
74LS32	$ .66	74LS138	$1.25	74LS221	$1.99	74LS378	$1.69
74LS33	$ .69	74LS139	$ .99	74LS240	$3.95	74LS386	$ .77
74LS37	$ .39	74LS145	$1.99	74LS241	$2.95	74LS393	$2.25
74LS38	$ .59	74LS147	$2.95	74LS242	$1.95	74LS395	$1.99
74LS40	$ .33	74LS151	$ .88	74LS243	$2.95	74LS399	$2.95
74LS42	$ .88	74LS153	$ .88	74LS244	$3.25	74LS424	$3.25
74LS47	$ .88	74LS154	$3.50	74LS245	$5.00	74LS670	$2.25
74LS48	$ .88	74LS155	$ .99	74LS247	$ .88	81LS95	$1.95
74LS51	$ .39	74LS156	$1.25	74LS248	$ .99	81LS96	$1.95
74LS54	$ .29	74LS157	$1.25	74LS251	$1.99	81LS97	$1.95
74LS55	$ .55	74LS158	$1.49	74LS253	$ .99	81LS98	$1.95
74LS73	$ .44	74LS160	$ .99	74LS256	$2.25		

TTL's

Order by Cat No. 999 and Type

7400	$.35	7445	$ .77	74109	$ .55	74176	$ .79
7401	$.35	7446	$ .69	74116	$1.89	74177	$ .77
7402	$.35	7447	$ .66	74120	$ .99	74179	$1.88
7404	$.44	7448	$ .77	74121	$ .44	74180	$ .77
7405	$.44	7450	$ .20	74122	$ .50	74181	$1.88
7406	$.39	7451	$ .50	74123	$ .52	74182	$1.99
7407	$.39	7453	$ .50	74125	$ .52	74184	$1.99
7408	$.35	7454	$ .20	74126	$ .49	74185	$1.99
7409	$.35	7460	$ .29	74132	$ .69	74190	$1.19
7410	$.35	7470	$ .29	74141	$ .77	74191	$1.19
7411	$.39	7472	$ .29	74143	$3.33	74192	$ .77
7412	$.49	7473	$ .36	74145	$7.77	74193	$ .89
7413	$.44	7474	$ .49	74148	$1.29	74195	$ .69
7414	$.66	7475	$ .49	74150	$ .88	74196	$ .88
7416	$.45	7476	$ .38	74151	$ .59	74197	$ .88
7417	$.29	7479	$3.99	74153	$ .69	74198	$1.49
7420	$.35	7480	$ .50	74155	$ .49	74199	$1.49
7422	$.44	7481	$ .99	74156	$ .99	74221	$1.99
7423	$.44	7483	$ .59	74157	$ .63	74251	$ .77
7425	$.38	7485	$ .85	74160	$ .77	74273	$1.10
7426	$.39	7486	$ .35	74161	$ .79	74278	$2.95
7427	$.35	7489	$1.66	74162	$ .79	74279	$ .82
7430	$.35	7490	$ .44	74163	$ .88	74365	$ .69
7432	$.39	7491	$ .59	74164	$ .88	74366	$ .69
7437	$.39	7492	$ .45	74165	$ .88	74367	$ .69
8438	$.39	7493	$ .45	74166	$1.29	74368	$ .69
7440	$.20	7495	$ .65	74170	$1.59	74393	$2.50
7441	$.77	7496	$ .65	74173	$1.09	8T26	$2.50
7442	$.49	74100	$1.69	74174	$ .79	8T97	$2.25
7433	$.69	74107	$ .44	74175	$ .79		

ATARI HOME VIDEO SYSTEM

$183.00

The nation's best selling home video entertainment center is here! Currently supports a library of 32 game cartridges with over 1500 game variations and options. Now you can enjoy all the fun and excitement of an arcade in your own home whenever you wish. Terrific for party entertainment, developing coordination and dexterity, education, or just plain family fun. Comes with interchangeable joystick and paddle controllers, special circuits to protect home T.V., and ATARI'S realistic "combat" game with 108 variations and options. ATARI'S realistic sound effects and crisp, bright colors make the home video center your number one entertainment choice.

CAT NO.	DESCRIPTION	WT	PRICE
2375	ATARI Home Video System	8 lb.	183.00
2206	Driving Controller-Pair	2 lb.	19.95
2207	Paddle Controller-Pair	2 lb.	19.95
2208	Joystick Controller Pair	2 lb.	19.95

ATARI GAME CARTRIDGES

2376	Starship	6 oz.	19.50
2377	Black Jack	6 oz.	18.75
2378	Space War	6 oz.	12.95
2379	Surround	6 oz.	12.95
2380	Slot Machine	6 oz.	18.75
2381	Outlaw	6 oz.	19.50
2382	Slot Racer	6 oz.	19.50
2383	Video Olympics	6 oz.	19.50
2384	Breakout	6 oz.	19.50
2385	Canyon Bomber	6 oz.	19.50
2386	Street Racer	6 oz.	19.50
2387	Homerun	6 oz.	19.50
2388	Basketball	6 oz.	19.50
2389	Football	6 oz.	19.50
2390	Bowling	6 oz.	19.50
2391	Skydiver	6 oz.	19.50
2392	Fun with Numbers	6 oz.	18.75
2393	Brain Game	6 oz.	19.50
2394	Superman	6 oz.	25.00
2395	Casino	6 oz.	25.00
2396	Backgammon	6 oz.	37.95
2397	Video Chess	6 oz.	37.95
2500	Space Invaders	6 oz.	25.00
2609	Adventure	6 oz.	25.00
2611	Indy 500	6 oz.	37.50
2634	Human Cannon Ball	6 oz.	19.50
2635	Codebreaker	6 oz.	19.50
2636	Flag Capture	6 oz.	19.50
2637	Air-Sea Battle	6 oz.	19.50
2638	Hunt and Score	6 oz.	19.50
2639	Miniature Golf	6 oz.	19.50
2640	Hangman	6 oz.	18.75
2653	Tic Tac Toe	6 oz.	19.50
2654	Circus	6 oz.	19.50

Note: Not for use with ATARI Programmable Computers

ANADEX
DP9500/DP9501 PRINTERS

$1650.00

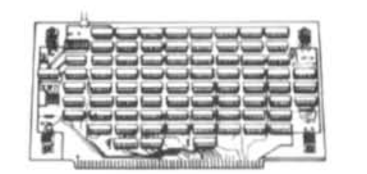
New from Anadex! Two low cost, high performance printers designed for all applications, including standard high-density graphics capability. Both models feature a 9 wire print head with an incredible life expectancy of 650 million printed characters! Full 96 character ASCII set with lower case descenders, double width printing, bi-directional with shortest distance sensing logic. Adjustable-width tractor feed, forms control, horizontal and vertical tabbing, and print up to five copies. Easy interfacing with parallel, RS-232 serial or current loop choices.

The DP9500 is the choice when you require mostly printing and occasional graphics. Select between a 9 x 9 character font and 132 columns, or a 7 x 9 font for 175 columns. Printer speed: 150/200 CPS. Wt. 35 lbs.

The DP9501 is mainly for graphics applications. The 11 x 9 character font produces superb graphics reproduction in 132 columns, and the 7 x 9 character font in 220 columns provides maximum graphics potential. Both models operate at 110VAC and 220 VAC for European use. Wt. 35 lbs.

Cat No. 2551 DP9500 printer
Cat No. 2552 DP9501 printer

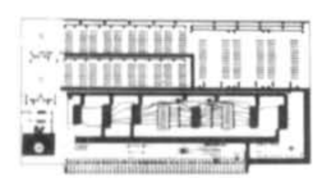
SSM MB6B
8K STATIC RAM BOARD **$135**



8K bytes by 8 bits, fully buffered, compatible with 8080, 8085, and Z80. Dip switch addressing of independant 4K halves lets the MB6B think like two 4K boards, or one 8K board. Independent 4K addressing allows the flexibility to meet varying software memory needs. Uses low power 21L02 RAM's operates at 2 or 4MHZ, and is compatible with direct memory access controllers.

Cat No.	Description	Price
*1400-A	450ns kit	$135.00
*1400-B	250ns kit	$147.50
1401-A	450ns a & t	$209.00
1401-B	250ns a & t	$225.00
*1402	Bareboard	$ 23.75

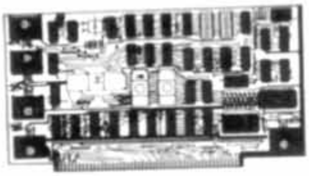
SSM OB1
VECTOR JUMP & PROTOTYPING CARD
$41.25*



Plug compatible for S-100 bus systems, features full 16 bit vector jump address with dip selection of 8080 or Z80. Can be set to jump on power-on clear, reset, or both. Prototyping areas on the card for ten 16-pin IC's three 24-28 pin IC's and two spare regulator patterns.

*Cat No. 1429 OB1 kit	$41.25
Cat No. 1430 OB1 a & t	$85.00
Cat No. 1431 OB1 bareboard	$32.00

Call Toll-Free: USA (800) 423-5387
In California: (800) 382-3651
Local & Outside USA: (213) 886-9200



SSM CB1-A

8080 CPU BOARD

$159 kit

Just add an I/O board and it's a computer! 256 bytes of on board RAM, with option for 2K of on board PROM. Includes a power-on, preset jump circuit, and MWRITE is available, allowing use without a front panel. There's a parallel input port with status, and AIP controlled addressing, or PROM in 2K blocks, vector jump in 2K increments, RAM in 256 byte increments, RAM in 256 byte increments, input port for addresses 0 – 31 in decimal

*Cat No. 1403	CB1-A kit	$159.00
*Cat No. 1441	CB1-A bareboard	$28.75

* Denotes excess inventory sale. No further discounts shall apply

HAYDEN **SARGON II**

- *Available in Cassette or Diskette*

Sargon II has seven levels of play! When setting up the board the user can scan up and down, left and right before choosing a move. The computer displays the level in which it is thinking, shows the moves that it is contemplating, and then displays the move it has chosen. It comes with a randomized book of opening moves for all 7 levels of play, and a special hint mode that will suggest moves for players who may need some help.

Cat No.	Description	Weight	Price
2082	TRS-80, L2 16K Cassette	6 oz.	$29.95
2083	Apple II, 24K Cassette	6 oz.	$29.95
2352	TRS-80, L2 16K Diskette	6 oz.	$35.95
2342	Apple II, 24K Diskette	6 oz.	$35.95

INVENTORY 'S'

Allows the inclusion of alphabetic information and a data index code in the form of data statements within the program. Includes: **1) Reports;** User specifies up to three numeric and either or both alpha informations to be listed. Can be vendor specific. **2) Cost/value summary;** Searches all stock areas and reports cost/value quantity, total value by line item, and grand total. **3) Recorder search;** Compares current stock level against specified reorder point and displays all line items in need of reorder, along with tentative reorder information. **4) Index;** Uses arbitrary file numbers reflecting the order in which the data codes are stored. Reveals file names and numbers in groups of 24. **5) Detailed Rept;** Stock files can be called by file number to reveal memory information. **6) Read and write file;** Stores and reenters data from day to day. **7) Data change;** Updates Data Base.

Cat No. 2058 TRS-80 L2, 16K, cassette **$24.95**

MICROSOFT

TRS-80 FORTRAN PLUS

Create your own subroutines, assembler source files, data files, and FORTRAN files. Fully compatible with TRSDOS, the FORTRAN compiler can operate 1200 lines per minute on a single pass. Also generates a fully symbolic listing of the machine language generated. Macro assembler accepts Z80 opcodes and supports complete Intel standard macro facility. Linking loader provides a variety of capabilities, executed by means of easy command lines and switches. Any number of programs may be loaded with just one command. Text editor random access, line oriented. Only the library routines required to run a particular FORTRAN program need be loaded before execution. Non-standard I/O drivers for each Logic Unit Number may be written, simplifying the task of interfacing non-standard devices to TRS-80 FORTRAN programs.

Cat No. 1341 TRS-80 L2, 32K w/disk **$185**

PROGRAMMA

TIELINE

$24.95

The most complete communications for the TRS-80, featuring: Host computer: Turns your TRS-80 into a timeshare mainframe. Smart Terminal: Operate another TRS-80 or mainframe timeshare system from your TRS-80. Send/recieve BASIC programs and data. Allows exchange of programs over phone lines (modem required). Operate a serial printer, test your modem and RS232, and much more! Complete with extensive manual on telephone communications.

Cat No. 2137 TRS-80 L2, 32K modem, DOS 2.1, RS232 cassette, used w/disk

Introduction to TRS-80 GRAPHICS

Bob Albrect and Don Inman

One of the most outstanding capabilities of the small computer is the ability to provide graphical data displays that formerly cost thousands of dollars. This book provides a basic introduction to graphic programming using dozens of real examples which run on the Radio Shack TRS-80 Computer. The book begins with basic concept line drawing, then leads the reader to more complexed geometric shapes, moving figure animation, and other more advanced topics. No mathematics is required, but an understanding of BASIC language is assumed. A TRS-80 Computer for running examples is recommended, but concepts will apply to most low cost computers with graphics capabilities. (175 pages)

Cat No. 2544 **$8.95**

PROGRAMMA

DATA BASE MANAGEMENT 5

$49.95

An easy method of creating data files and storing them in disk memory for future use. Allows you to store and manipulate data for maximum productivity, and modify or incorporate your own routines. Uses Radio Shack's TRSDOS/BASIC language.

Cat No. 2146 TRS-80 L2, 16K

TRS-80 ELECTRIC PENCIL

Allows you to produce mailing lists, forms, large numbers of original correspondence, etc. A character-oriented word processing system, providing maximum freedom and simplicity in the handling of text. Eliminates the need for word hyphenations or carriage returns. Line formatting is done automatically. Insert, delete, or relocate any text using simple keyboard commands.

Cat No. 1338	TRS-80, L1 & L2, 16K, cassette	6 oz.	$ 95.00
Cat No. 1338-D	TRS-80, L1 & L2, 16K, disk	6 oz.	$145.00

TRS-80 CP/M

$149.95

A file-oriented disk operating system that provides a common set of utilities for program development and operation. There are six built-in commands, plus utilities called in from disk. Runs on as little as 16K of memory and one disk drive. Complete with six manuals. CP/M is a registered trademark of Digital research.

Cat No. 1679 TRS-80 L2, 16K w/disk

VERBATIM 525 SERIES

5¼" Diskettes

- *Double Density*
- *Single sided*
- *Perfect for commercial and general applications*

Cat No.	Description	Type	Use for	10 for
1147	Soft sector	525-01	TRS-80	$33.00
1148	10 hole, hard	525-10	North Star, Apple	$33.00
1149	16 hole, hard	525-16	Micropolis	$33.00

VERBATIM 577 SERIES

10 PER BOX

- *Certified twice, 77 tracks*
- *Single sided, double sided*
- *Built-in hub protector ring*
- *For critical data applications*

2330	Soft sector	577-01	TRS-80, etc	$49.95
2331	10 hole, hard	577-10	North Star, Apple	$54.95
2332	16 hole, hard	577-16	Micropolis, etc	$49.95

DISK/DISKETTE DRIVE HEAD CLEANING KIT

"Diskette drive heads require periodic maintenance to assure efficient and error-free operation. Unlike other peripheral devices, the read/write head(s) on disk drives are extremely difficult to clean without partially disassembling the unit. But now, with Hobbyworld's disk drive head cleaning kit, the user can clean these hard-to-reach heads in just minutes!" Available for both 5¼ and 8" drives, single and double sided. Comes complete with two cleaning disks, 4 oz of CS 85 cleaning solution, and easy pour dispenser. Wt. 12 oz.

Cat No. 2499 8" disk **$30**
Cat No. 2534 5¼" diskette

TRS-80 and APPLE

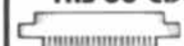
16K MEMORY ADD-ON

$55.00

Everything you need to upgrade your system! Includes 4 pages of illustrated instructions. Complete with RAM's and pre-programmed jumpers. No special tools required! Wt. 4 oz.

Cat No	Description
1156	For TRS-80 Keyboard Unit
1156A	For TRS-80 Exp. Interface purchased before 4/1/79
1156B	For TRS-80 Exp. Interface purchased after 4/1/79
1156C	Apple II
1156D	EXIDY

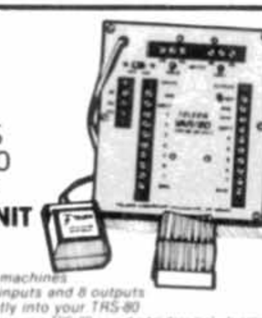
TRS-80 EDGE CONNECTOR



- *40 contacts (2 x 20)*
- *.01" spacing*

For expansion or main unit. Wt. 4 oz. **$4.75**

Cat No. 1980 **2 for $9**



TELESIS VAR/80

TRS-80 I/O UNIT

$105

- *For level II machines*
- *Provides 8 inputs and 8 outputs*
- *Plugs directly into your TRS-80*

Now you can use your TRS-80 as a digital door lock, burglar alarm, power manager, frequency counter, light dimmer, dark room timer, just to name a few! Comes fully assembled and tested. Use it with or without the expansion interface! Data pack includes instructions, applications, sample circuits, and several programs. With power supply. Wt. 5 lbs.

Cat No. 1092

SEND FOR FREE FLYER FEATURING . . .

Page after page of impressive, state of the art products. Included in flyer are computers, terminals, disk-drives, printers and many more electronics peripherals that can add dimension to your personal computing. We also stock computerized games and toys, application boards, a large selection of comprehensive software, electronics books, electronics parts, integrated circuits, P.C. & soldering accessories and much, much more.

For all your electronics' or hobbyist's needs, circle our Reader Service number or write/phone us for your free illustrated flyer today!

HOW TO ORDER

Minimum Order $15.00. Order by phone, mail, or at our retail stores. Pay by check, Mastercharge, Visa or C.O.D. Please include expiration date with charge card orders. U.S. $$ only. Include phone number and magazine issue you are ordering from. Add $1.25 for C.O.D. and shipping charges from rates below.

Shipping Rates: U.S.A.
Ground: Add $2.25 for first 2 lbs. and 40¢ each add'l. lb.
Air: Add $3.25 for first 2 lbs. and 70¢ each add'l. lb.

Shipping Rates: Foreign
Ground: Add $3.00 for first 2 lbs. and 60¢ each add'l. lb.
Air: Add $11.25 for first 2 lbs. and $5.00 each add'l. lb.

Prices Valid through month of issue. Not responsible for typographical errors. Some items subject to prior sale or quantity limits. 120 day guaranteed satisfaction.
Exception: Partially assembled kits, abuse or misuse.

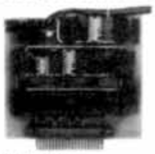
HOBBYWORLD® ELECTRONICS INC.

19511 BUSINESS CTR. DR. DEPT V9 NORTHRIDGE, CALIF 91324

✓Reader Service—see page 226

TRS-80 SERIAL I/O

• Can input into basic • Can use LLIST and LPRINT to output, or output continuously • RS-232 compatible • Can be used with or without the expansion bus • On board switch selectable baud rates of 110, 150, 300, 600, 1200, 2400, parity or no parity odd or even, 5 to 8 data bits, and 1 or 2 stop bits. D.T.R. line • Requires +5, -12 VDC • Board only $19.95 Part No. 8010, with parts $59.95 Part No. 8010A, assembled $79.95 Part No. 8010 C. No connectors provided, see below.



EIA/RS-232 connector Part No DB25P $6.00, with 9', 8 conductor cable $10.95 Part No. DB25P9

3' ribbon cable with attached connectors to fit TRS-80 and our serial board $19.95 Part No. 3CAB40

COMPUCRUISE

$129.95; with cruise control $169.95

PAPER TIGER

Prints address labels, multicopy invoices and legal-size reports. Adjust the tractor width from 1-3/4 to 9-1/2 inches. 8 switch-selectable forms lengths. Print 6 or 8 lines per inch. Add the software-selectable full dot plotting graphics option to print illustrations, block letters, charts, graphs. Part No. 162172 $899.95 • with graphics option Part No. 162173 $1099.95

GAME PADDLES & SOUND

Includes: 2 game paddles, interface, software, speaker, power supply, full documentation including: schematics, theory of operation, and user guide; plus 2 games on cassette (Pong and Starship War). $79.95 Complete Part No. 7922C

DIGICOM DATA PRODUCTS INC. Series 312 Acoustic Coupler

300 BAUD Originate, Part No. AC3122, $219.95. 300 BAUD Answer, Part No. AC3122, $219.95. 300 BAUD Answer/Originate, Part No. AC3123, $229.95.

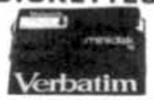
IBEX LIGHT PEN

Comes with Backgammon and Tic-Tac-Toe on tape with full documentation and program listing. Requires 9v. battery. Part No. IBEX $19.95

SYSTEM EXPANSION from LNW Research

• Serial RS232C/20 mA I/O • Floppy controller • 32K bytes memory • Parallel printer port • Dual cassette port • Real-time clock • Screen printer bus • Onboard power supply • Software compatible • Solder mask, silk screen, PC board and user manual, Part No. LNW80, $69.95.

DISKETTES



Box of 10, 5" $29.95, 8" $39.95. Plastic box, holds 10 diskettes, 5" - $4.50, 8" - $6.50.

16K RAMS

For the Apple, TRS-80 or Pet $8 each Part No. 4116/2117.

LEEDEX MONITOR

12" Black and White • 12 MHz Bandwidth • Handsome Plastic Case • $139.00

S-100 INTERFACE

AN S-100 bus Adapter—Motherboard for the TRS-80. Kit, Part No. HUH81DLXK, $295.95. Assembeled, Part No. HUH81DLXA, $375.95.

NOW! A FULL SUPPORT SYSTEM FOR TRS-80

• 32K of RAM • EPROM firmware • Disk control • Data acquisition • Parallel I/O • Serial I/O • Plug into GPA's Motherboard. GPA's quality design includes • 6-44 pin edge connectors • +5V, -5V, +12V, -12V external power supply required • Active termination. The Motherboard, Part No. GPA80, is only $149.95.

TAKE ADVANTAGE OF GPA-EXPANSION CARDS FOR THE GPA80

Memory cards: Now with Fortran compilers available for your TRS-80, additional expansion memory is a must! Card with sockets only, Part No. GPA801, $119.95. Card with 16K of 4116 Dynamic Ram, Part No. GPA802, $224.95. Card with 32K of 4116 Dynamic Ram, Part No. GPA803, $329.95. All cards come equipped with sockets to accomodate 32K of Ram.
EPROM firmware card. Put those valuable subroutines in firmware. Don't waste time loading and unloading tapes and disks. For 2708 or 2716 EPROMS, Part No. GPA806, $79.95.
Serial I/O card. Here's what you've been asking for, a full serial terminal interface, with RS-232C or 20 mA. Current loop. Input/output capabilities. Part No. GPA807, $79.95.
Parallel I/O Card. Control functions in the outside world, monitor and store real time events. Two parallel output ports. Dip switches select ports (0-254). Part No. GPA808, $79.95.

FLOPPY DISK STORAGE BINDERS

Three ring binder comes with ten transparent plastic sleeves which accommodate either twenty, five-inch or ten, eight-inch floppy disks. Binder & 10 holders, Part No. 810B—$9.95 • Extra holders, Part No. 810—69¢ each.

Three-ring binder with ten 5 1/4 inch jackets Part No. 510B—$9.95• Jackets only, fits standard 3-ring binders, Part No. 510—69¢ each.

DIGITAL CASSETTE

5 min. each side. Box of 10 $9.95. Part No. C-5.

TRENDCOM PRINTER

TRENDCOM 200, Part No. TRC0200 $495.95. **Interface** for TRS-80, Part No. T80A $49.95. For Apple II, Part No. TRCALL, $75.95. For PET, NO. TRCP2, $79.95. For Scoccerer, TRCSR1 $45.95.

SARGON: A Computer Chess Program

Features the complete program that won the 1978 West Coast Computer Faire Tournament. Part No. 00603 — TRS-80 Level II; Part No. 00604 — Apple II (24K). $19.95

SPINWRITER MODELS 5510 and 5520

Features—EIA RS-232C/CCITT V.24 Interface Standard • 55 Characters Per Second Maximum Print Rate • Impeccable Print Quality (DCR Quality) • Microprocessor Electronics • High Resolution Plotting/Graphing • Lowest Operating Noise Level • Self-Test Printing • Operator Engineered Control Panel • Prints Original and up to Seven Copies • NEC Information Systems new Model 5510 Receive Only and Model 5520 Keyboard Send/Receive SPINWRITER terminals are microprocessor controlled serial, impact terminals designed for remote printing applications where impeccable print quality is required. Model 5510 RO, Part No. NECA30759 $2795.95 • Model 5520 KSR, Part No. NECA30762 $3095.95

Send for FREE Catalog...a big self addressed envelope with 80¢ postage gets it fastest!

To Order: Mention part no., description, and price. In USA shipping paid by us for orders accompanied by check or money order. We accept C.O.D. orders (U.S. only) or a VISA or Master Charge no., expiration date, signature and phone no., shipping charges will be added. CA residents add 6.5% for tax. Outside USA add 15% for air mail postage and handling. Payment must be in U.S. dollars. Dealer inquiries invited. Prices subject to change without notice.

✓26

Order Line: (408) 448-0800

ELECTRONIC SYSTEMS

Dept. 80, P.O. Box 21638, San Jose, CA USA 95151

ADVERTISERS

*This advertiser prefers to be contacted directly.

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
Wafer Organizer	5.00
	$319.45

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.*

If you have any questions about the product, about Exatron, or ESFOA, please call the Hot Line. Address letters to ESFOA, 3559 Ryder St., Santa Clara, CA 95051.

Stringy Floppy is a trademark of Exatron Corporation.

HOT LINE (For Calls Outside CA) 800-538-8559

excellence in electronics

exatron

3555 Ryder Street ▪ Santa Clara, California 95051
(408) 737-7111