

also
Including

\$2.95 in USA

ON LOCALtion
THE INTERNATIONAL JOURNAL OF
COMPUTER ASSISTED INSTRUCTION

99'er

September/October 1981 Vol. 1, No. 3 **magazine**™

For Users of the
Texas Instruments
TI-99/4 and other
16-Bit TMS9900-Based
Personal Computer
Systems

Personal Record Keeping Applications

Business Software Evaluations

Printing Customized Forms

TI's New Editor/Assembler

Frugal Floppies

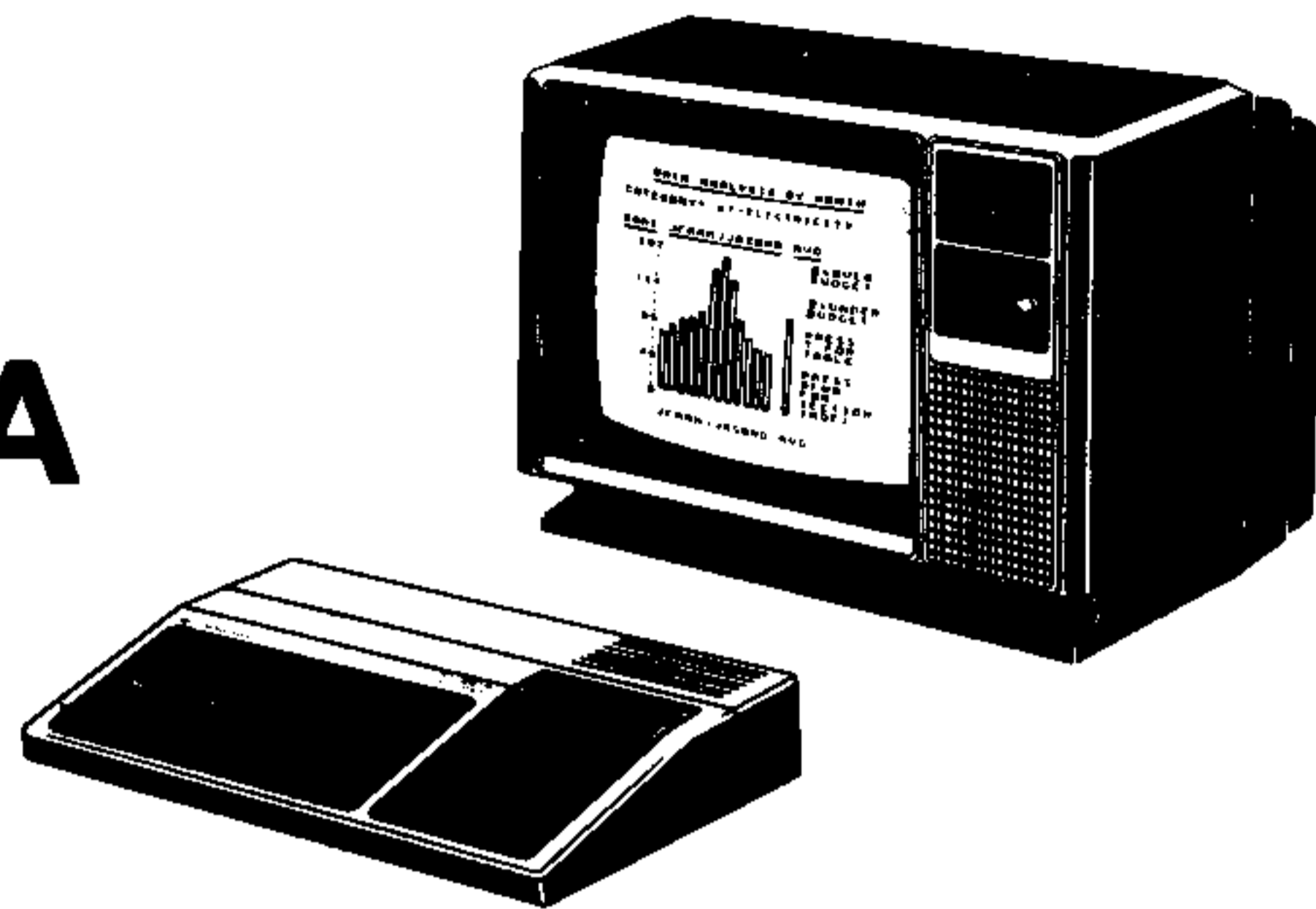
- GENERAL LEDGER
- ACCOUNTS PAYABLE
- ACCOUNTS RECEIVABLE
- INVENTORY
- MAILING LIST
- ORDER ENTRY
- PRODUCTION SCHEDULING
- SALES ANALYSIS
- CHECKBOOK RECONCILIATION
- HOUSEHOLD INVENTORY
- INCOME TAX RECORDS
- BUDGET MANAGEMENT
- REFERENCE FILING
- INVESTMENT TRACKING
- SCHOOL GRADE RECORDING

0882 501501 97381STE100
DONALD STEFFEN
10082 SILVERTON RD
SILVERTON OR 97381

Controlling the Paperwork Flood:
Microcomputer Management of Business & Personal Information

SAVE BIG ON THE TI-99/4A

Great savings on all TI-99/4A
Products and Accessories



"We will meet any price"
In this issue, if our competitor has goods on hand.

CONSOLE

TI-99/4A Home Computer ~~\$379.95~~

PERIPHERALS

Solid State Speech Synthesizer 114.95
Telephone Coupler (Modem) ~~169.95~~
RS-232 Accessories Interface ~~169.95~~
Disk Drive Controller (One Disk Manager
packed with Disk Controller) 229.95
Disk Memory Drive 369.95
Solid State Printer 299.95
Memory Expansion (32K RAM) ~~289.95~~
R. F. Modulator (TV Adapter) ~~39.95~~
10" Color Monitor 329.95

OPTIONAL ACCESSORIES

Wired Remote Controllers (Pair) 28.95
Dual Cassette Cable 12.95
Monitor Cable 17.95
Audio Adapter (Headphone Jack) 17.95
Thermal Paper (2 pack) 9.95
Blank Overlays (4 pack) 7.95

DOCUMENTATION

Beginning BASIC Manual 9.95
User's Reference Guide 9.95

APPLICATION PROGRAMS

Home Management/Personal Finance

Command Modules

Home Financial Decisions 26.95
Household Budget Management 34.95
Securities Analysis 49.95
Personal Record Keeping 39.95
Tax/Investment Record Keeping 59.95
Personal Real Estate 59.95

Diskette

Mailing List 54.95
Personal Financial Aids 18.95
Checkbook Manager 18.95
Business Aids Library-Finance Mgmt 36.95
Inventory Management 59.95

Cassette

Personal Financial Aids 14.95

Education/Personal Enrichment

Command Modules

Early Learning Fun 27.95
Beginning Grammar 27.95
Number Magic 18.95
Video Graphs 18.95
Video Chess 54.95
Physical Fitness 27.95
Early Reading 49.95
Music Maker ~~36.95~~
Weight Control and Nutrition 54.95
Addition and Subtraction I 36.95
Addition and Subtraction II 36.95
Multiplication I 36.95
TI-LOGO call

Diskette

Teach Yourself BASIC 29.95
Music Skills Trainer 27.95
Computer Music Box ~~18.95~~
Market Simulation 18.95
Teach Yourself Extended BASIC 21.95
Music Maker Demonstration 14.95
Basketball Statistics 22.95
Bridge Bidding I 27.95

Cassette

Teach Yourself BASIC 27.95
Music Skills Trainer 22.95
Computer Music Box 14.95
Market Simulation 14.95
Teach Yourself Extended BASIC 18.95
Bridge Bidding I 22.95

Entertainment

Command Modules

Football 24.95
Video Games I 27.95
Hunt the Wumpus 21.95
Indoor Soccer 27.95
Mind Challengers 22.95
A-Maze-Ing 22.95
The Attack † 34.95
Blasto † 22.95

Blackjack and Poker 22.95
Hustle † 22.95
Zero Zap † 18.95
Hangman 18.95
Connect Four † 18.95
Yahtzee † 22.95

Diskette

TI-Trek (w/speech) ~~12.95~~
Mystery Melody 14.95
Oldies But Goodies-Game I 18.95
Oldies But Goodies-Game II 22.95

OTHER APPLICATION PROGRAMS

Command Modules

Diagnostic 27.95
Demonstration 62.95
Speech Editor 36.95
Statistics 39.95
Terminal Emulator I 39.95
Extended BASIC 76.95
Terminal Emulator II ~~39.95~~

Diskette

Programming Aids I 14.95
Programming Aids II 22.95
Math Routine Library 27.95
Electrical Engineering Library 27.95
Programming Aids III ~~18.95~~
Graphing Package 18.95
Structural Engineering Library 27.95

Cassette

Programming Aids I 9.95
Math Routine Library 22.95
Electrical Engineering Library 22.95
Graphing Package 14.95
Structural Engineering Library 22.95

† Attack, Blasto, Hustle, Zero Zap, Connect Four and Yahtzee are trademarks of Milton Bradley.

• 32K Expansion requires use of Extended BASIC or TI-LOGO

• Speech Synthesizer requires use of Speech Editor or Terminal Emulator #2

tam's

INCORPORATED

14932 GARFIELD AVENUE
PARAMOUNT, CA 90723



Order Toll-Free
800-421-5188

In California call (213) 633-3262

Payment by check:

For faster delivery use cashier's check or money order (personal checks take 3 weeks to clear). Add shipping charges: 1% of your order (\$3.00 minimum). East of Mississippi River add \$1.50. California residents add 6% sales tax. Subject to availability. USA prices only.

Information line (213) 633-3262

Texas Instruments Personal Computer

Fifty Years
of
Innovation



It has a constantly growing library of quality software.

And there's still more.

Much, much more . . . our growing library of software programs

We know how important quality programs are to your experience with your computer. That's why, at Texas Instruments, they use professional talents — writers, artists, musicians, teachers, scientists, programmers, consultants — and nothing less developing our programs. The people of the

Texas Instruments Learning Center are even now working with experts in education like Scott, Foresman & Co., experts in entertainment like Milton Bradley and many others to make our software library richer every day and of the highest possible quality.

It's a gift that can grow and grow.

We carry the complete Texas Instruments personal computer line, hardware and software. Write for our very competitive price list including **TEXNETSM**

SM TEXNET is a servicemark of Texas Instruments Incorporated.

Mini Diskettes



Rely on Scotch® mini diskettes to keep your valuable data safe. Dependable Scotch diskettes are tested and guaranteed error-free. And the low abrasivity saves your read/write heads. They're compatible with most 5¼ inch diskette drives and are available in storage boxes, convenient library cases, and soft cartons.

\$249

Quantity 1

PRINTER DIP-81 \$399⁹⁵

Look at all the Features included as **STANDARD** for the price.

Interface: RS232

Print Speed: 100 characters per second.

Thruput: 60 lines per minute (80 characters per line bidirectional).

Character Formats: 7X7 matrix and 14X7 matrix expanded.

Character Set: Full upper/lower case 96 characters ASC II.

Media: Cut Sheet plain paper, Roll paper and Fan Fold single or multiple copy. Max. Width 8.5 in.

Feed: Friction Feed.

Loading: Either Rear or Bottom.

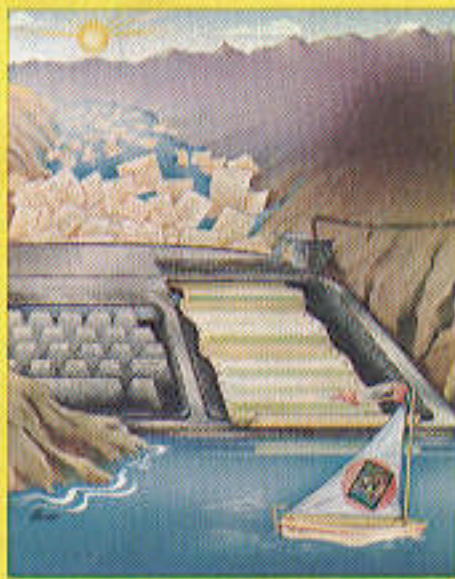
Operator Controls: Power on/off with indicator, Self Test, Top of Form, Line Feed.

Prices shown in this ad are our offer to end users. Dealer inquiries invited on all our products. **CHECK** our very competitive Dealer/OEM prices. Dealers must be multiple unit purchasers who are in business to resell. Our Dealer net prices are generally lower than dealers purchasing direct from the manufacturer receive.

A ACE Computer

7 W. Airline Hwy.
Corner of Hwys. 57 & 63
Waterloo, Iowa 50701
Phone: 319/

236-3861



THIS ISSUE'S COVER :

Don Fexer's cover painting depicts the menacing paper flood of raw data held in check by the massive microcomputer dam. Below, information is released into a calm reservoir in controllable amounts via the orderly printout. The boat with its recursive sail design signifies 99'er Magazine's contribution to the formation of this information reservoir.

Publisher / Editor
Gary M. Kaplan

Technical Editors
William K. Balthrop
David G. Brader
Mike Kovacich
G. R. Michaels
Patricia Swift

Program Editor
Cheryl Whitelaw

Contributing Editors
F. T. Berkey
Norma & John Clulow
Henry Gorman, Jr.
Duff Kurland
Mark Moseley
Samuel Pincus
Regena
George Struble
Dennis Thurlow
Jerry Wolfe

Production Manager
Norman Winney, Jr.

Production & Design
Greg Davis
Coleen Peplow
Corby Poticha
Cheryl Vigna

Office Manager
Pat Kaplan

Circulation & Fulfillment
Rod Hoyle

Marketing
Benjamin Kaplan

Advertising
Patana Ratanapreux
Tel. 503-485-8796

8 Interactive Forms Generator

By David G. Brader / Print your own customized business forms.

14 Getting Down to Business

By George Struble / Reviews of General Ledger and project budgeting software.

18 The 99'er Book Review

By John Clulow / A look at three books of applications software and the procedures for conversion to TI BASIC.

20 How to Write a BASIC Program that Writes BASIC Programs

By John Clulow / A 99'er Review of TI's Programming Aids III and an explanation of condensed format codes.

23 Software Conversion : Applesoft to TI BASIC

By Samuel D. Pincus / Converting non-graphic Apple programs is easier than you think.

25 Kelley's Korner : Games of Agility and Derring-Do

By W. K. Balthrop & Mark Moseley / Test your skill and nerve with these two aerial combat games.

29 The Third-Party Pascal Development System : Some User Impressions

By F. T. Berkey / Structured programming, speed, portability, and machine control for serious programmers and software developers.

32 Starting from Square One :

Murphy's Law and the Home Computer

By Samuel D. Pincus / Whatever can go wrong around computers, usually does . . . unless you guard against it.

Now What?

By Regena / You've just unboxed your new computer. Now what?

40 Personal Record Keeping : Managing a Mobile Home Park

By David G. Brader / TI's Command Module with math transformations is hard at work in this interesting application.

46 Memory & I/O Expansion plus an EPROM Programmer for the TM 990/189 University Board

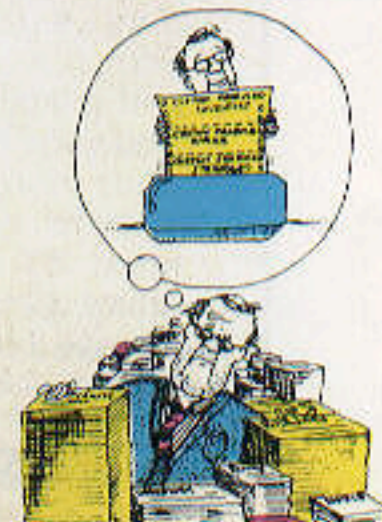
By John Caulfield / Turning the University Board into a super 16-bit development system.

48 Frugal Floppies : Adding Your Own Disk Drives

By Richard M. Bies / Save money by adding surplus drives to your TI-99/4 computer system.



Page 40



Page 8

Contents: September / October 1981

54 Super Language : A Preliminary Look at TI's Editor/Assembler
By Patricia Swift / The power and versatility of the 16-bit machine is finally unleashed.

56 How the New Tax Laws Affect Computer Owners
By Vernon K. Jacobs / A timely discussion of deductions, depreciation, and investment tax credit.



58 Let's Learn Notes
By Regena / Learning to play the piano is fun for all ages when you use this colorful program.

64 Computer Assisted Instruction : The State of the Art
By Gary M. Kaplan / Standards for evaluating educational software, and an OnloCAITion Review of Addition and Subtraction I.

69 Computer Chess Corner
By Jerry Wolfe / The author plays against Video Chess and reveals how the Command Module "thinks!"

72 The LOGO Poet : Using Recursion for List Handling
By Henry Gorman, Jr. / LOGO's non-graphics features are powerfully simple, yet elegant.

80 Typing Tutor, Part 2: Typing for Accuracy
By Regena / Improving your skills is easy when you practice with this programmed drill.



- | | |
|----------------------|--------------------------|
| 6 On Screen | 88 Dealer Directory |
| 84 Programming Hints | 91 Letters to the Editor |
| 86 99'er Bookstore | 94 Index to Advertisers |

99'er Magazine is published bimonthly by Emerald Valley Publishing Co., P.O. Box 5537, Eugene, OR 97405. The editorial office is located at 2715 Terrace View Drive, Eugene, OR 97405 (Tel. 503-485-8796). Subscription rates in U.S. and its possessions are \$15 for one year, \$28 for two years, and \$39 for three years. In Canada and Mexico \$18 for one year, \$34 for two years, \$48 for three years. Other foreign countries \$25 for one year surface, \$40 for one year air delivery. Single copy price in U.S. and its possessions is \$2.95, and \$3.50 in Canada and Mexico. Foreign subscription payment should be in United States funds drawn on a U.S. bank. Application to mail at controlled circulation postage rates is pending at Eugene, OR 97401. POSTMASTER: Send address changes to 99'er Magazine, P.O. Box 5537, Eugene, OR 97405. Subscribers should send all correspondence about subscriptions to above address.

Address all editorial correspondence to the Editor at 99'er Magazine, 2715 Terrace View Drive, Eugene, OR 97405. Unacceptable manuscripts will be returned if accompanied by sufficient first class postage and self-addressed envelope. Not responsible for lost manuscripts, photos, or program media. Opinions expressed by the authors are not necessarily those of 99'er Magazine. All mail directed to the "Letters to the Editor" column will be treated as unconditionally assigned for publication, copyright purposes, and use in any other publication or brochure, and are subject to 99'er Magazine's unrestricted right to edit and comment. 99'er Magazine assumes no liability for errors in articles or advertisements. Mention of products by trade name in editorial material or advertisements contained herein in no way constitutes endorsement of the product or products by 99'er Magazine or the publisher unless explicitly stated.

Each separate contribution to this issue and the issue as a collective work Copyright©1981 by Emerald Valley Publishing Co. All rights reserved. Where necessary, permission is granted by the copyright owner for libraries and others registered with the Copyright Clearance Center (CCC) to photocopy any article herein for the base fee of \$1.00 per copy of the article or item plus 25 cents per page. Payment should be sent directly to the CCC, 21 Congress Street, Salem, MA 01970. Copying done for other than personal or internal reference use without the permission of Emerald Valley Publishing Co. is prohibited. Requests for special permission or bulk orders should be addressed to the publisher.

Programming Conventions

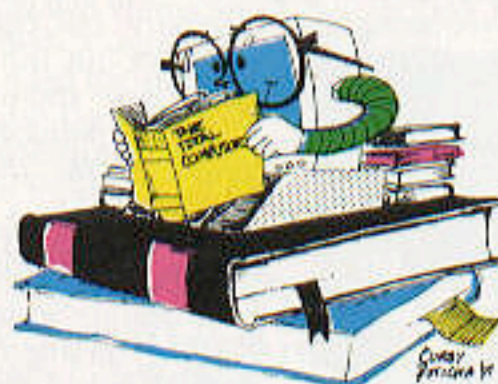
-  = Program as listed will completely fill available memory and cannot be RUN with disk controller (and possibly RS232 Interface) turned on.
-  = End of Program or Article

99'ER VERSION 7.81.1 X B M

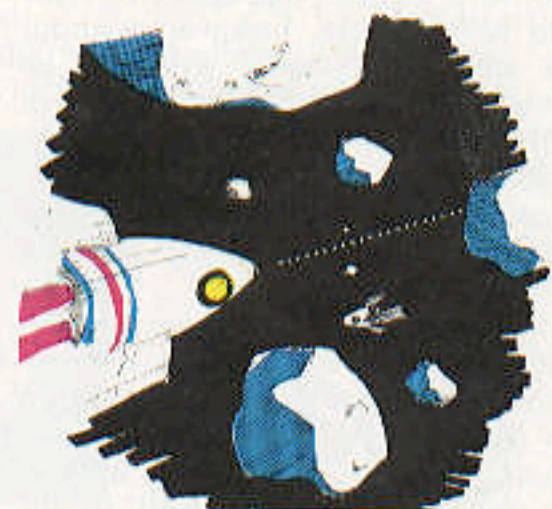
month _____
 year _____
 version _____
 1 = original program
 2 } = no. of update
 ... }
 n }
 TI Extended BASIC _____
 Expansion Memory Required _____



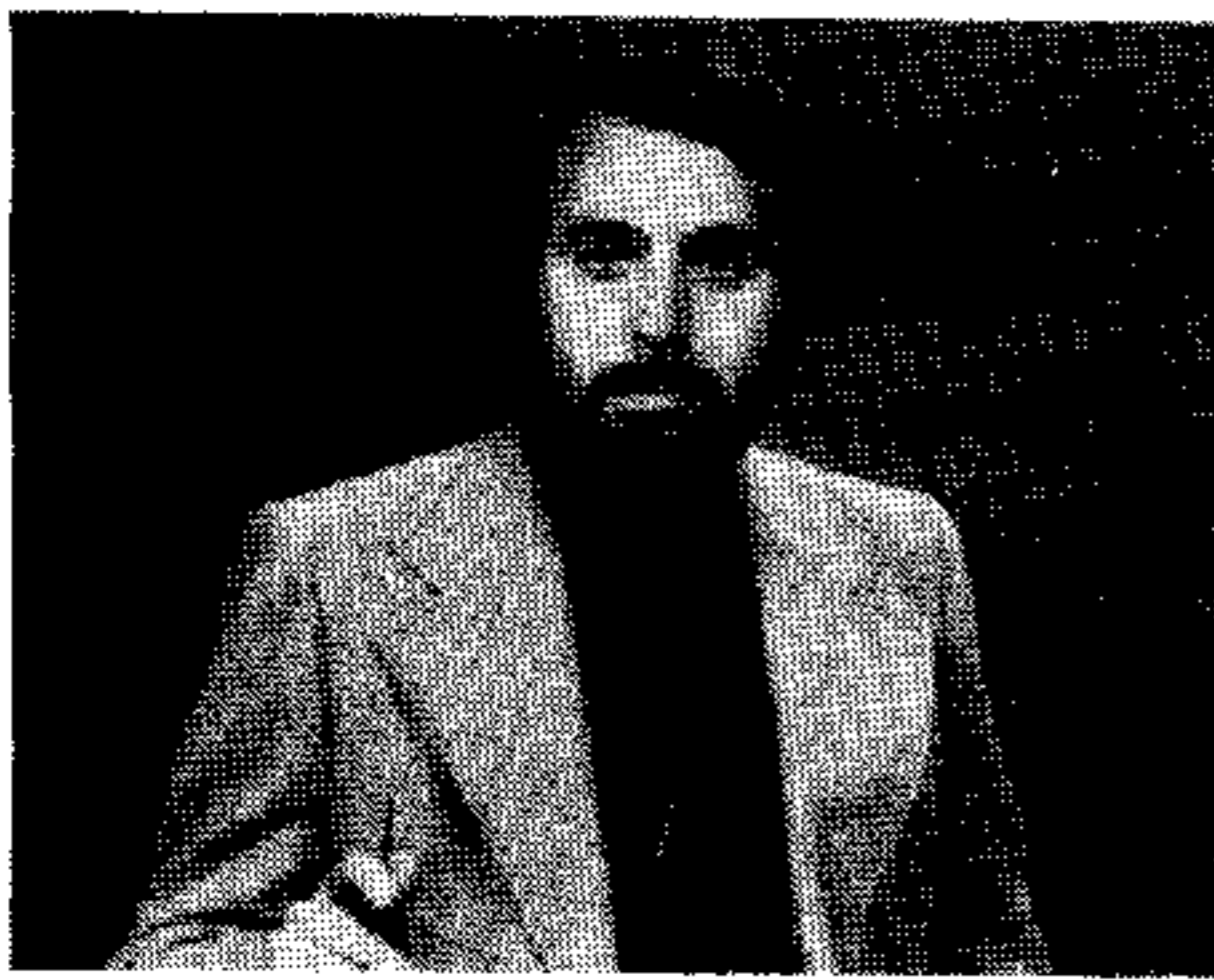
Page 32



Page 18



Page 25



ON SCREEN

By Gary M. Kaplan
Editor & Publisher

A personal computer can indeed be a many splendid thing. It can be a patient teacher, a faithful companion, a worthy opponent, as well as a partner in analysis and decision making. But it is the computer's particular suitability for processing data and managing information that has suggested this issue's theme and cover art—*Controlling the Paperwork Flood: Microcomputer Management of Business & Personal Information*.

When starting a new business, customized printing is often a considerable and unavoidable expense. This usually consists of a multitude of business forms including invoices, statements, mailing labels, purchase orders, shipping documents, and price lists. In our lead article, *Interactive Forms Generator*, Dave Brader will show you how to produce this customized paper arsenal *on demand*, at a cost of pennies—with all forms neatly filled out and ready for the mail.

I've selected another of Dave's articles to inaugurate a new feature column *Command Module Applications* (for which we are presently soliciting manuscripts). This time, in keeping with our theme, we'll examine some *Personal Record Keeping* routines for managing a trailer park. I think you'll find the use of the module's math transformations especially interesting.

No information management issue would be complete, however, without some software reviews, so we'll be *Getting Down to Business* with George Struble's evaluations of two presently available accounting and budgeting packages. You'll pick up some useful pointers here for evaluating *any* software product.

To round out our reviews, John Clulow examines three books of applications software. Many of the programs in these books are useful for maintaining business and personal records, and become ready-to-RUN in TI BASIC (if not so originally) with John's conversion tips that are easy to implement.

Managing information does necessitate some means of mass storage. Many of you have undoubtedly considered purchasing a disk controller and one or more disk drives, but have not been able to do so because of cost. If that's your situation, and you are willing to handle a little do-it-yourselfing, Dick Bies has good news. See his first-hand account of adding *Frugal Floppies*. And if the peripherals or additional software you buy is for a business purpose, don't overlook page 56's timely and informative discussion of *How New Tax Laws Affect Computer Owners*.

As soon as you have a floppy disk system, a whole new world of computer performance is yours. Good disk software, however, requires good programmers . . . and good programmers need (and get inspired by) good tools. Therefore, whenever new, *powerful* development tools—like the two you'll read about in this issue—appear, we can expect a host of good software to soon follow: In *Super Language*, Patricia Swift will introduce the soon-to-be-forthcoming Editor/Assembler for the TI-99/4(A), and John Clulow will do the honors to Programming Aids III in *How to Write a BASIC Program that Writes BASIC Programs* (the initial article in our new column *Advanced Programming Techniques*).

Those of you with the University Board, of course, already have a taste of TMS9900 assembly language. But now you may want to get involved in applications requiring more memory and customized EPROMs. If so, check out John Caulfield's overview of a new add-on board in *Memory & I/O Expansion, plus an EPROM Programmer for the TM 990/189 University Board*.

Serious programmers never seem to have enough languages in their tool kits. So if you're thinking of adding more languages to your TI-99/4(A), be sure to scan Tom Berkey's impressions of *The Third-Party Pascal Development System*.

But lest we forget the rest of you—novice programmers and new computer

users amongst you—who choose to stay with the console's built-in TI BASIC, this issue's menu also contains some useful fare: In *Software Conversions: Applesoft to TI BASIC*, Sam Pincus will show you a quick and easy way to add plenty of compatible programs to your own software library. Incidentally, Sam's by-line also appears this time in *Starting From Square One*—our new column for beginners where you'll learn about *Murphy's Law and the Home Computer* as well as pick up some applications ideas from our prolific (and mysterious—see page 84) Regena in *Square One's* second article, *Now What?*

As for all you games aficionados, have no fear . . . we've not forgotten you either. *Kelley's Korner* is back again—this time with two *Games of Agility and Derring-Do*. Be sure to read the Micro Editorial about these two games that appears in the *Letters to the Editor* pages.

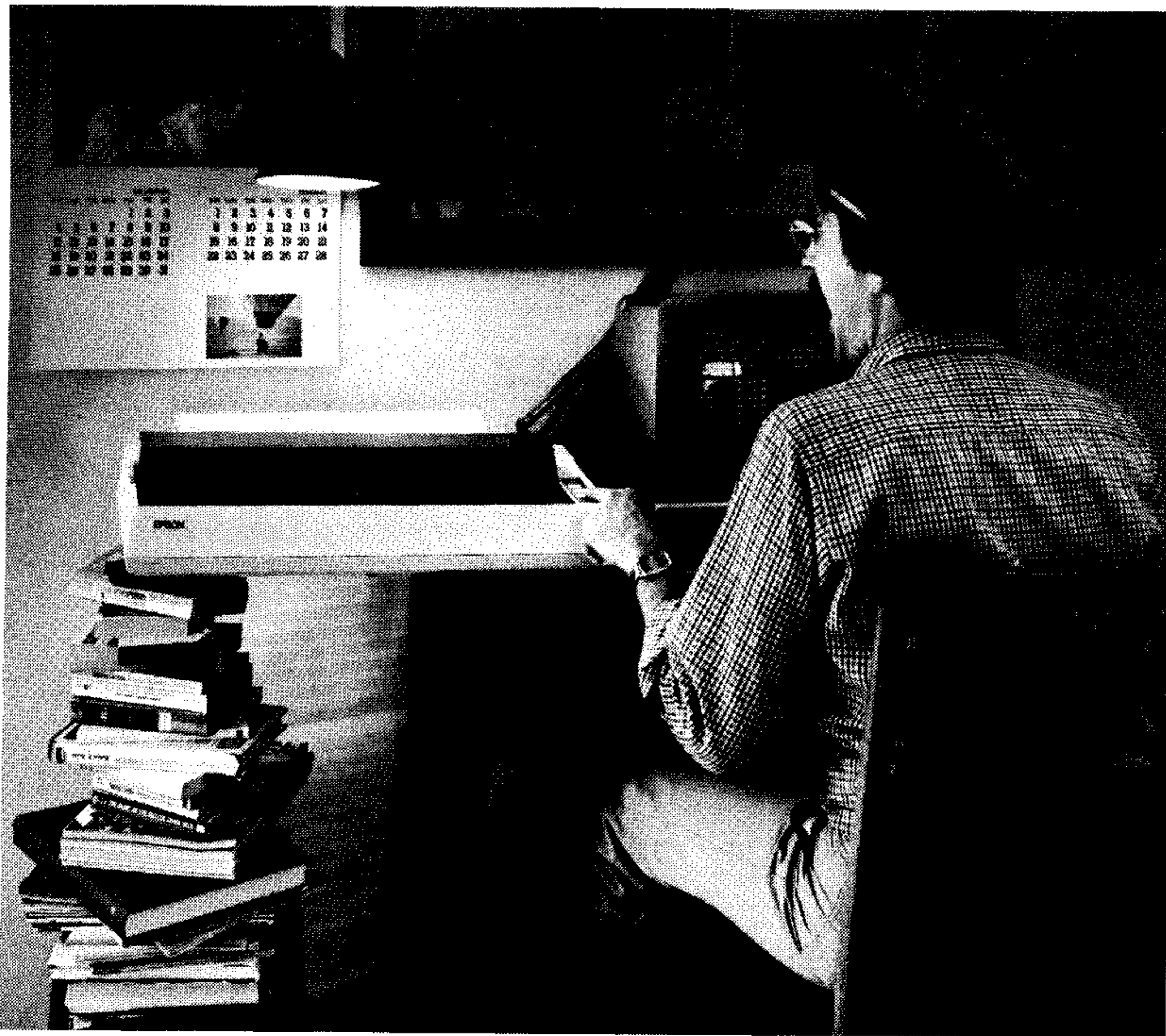
No issue of *99'er* would be complete without *OnLoCAItion*—our "magazine within a magazine." This time around, we take you on location to Glenview, Illinois for a look at a leading educational software developer, and a discussion of *Computer-Assisted Instruction: The State of the Art*.

Then it's off to LOGOland, where Hank Gorman will show you how to combine recursion and list handling operations and create *The LOGO Poet*—once more proving that programming is "kid's play."

As Part 3 in our chess series unfolds, you'll find Jerry Wolfe on the offense as he backs the TI Command Module into a *Computer Chess Corner*, and proceeds to beat Video Chess at its own game.

And finally, rounding out this issue, we have two programs that provide practice drills for all you aspiring typists and musicians—Regena's colorful *Typing for Accuracy* and *Let's Learn Notes*.

Until next issue—Have fun reading, learning, and RUNNING.



The MX-100. Not just better. Bigger.

Epson.

Our MX-80 was a pretty tough act to follow. I mean, how do you top the best-selling printer in the world?

Frankly, it wasn't easy. But the results of all our sleepless nights will knock your socks off.

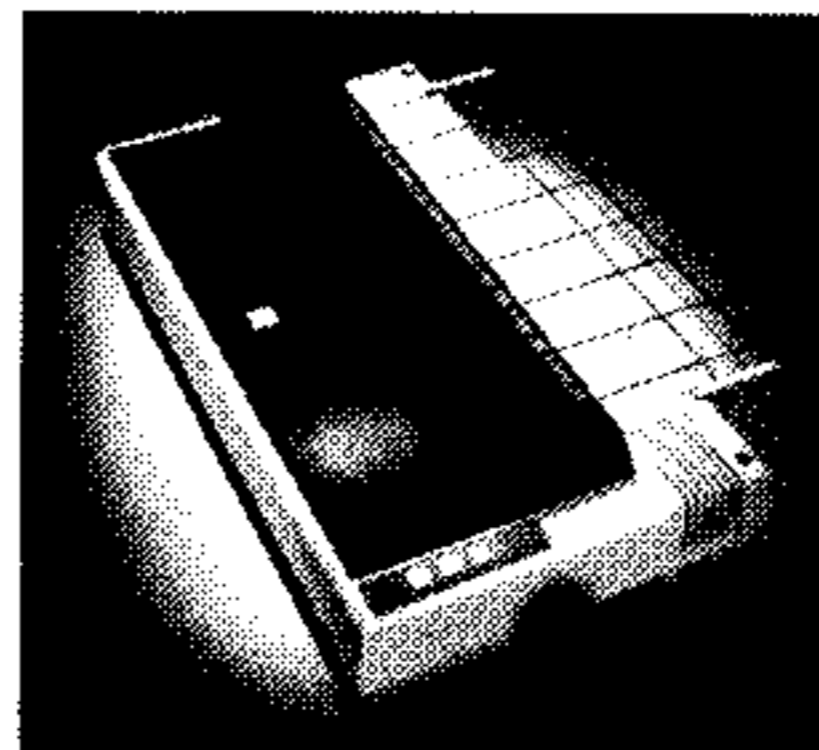
The MX-100 is a printer that must be seen to be believed. For starters, we built in unmatched correspondence quality printing, and an ultra-high resolution bit image graphics capability. Then we added the ability to print up to 233 columns of information on 15" wide paper to give you the most incredible spread sheets you're ever likely to see. Finally, we topped it all off with *both* a satin-smooth friction feed platen *and* fully adjustable, removable tractors. And the list of standard features goes on and on and on.

Needless to say, the specs on this machine — and especially at under \$1000 — are practically unbelievable. But there's something about the MX-100 that goes far

beyond just the specs; something about the way it all comes together, the attention to detail, the fit, the feel. Mere words fail us. But when you see an MX-100, you'll know what we mean.

All in all, the MX-100 is the most remarkable printer we've ever built. Which creates rather a large problem for those of us at Epson.

How are we going to top this?

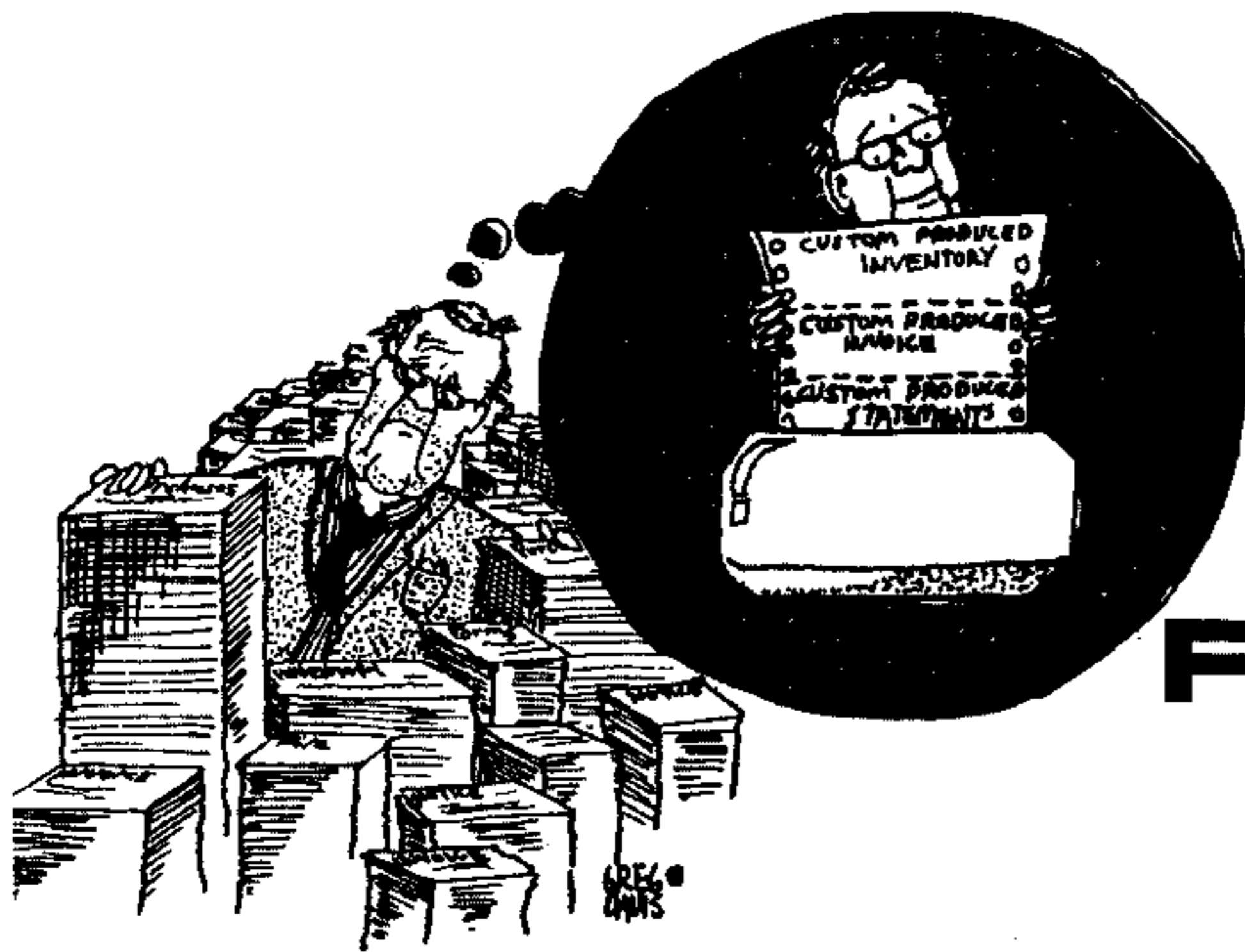


Your next printer.

EPSON
EPSON AMERICA, INC.

3415 Kashiwa Street • Torrance, California 90505 • (213) 539-9140

See the whole incredible Epson MX Series of printers at your Authorized Epson Dealer.



INTERACTIVE FORMS GENERATOR

By David G. Brader

When I started in business this past year, I decided to utilize my TI-99/4 as much as possible. One of the things I wanted to do with the computer was generate customized business forms such as purchase orders, price lists, invoices, and sales orders.

Right away you may be thinking: "He could buy all those forms ready made . . ." Yes, but that is not challenging or as much fun. Plus, to print up custom forms (ones that have your company name and address) is not cheap. Around here a minimum order of triplicate invoices costs about \$40 for 500. (And I probably wouldn't use all 500 before wanting to modify the form anyway . . .). Furthermore, if you multiply that \$40 figure by the 12 *different* forms (including price list pages) I presently have, you come up with a starting cost of \$480! That is almost enough money to buy an Epson MX-80 printer . . .

Well, you guessed it. I bought one (plus the serial interface, the RS232 cable, and the TI RS232 interface). The whole setup did cost more than the original estimate, but the added cost can be written off as "hobby money" for now. Then, with the right software I could sit down at the TI-99/4 keyboard, activate a program that would ask me questions to fill in the blanks of a form that was in memory, and finally print out as many copies as I wanted on the MX-80.

I wrote such a program that I call the INTERACTIVE FORMS GENERATOR. It is written in a general fashion to work with any correctly formatted data file. I then made up a FORM DATA FILE for each of my forms. A FORM DATA FILE is just a bunch of ASCII text lines stored in a "string array." Each text line may be written as a DATA LINE to be printed on the MX-80 or as a COMMAND LINE to direct the "INTERACTIVE FORMS GENERATOR" program.

How does it work?

The INTERACTIVE FORMS GENERATOR (IFG) program asks questions of the operator via the TI-99/4 screen. IFG accepts inputs from the operator via the keyboard and interprets instructions from the FORM DATA FILE's COMMAND LINES. In other words, the IFG program works with you to load your FORM DATA FILE, fill out the form, and finally print it out on the MX-80.

Let's say I am generating a Sales Order Acknowledgement form to send to a customer. First, I load the IFG program from diskette (or cassette). Second, I type RUN and hit enter. Third, the IFG program asks:

- MAKE A CHOICE—
1. LOAD NEW FORM FILE
 2. FILL OUT SAME FORM
 3. PRINT COPIES
 4. TERMINATE
- ?

I enter "1" and follow the instructions from the IFG program to load the Sales Order Acknowledgement FORM DATA FILE. Fourth, the program asks:

- MAKE A CHOICE—
1. LOAD NEW FORM FILE
 2. FILL OUT SAME FORM
 3. PRINT COPIES
 4. TERMINATE
- ?

I enter "2". Fifth, IFG will look through the FORM DATA FILE for the COMMAND LINES. Interpreting the lines, IFG will prompt me via the screen for the information needed to fill out the form's blanks. Also, interpreting the COMMAND LINES, IFG may perform simple math functions on fields of DATA LINES to calculate tax, totals, etc. After all the COMMAND LINES have been used, IFG again asks:

- MAKE A CHOICE—
1. LOAD NEW FORM FILE
 2. FILL OUT SAME FORM
 3. PRINT COPIES
 4. TERMINATE
- ?

This time I enter "3" and the IFG program asks:

ENTER NUMBER OF COPIES
TO PRINT—

I enter some number and IFG sends only the DATA LINES of the FORM DATA FILE to the MX-80 which does the rest! See Figure 2 for a look at the completed form sample.

Boy, isn't that slick . . . Just like the big guys (perhaps a little slower, but that's OK until the business grows to the point that speed is important). By the way, for Christmas I can generate a very long "form" letter with a year's worth of family news, then use IFG to fill out a separate salutation for each relative. So the whole family gets the latest without my getting writer's cramp! I'll bet that with your imagination and creativity you will come up with some neat applications for IFG too . . .

OK, OK. You want to know how you can make one of these FORM DATA FILES, don't you? Well then, there are a couple ways:

Building a FORM DATA FILE : Method 1

If you have some kind of "editor" program that will build an "ASCII text string array" you are all set. All you have to do is make sure it will output the special ASCII control codes used by the printer to do its tricks. Also, it must output the FORM DATA FILE to cassette or diskette in a compatible format. Listings 1 and 2 for subroutines CASSOUT and

DISKOUT illustrate what is needed. If you don't have an "editor" program, see Method 2, below:

Building a FORM DATA FILE : Method 2

This is more difficult. Nevertheless, I have a real simple (but tedious) method of building the FORM DATA FILE that you can use.

STEP 1.

Sit down with a pad of paper and a pencil. Now design each character-string line of the form. Use the "CHR\$()" function to put special codes in the string that can't be directly entered by a key on the 99/4 keyboard. The codes can be looked up in the MX-80 (or other printer's) manual. The samples shown below are: CHR\$(27) "escape code", CHR\$(13) "carriage return code", CHR\$(10) "line feed code":

```
CHR$(27)&"E"&"THE DOG RAN HOME QUICKLY"&
CHR$(10)&CHR$(13)
```

STEP 2.

Now fire up your 99/4. Enter the following program lines:

```
100 REM
110 REM "FILEBUILD" PROGRAM
120 REM
130 OPTION BASE 1
140 DIM A$(70)
150 REM
```

Then enter your character-string lines from paper into the string array via the TI-99/4 keyboard as follows:

```
160 REM NOW FOR THE CHARACTER STRINGS IN THE ARRAY
170 REM
180 A$(1)=CHR$(27)&"E"&"THE DOG RAN HOME QUICKLY"
&CHR$(10)&CHR$(13)
190 A$(2)="AFTER DINNER THE DOG BURPED AND
SCRATCHED HIS EAR."&CHR$(10)&CHR$(13)
200 A$(3)=
* *
* *
* *
??? A$(??)="THAT'S ALL FOLKS!"&CHR$(10)&CHR$(10)
&CHR$(13)
1000 REM
```

Now enter the following lines of program code:

```
1010 REM MAKE X EQUAL TO THE NUMBER OF LINES IN A*
1020 REM
1030 X=??
1040 PRINT "WRITE FILE TO?"
1050 PRINT " 1 - CS1"
1060 PRINT " 2 - DSK"
1070 INPUT CHOICE
1080 IF (CHOICE<1)+(CHOICE>2)=-1 THEN 1040
1090 ON CHOICE GOSUB 2000,3000
1100 PRINT " C O M P L E T E "
1110 PRINT "NOW SAVE ME ON TAPE OR DISK....."
1120 END
1130 REM
1140 REM OUTPUT SUBROUTINES FOLLOW.....
1150 REM
```

Finally, enter the two subroutines CASSOUT and DISKOUT starting at lines 2000 and 3000.

STEP 3.

Type in RUN. You should end up with your own FORM DATA FILE on tape or diskette. This can now be used with the IFG program.

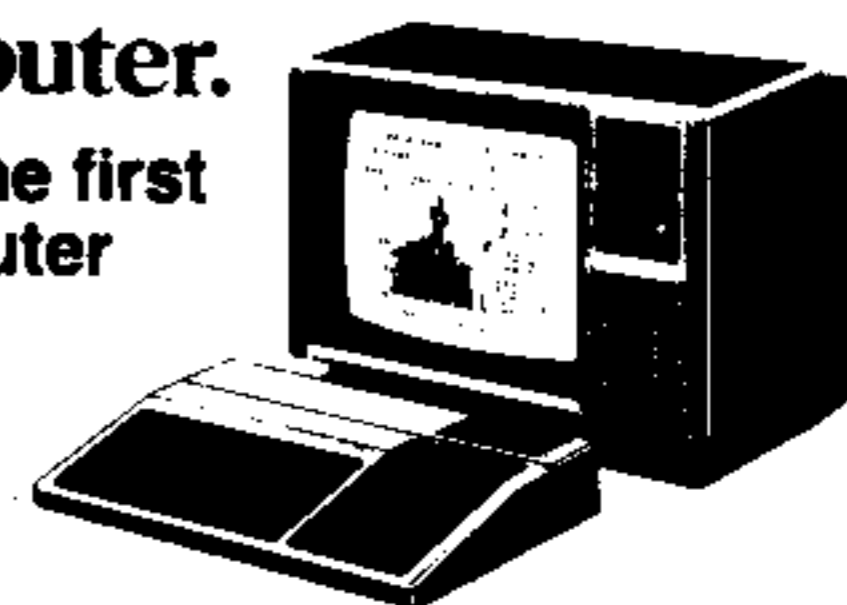
STEP 4.

Hold it! Don't turn off the TI-99/4 yet! SAVE your "FILEBUILD" program on tape or diskette too. Chances are you will want to modify that form because of errors or change of design in the future. OK, now you can turn off the computer and hit the sack. (Notice that this kind of work is always done at midnight...)

To help clarify the above process, I generated a simple "FILEBUILD" program (Listing 3). Note that text lines A\$(1)-A\$(13) are DATA LINES and text lines A\$(14)-A\$(19) are COMMAND LINES (more on these next). Listing 4 shows the resultant FORM DATA FILE (as printed by

Texas Instruments TI-99/4A Home Computer.

Designed to be the first true home computer — for skilled computer users or for beginners.



DISCOUNT PRICES

TI-99/4A
TI Software
TI peripherals



EPSON Printers

Write for Price List



KOMPUTAR WORKS

P.O. Box 489
Electric City, Washington 99123-0489
509/633-2653

TI-PWRITER

for
*** Extended Basic ***
a complete WORD PROCESSOR
for the 99/4

ANY Printer — T.P., RS232C (or screen only).

FULL upper & lower case (including 99/4) — even on screen print.

Input typing SPEED — over 100 words per minute.

ANY Line Length — 28 to 254 characters per print line.

Complete text EDITING — by cursor control; including insert & delete lines, partial text print, printer halt or abort without text loss, page FWD & BKWD, and automatic line centering.

ANY Input/Output storage of text — disc, cassette, cassette input/disc output, or vice versa.

Holds 3000 characters (before storage) — 50,000 characters per disc or 60 minute cassette.

COMPLETE Software Control of Printer (depending upon its capabilities) — for enhanced print, underlining, formatting, variable cpi.

No Special Equipment — monitor, console, Extended Basic module, storage device (preferably), printer (or a friend with one).

Comes with a 20 page instruction booklet.

If you are not completely satisfied you may return it within 15 days for a full refund of purchase price.

CASSETTE: \$32.00

DISCETTE: \$35.00

Send check to: **Extended Software Company**
11987 Cedar Creek Drive
Cincinnati, Ohio 45240
(513) 825-6645

```

100 REM *****
110 REM *****SAMPLE*****
120 REM * FILE BUILD FOR INTERACTIVE FORMS GENERATOR: *
130 REM *****SHIPPING LABEL*****
140 REM *****
150 REM 99'ER VERSION 9.81.1
160 REM BY DAVID G. BRADER
170 REM
180 REM
190 OPTION BASE 1
200 DIM A$(70)
210 REM
220 REM NOW FOR THE CHARACTER STRINGS IN THE ARRAY
230 REM
240 A$(1)=CHR$(27)&"E"&CHR$(15)&CHR$(14)&"KOMPUTAR WORKS"&CHR$(10)&CHR$(13)
250 A$(2)="P.O.BOX 483"&CHR$(10)&CHR$(13)
260 A$(3)="143 SUNSET DRIVE"&CHR$(10)&CHR$(13)
270 A$(4)="ELECTRIC CITY, WASHINGTON"&CHR$(10)&CHR$(13)
280 A$(5)=" 99123"&CHR$(10)&CHR$(13)
290 A$(6)=CHR$(10)&CHR$(10)&CHR$(10)&CHR$(10)&CHR$(10)&CHR$(10)&CHR$(13)
300 A$(7)=CHR$(14)&" SHIP TO:"&CHR$(10)&CHR$(13)
310 A$(8)=" "&CHR$(10)&CHR$(13)
320 A$(9)=" "&CHR$(10)&CHR$(13)
330 A$(10)=" "&CHR$(10)&CHR$(13)
340 A$(11)=" "&CHR$(10)&CHR$(13)
350 A$(12)=" "&CHR$(10)&CHR$(13)
360 A$(13)=CHR$(10)&CHR$(10)&CHR$(10)&CHR$(10)&CHR$(10)&CHR$(13)
370 A$(14)="!!"&CHR$(34)&"SHIPPING LABEL"&CHR$(10)&CHR$(34)&CHR$(13)
380 A$(15)="!!"&CHR$(34)&"CUSTOMER NAME"&CHR$(34)&" :8:35:70:"&CHR$(13)
390 A$(16)="!!"&CHR$(34)&"STREET ADDRESS"&CHR$(34)&" :9:35:70:"&CHR$(13)
400 A$(17)="!!"&CHR$(34)&"CITY"&CHR$(34)&" :10:35:70:"&CHR$(13)
410 A$(18)="!!"&CHR$(34)&"STATE"&CHR$(34)&" :11:35:70:"&CHR$(13)
420 A$(19)="!!"&CHR$(34)&"ZIP CODE"&CHR$(34)&" :12:40:60:"&CHR$(13)
430 REM
440 REM MAKE X EQUAL THE NUMBER OF LINES IN A$
450 REM
460 X=19
470 PRINT "WRITE FILE TO?"
480 PRINT " 1 - CS1"
490 PRINT " 2 - DISK"
500 INPUT CHOICE
510 IF (CHOICE<1)+(CHOICE>2)=-1 THEN 470
520 ON CHOICE GOSUB 590,720
530 PRINT "C O M P L E T E"
540 PRINT "NOW SAVE ME ON TAPE OR DISK....."
550 END
560 REM
570 REM OUTPUT SUBROUTINES FOLLOW.....
580 REM
590 REM
600 REM SUBROUTINE "CASSOUT"
610 REM
620 OPEN #1:"CS1",INTERNAL,OUTPUT,FIXED 192
630 REM
640 REM "X" MUST EQUAL THE NUMBER OF TEXT LINES
650 REM
660 PRINT #1:X
670 FOR I=1 TO X+1 STEP 2
680 PRINT #1:A$(I),A$(I+1)
690 NEXT I
700 CLOSE #1
710 RETURN
720 REM
730 REM SUBROUTINE "DISKOUT"
740 REM
750 PRINT "ENTER WHICH DISK, 1-3?"
760 INPUT DISK
770 IF (DISK<1)+(DISK>3)=-1 THEN 750
780 PRINT "ENTER FILENAME:"
790 INPUT NAME$
800 IF (LEN(NAME$)<1)+(LEN(NAME$)>10)=-1 THEN 780
810 OPEN #1:"DISK"&STR$(DISK)&". "&NAME$,OUTPUT,
INTERNAL,VARIABLE 132
820 REM
830 REM "X" MUST EQUAL THE NUMBER OF TEXT LINES
840 REM
850 PRINT #1:X
860 FOR I=1 TO X
870 PRINT #1:A$(I)
880 NEXT I
890 CLOSE #1
900 RETURN

```

Listing 1.

```

2000 REM
2010 REM SUBROUTINE "CASSOUT"
2020 REM
2030 OPEN #1:"CS1",INTERNAL,OUTPUT,FIXED 192
2040 REM
2050 REM "X" MUST EQUAL THE NUMBER OF TEXT LINES
2060 REM
2070 PRINT #1:X
2080 FOR I=1 TO X+1 STEP 2
2090 PRINT #1:A$(I),A$(I+1)
2100 NEXT I
2110 CLOSE #1
2120 RETURN

```

Listing 2.

```

3000 REM
3010 REM SUBROUTINE "DISKOUT"
3020 REM
3030 PRINT "ENTER WHICH DISK, 1-3?"
3040 INPUT DISK
3050 IF (DISK<1)+(DISK>3)=-1 THEN 3030
3060 PRINT "ENTER FILENAME:"
3070 INPUT NAME$
3080 IF (LEN(NAME$)<1)+(LEN(NAME$)>10)=-1 THEN 3060
3090 OPEN #1:"DISK"&STR$(DISK)&". "&NAME$,OUTPUT,
INTERNAL,VARIABLE 132
3100 REM
3110 REM "X" MUST EQUAL THE NUMBER OF TEXT LINES
3120 REM
3130 PRINT #1:X
3140 FOR I=1 TO X
3150 PRINT #1:A$(I)
3160 NEXT I
3170 CLOSE #1
3180 RETURN

```

Listing 4.

```

1 ESCESISKOMPUTAR WORKSLFCR
2 P.O.BOX 483LFCR
3 143 SUNSET DRIVELFCR
4 ELECTRIC CITY, WASHINGTONLFCR
5 99123LFCR
6 LFLFLFLFLFCR
7 SO SHIP TO:LFCR
8 LFCR
9 LFCR
10 LFCR
11 LFCR
12 LFCR
13 LFLFLFLFLFCR
14 !!"SHIPPING LABEL"CR
15 !!"CUSTOMER NAME":8:35:70:CR
16 !!"STREET ADDRESS":9:35:70:CR
17 !!"CITY":10:35:70:CR
18 !!"STATE":11:35:70:CR
19 !!"ZIP CODE":12:40:60:CR

```



Now, you can separate your TI 99/4 Computer from your TI Peripherals



BETTER CONVENIENCE • EASE OF OPERATION • VERSATILITY

Our (2') Two Foot Cable is made to interconnect your Texas Instruments TI 99/4 Computer with almost any combination of Printer, RS 232, Tape, or Controller.

ORDER FORM

SEND ORDER WITH PAYMENT TO:

RCL COMPUTERS

411 Allegheny River Blvd.

Oakmont, PA. 15139

(412) 828-4301



COMPUTERS

SHIP THE FOLLOWING:

_____ Heavy Duty 2' Cable At \$44.95 ea.

_____ Heavy Duty 3' Cable At \$54.95 ea.

Check No. _____ Amount _____

Money Order _____ Amount _____

Price F.O.B. Pittsburgh, PA.

PA. Deliveries add 6% Sales Tax

Allow 3-4 weeks delivery

SHIP TO:

NAME _____

ADDRESS _____

CITY _____ STATE _____ ZIP _____

KOMPUTAR WORKS
P.O. BOX 483
143 SUNSET DRIVE
ELECTRIC CITY, WASHINGTON
99123

Figure 1.

SHIP TO:

MR. CHIP BUGGS
9900 SEAMOSS AVE. APT. # 102
ELECTRIC CITY
WASHINGTON
99123

my "editor" program). Figure 1 shows the results of running IFG using this FORM DATA FILE.

Power to the IFG !

How do we get the FORM DATA FILE to tell the IFG what to do? By making up COMMAND LINES. What makes a COMMAND LINE special? It must start with these two characters !!. What can a COMMAND LINE tell IFG to do? It can tell it to output a message to the T1-99/4 display. Here is a sample:

!! "THIS MESSAGE WILL BE WRITTEN ON THE 99/4 DISPLAY"

Note that anything between quotes will be displayed.

What about telling it to get something from the 99/4 keyboard? OK—whenever IFG does this, he stuffs the information obtained into a line of the FORM DATA FILE either right-justified or left-justified. To get input from the keyboard and stuff it left-justified, use this FIELD DEFINITION syntax:

!! :28:1:32: (ie: :line#:first character:last character:)

To get it stuffed right-justified (needed for lining up decimals) do the same except add a ">" sign after the first ":". Example: !! :>28:1:32:

By the way, IFG will show you on the display how much space you have to write in and will let you know if you overflow.

Didn't you say something about IFG doing math calculations, you ask? Right. You can write COMMAND LINES to add, subtract, multiply, and divide. Each term and operator simply must be enclosed in parentheses. A term may be a FIELD DEFINITION or a constant. Here are some samples:

!! (:28:1:32:)(*) (.05) (=) (:34:17:24:)
!! (:2:1:4:)(+) (:3:1:4:)(+) (:4:1:4:)(=) (:6:1:4:)
!! (:34:57:68:)(*) (:34:22:32:)(=) (:>34:70:82:)
!! (12.1)(/)(:2:22:32:)(-)(33.3)(=)(:2:22:32:)
I know what you're probably saying right now: "Wow, that really is a lot of power! Is that all IFG can do?" Well, there is one more small thing. You can write a COMMAND LINE sequence that will repeat a given number of times. Each time the sequence is repeated, all included FIELD DEFINITION line numbers are incremented. IFG always asks after each repeat cycle if you want to do another. This last feature makes it simple to fill out a form that is a multi-line list. Here is a sample repeat sequence:

!!@10; "stock #?" :28:2:10: "description?" :28:12:44:
!!"scheduled ship date?" :28:46:56: "quantity?" :>28:58:62:
!!"unit price?" :>28:65:70:
!! (:28:58:62:)(*) (:28:65:70:)(=) (:>28:72:79:) @

This sequence will start at FORM DATA FILE line 28 and go to line 38. Notice the repeat sequence is bracketed by "@" symbols and the number between the first "@" and the ";" tells how many repeat cycles. Study this sample for a bit and figure out what it does. Then you can look over Listing 5. It

Listing 5.

```

1 ESCFESCHDC4DC2LFCR
2 SIS0ESCEESCB          KOMPUTAR WORKS LFLFCR
3                          P.O.Box 483 LFCR
4                          Electric City LFCR
5                          Washington LFCR
6                          99123 LFCR
7                          (509) 633-2653 LFLFLFCR
8 DC2DC4ESCFESCHLFCR
9 SOESCE          SALES ORDER ACKNOWLEDGEMENT LFLFLFCR
10 ESCF87S0Date-          S.O.number-CR
11 DC4DC2                                          LFCR
12 LFCR
13 SIS0Sold to-          Ship to-CR
14 DC4DC2                                          LFCR
15 HT                                          LFCR
16 HT                                          LFCR
17 HT                                          LFCR
18 HT                                          LFCR
19 LFSIS0Ship via-CR
20 DC4DC2                                          LFLFCR
21 -----LFCR
22 SIS0Stock #          Description          Scheduled Quan- Unit AmountCR
23 DC4DC2                                          LFCR
24 SIS0          ship date tity priceCR
25 DC4DC2                                          LFCR
26 DC4DC2CR
27 -----LFCR
28 HT                                          LFCR
29 HT                                          LFCR
30 HT                                          LFCR
31 HT                                          LFCR
32 HT                                          LFCR
33 HT                                          LFCR
34 HT                                          LFCR
35 HT                                          LFCR
36 HT                                          LFCR
37 HT                                          LFCR
38 -----LFCR
39 SIS0          SUBTOTAL =CR
40 DC4DC2                                          LFCR
41 SIS0          TAX =CR
42 DC4DC2                                          LFCR
43 SIS0          FREIGHT =CR
44 DC4DC2                                          LFCR
45 SIS0          TOTAL =CR
46 DC4DC2                                          LFCR
47 ESCB:NUCLCR
48 VTSIThank you for the order. Remember "word of mouth" advertising LFCR
49 keeps our costs down..... So help spread the word!LFCR
50 FFCR
51 !! "SALES ORDER ACKNOWLEDGEMENT LFLFL" LFCR
52 !! "date?" :11:11:30: "S.O. number?" :11:53:64:CR
53 !! "sold to?" :14:14:37: "sold to address lines=4LF"CR
54 !! "address #1" :15:13:36: "address #2" :16:13:36:CR
55 !! "address #3" :17:13:36: "address #4" :18:13:36:CR
56 !! "ship to?" :14:32:76: "ship to address lines=4LF"CR
57 !! "address #1" :15:51:75: "address #2" :16:51:75:CR
58 !! "address #3" :17:51:75: "address #4" :18:51:75:CR
59 !! "ship via?" :20:15:54:CR
60 !!@10; "stock #?" :28:2:10: "description?" :28:12:44:CR
61 !! "scheduled ship date?" :28:46:56: "quantity?" :>28:58:62:CR
62 !! "unit price?" :>28:65:70: (:28:58:62:)(*) (:28:65:70:)(=) (:>28:72:79:) @CR
63 !! (:28:72:79:)(+)(:29:72:79:)(+)(:30:72:79:)(+)(:31:72:79:)(+)(:32:72:79:)(+)(:33:72:79:)(+)(:34:72:79:)(+)(:35:72:79:)(+)(:36:72:79:)(+)(:37:72:79:)(+)(:38:72:79:)(+)(:39:72:79:)(+)(:40:72:79:)(+)(:41:72:79:)(+)(:42:72:79:)(+)(:43:72:79:)(+)(:44:72:79:)(+)(:45:72:79:)(+)(:46:72:79:)(+)(:47:72:79:)(+)(:48:72:79:)(+)(:49:72:79:)(+)(:50:72:79:)(+)(:51:72:79:)(+)(:52:72:79:)(+)(:53:72:79:)(+)(:54:72:79:)(+)(:55:72:79:)(+)(:56:72:79:)(+)(:57:72:79:)(+)(:58:72:79:)(+)(:59:72:79:)(+)(:60:72:79:)(+)(:61:72:79:)(+)(:62:72:79:)(+)(:63:72:79:)(+)(:64:72:79:)(+)(:65:72:79:)(+)(:66:72:79:)(+)(:67:72:79:)(+)(:68:72:79:)(+)(:69:72:79:)(+)(:70:72:79:)(+)(:71:72:79:)(+)(:72:72:79:)(+)(:73:72:79:)(+)(:74:72:79:)(+)(:75:72:79:)(+)(:76:72:79:)(+)(:77:72:79:)(+)(:78:72:79:)(+)(:79:72:79:)(+)(:80:72:79:)(+)(:81:72:79:)(+)(:82:72:79:)(+)(:83:72:79:)(+)(:84:72:79:)(+)(:85:72:79:)(+)(:86:72:79:)(+)(:87:72:79:)(+)(:88:72:79:)(+)(:89:72:79:)(+)(:90:72:79:)(+)(:91:72:79:)(+)(:92:72:79:)(+)(:93:72:79:)(+)(:94:72:79:)(+)(:95:72:79:)(+)(:96:72:79:)(+)(:97:72:79:)(+)(:98:72:79:)(+)(:99:72:79:)(+)(:100:72:79:)(+)(:101:72:79:)(+)(:102:72:79:)(+)(:103:72:79:)(+)(:104:72:79:)(+)(:105:72:79:)(+)(:106:72:79:)(+)(:107:72:79:)(+)(:108:72:79:)(+)(:109:72:79:)(+)(:110:72:79:)(+)(:111:72:79:)(+)(:112:72:79:)(+)(:113:72:79:)(+)(:114:72:79:)(+)(:115:72:79:)(+)(:116:72:79:)(+)(:117:72:79:)(+)(:118:72:79:)(+)(:119:72:79:)(+)(:120:72:79:)(+)(:121:72:79:)(+)(:122:72:79:)(+)(:123:72:79:)(+)(:124:72:79:)(+)(:125:72:79:)(+)(:126:72:79:)(+)(:127:72:79:)(+)(:128:72:79:)(+)(:129:72:79:)(+)(:130:72:79:)(+)(:131:72:79:)(+)(:132:72:79:)(+)(:133:72:79:)(+)(:134:72:79:)(+)(:135:72:79:)(+)(:136:72:79:)(+)(:137:72:79:)(+)(:138:72:79:)(+)(:139:72:79:)(+)(:140:72:79:)(+)(:141:72:79:)(+)(:142:72:79:)(+)(:143:72:79:)(+)(:144:72:79:)(+)(:145:72:79:)(+)(:146:72:79:)(+)(:147:72:79:)(+)(:148:72:79:)(+)(:149:72:79:)(+)(:150:72:79:)(+)(:151:72:79:)(+)(:152:72:79:)(+)(:153:72:79:)(+)(:154:72:79:)(+)(:155:72:79:)(+)(:156:72:79:)(+)(:157:72:79:)(+)(:158:72:79:)(+)(:159:72:79:)(+)(:160:72:79:)(+)(:161:72:79:)(+)(:162:72:79:)(+)(:163:72:79:)(+)(:164:72:79:)(+)(:165:72:79:)(+)(:166:72:79:)(+)(:167:72:79:)(+)(:168:72:79:)(+)(:169:72:79:)(+)(:170:72:79:)(+)(:171:72:79:)(+)(:172:72:79:)(+)(:173:72:79:)(+)(:174:72:79:)(+)(:175:72:79:)(+)(:176:72:79:)(+)(:177:72:79:)(+)(:178:72:79:)(+)(:179:72:79:)(+)(:180:72:79:)(+)(:181:72:79:)(+)(:182:72:79:)(+)(:183:72:79:)(+)(:184:72:79:)(+)(:185:72:79:)(+)(:186:72:79:)(+)(:187:72:79:)(+)(:188:72:79:)(+)(:189:72:79:)(+)(:190:72:79:)(+)(:191:72:79:)(+)(:192:72:79:)(+)(:193:72:79:)(+)(:194:72:79:)(+)(:195:72:79:)(+)(:196:72:79:)(+)(:197:72:79:)(+)(:198:72:79:)(+)(:199:72:79:)(+)(:200:72:79:)(+)(:201:72:79:)(+)(:202:72:79:)(+)(:203:72:79:)(+)(:204:72:79:)(+)(:205:72:79:)(+)(:206:72:79:)(+)(:207:72:79:)(+)(:208:72:79:)(+)(:209:72:79:)(+)(:210:72:79:)(+)(:211:72:79:)(+)(:212:72:79:)(+)(:213:72:79:)(+)(:214:72:79:)(+)(:215:72:79:)(+)(:216:72:79:)(+)(:217:72:79:)(+)(:218:72:79:)(+)(:219:72:79:)(+)(:220:72:79:)(+)(:221:72:79:)(+)(:222:72:79:)(+)(:223:72:79:)(+)(:224:72:79:)(+)(:225:72:79:)(+)(:226:72:79:)(+)(:227:72:79:)(+)(:228:72:79:)(+)(:229:72:79:)(+)(:230:72:79:)(+)(:231:72:79:)(+)(:232:72:79:)(+)(:233:72:79:)(+)(:234:72:79:)(+)(:235:72:79:)(+)(:236:72:79:)(+)(:237:72:79:)(+)(:238:72:79:)(+)(:239:72:79:)(+)(:240:72:79:)(+)(:241:72:79:)(+)(:242:72:79:)(+)(:243:72:79:)(+)(:244:72:79:)(+)(:245:72:79:)(+)(:246:72:79:)(+)(:247:72:79:)(+)(:248:72:79:)(+)(:249:72:79:)(+)(:250:72:79:)(+)(:251:72:79:)(+)(:252:72:79:)(+)(:253:72:79:)(+)(:254:72:79:)(+)(:255:72:79:)(+)(:256:72:79:)(+)(:257:72:79:)(+)(:258:72:79:)(+)(:259:72:79:)(+)(:260:72:79:)(+)(:261:72:79:)(+)(:262:72:79:)(+)(:263:72:79:)(+)(:264:72:79:)(+)(:265:72:79:)(+)(:266:72:79:)(+)(:267:72:79:)(+)(:268:72:79:)(+)(:269:72:79:)(+)(:270:72:79:)(+)(:271:72:79:)(+)(:272:72:79:)(+)(:273:72:79:)(+)(:274:72:79:)(+)(:275:72:79:)(+)(:276:72:79:)(+)(:277:72:79:)(+)(:278:72:79:)(+)(:279:72:79:)(+)(:280:72:79:)(+)(:281:72:79:)(+)(:282:72:79:)(+)(:283:72:79:)(+)(:284:72:79:)(+)(:285:72:79:)(+)(:286:72:79:)(+)(:287:72:79:)(+)(:288:72:79:)(+)(:289:72:79:)(+)(:290:72:79:)(+)(:291:72:79:)(+)(:292:72:79:)(+)(:293:72:79:)(+)(:294:72:79:)(+)(:295:72:79:)(+)(:296:72:79:)(+)(:297:72:79:)(+)(:298:72:79:)(+)(:299:72:79:)(+)(:300:72:79:)(+)(:301:72:79:)(+)(:302:72:79:)(+)(:303:72:79:)(+)(:304:72:79:)(+)(:305:72:79:)(+)(:306:72:79:)(+)(:307:72:79:)(+)(:308:72:79:)(+)(:309:72:79:)(+)(:310:72:79:)(+)(:311:72:79:)(+)(:312:72:79:)(+)(:313:72:79:)(+)(:314:72:79:)(+)(:315:72:79:)(+)(:316:72:79:)(+)(:317:72:79:)(+)(:318:72:79:)(+)(:319:72:79:)(+)(:320:72:79:)(+)(:321:72:79:)(+)(:322:72:79:)(+)(:323:72:79:)(+)(:324:72:79:)(+)(:325:72:79:)(+)(:326:72:79:)(+)(:327:72:79:)(+)(:328:72:79:)(+)(:329:72:79:)(+)(:330:72:79:)(+)(:331:72:79:)(+)(:332:72:79:)(+)(:333:72:79:)(+)(:334:72:79:)(+)(:335:72:79:)(+)(:336:72:79:)(+)(:337:72:79:)(+)(:338:72:79:)(+)(:339:72:79:)(+)(:340:72:79:)(+)(:341:72:79:)(+)(:342:72:79:)(+)(:343:72:79:)(+)(:344:72:79:)(+)(:345:72:79:)(+)(:346:72:79:)(+)(:347:72:79:)(+)(:348:72:79:)(+)(:349:72:79:)(+)(:350:72:79:)(+)(:351:72:79:)(+)(:352:72:79:)(+)(:353:72:79:)(+)(:354:72:79:)(+)(:355:72:79:)(+)(:356:72:79:)(+)(:357:72:79:)(+)(:358:72:79:)(+)(:359:72:79:)(+)(:360:72:79:)(+)(:361:72:79:)(+)(:362:72:79:)(+)(:363:72:79:)(+)(:364:72:79:)(+)(:365:72:79:)(+)(:366:72:79:)(+)(:367:72:79:)(+)(:368:72:79:)(+)(:369:72:79:)(+)(:370:72:79:)(+)(:371:72:79:)(+)(:372:72:79:)(+)(:373:72:79:)(+)(:374:72:79:)(+)(:375:72:79:)(+)(:376:72:79:)(+)(:377:72:79:)(+)(:378:72:79:)(+)(:379:72:79:)(+)(:380:72:79:)(+)(:381:72:79:)(+)(:382:72:79:)(+)(:383:72:79:)(+)(:384:72:79:)(+)(:385:72:79:)(+)(:386:72:79:)(+)(:387:72:79:)(+)(:388:72:79:)(+)(:389:72:79:)(+)(:390:72:79:)(+)(:391:72:79:)(+)(:392:72:79:)(+)(:393:72:79:)(+)(:394:72:79:)(+)(:395:72:79:)(+)(:396:72:79:)(+)(:397:72:79:)(+)(:398:72:79:)(+)(:399:72:79:)(+)(:400:72:79:)(+)(:401:72:79:)(+)(:402:72:79:)(+)(:403:72:79:)(+)(:404:72:79:)(+)(:405:72:79:)(+)(:406:72:79:)(+)(:407:72:79:)(+)(:408:72:79:)(+)(:409:72:79:)(+)(:410:72:79:)(+)(:411:72:79:)(+)(:412:72:79:)(+)(:413:72:79:)(+)(:414:72:79:)(+)(:415:72:79:)(+)(:416:72:79:)(+)(:417:72:79:)(+)(:418:72:79:)(+)(:419:72:79:)(+)(:420:72:79:)(+)(:421:72:79:)(+)(:422:72:79:)(+)(:423:72:79:)(+)(:424:72:79:)(+)(:425:72:79:)(+)(:426:72:79:)(+)(:427:72:79:)(+)(:428:72:79:)(+)(:429:72:79:)(+)(:430:72:79:)(+)(:431:72:79:)(+)(:432:72:79:)(+)(:433:72:79:)(+)(:434:72:79:)(+)(:435:72:79:)(+)(:436:72:79:)(+)(:437:72:79:)(+)(:438:72:79:)(+)(:439:72:79:)(+)(:440:72:79:)(+)(:441:72:79:)(+)(:442:72:79:)(+)(:443:72:79:)(+)(:444:72:79:)(+)(:445:72:79:)(+)(:446:72:79:)(+)(:447:72:79:)(+)(:448:72:79:)(+)(:449:72:79:)(+)(:450:72:79:)(+)(:451:72:79:)(+)(:452:72:79:)(+)(:453:72:79:)(+)(:454:72:79:)(+)(:455:72:79:)(+)(:456:72:79:)(+)(:457:72:79:)(+)(:458:72:79:)(+)(:459:72:79:)(+)(:460:72:79:)(+)(:461:72:79:)(+)(:462:72:79:)(+)(:463:72:79:)(+)(:464:72:79:)(+)(:465:72:79:)(+)(:466:72:79:)(+)(:467:72:79:)(+)(:468:72:79:)(+)(:469:72:79:)(+)(:470:72:79:)(+)(:471:72:79:)(+)(:472:72:79:)(+)(:473:72:79:)(+)(:474:72:79:)(+)(:475:72:79:)(+)(:476:72:79:)(+)(:477:72:79:)(+)(:478:72:79:)(+)(:479:72:79:)(+)(:480:72:79:)(+)(:481:72:79:)(+)(:482:72:79:)(+)(:483:72:79:)(+)(:484:72:79:)(+)(:485:72:79:)(+)(:486:72:79:)(+)(:487:72:79:)(+)(:488:72:79:)(+)(:489:72:79:)(+)(:490:72:79:)(+)(:491:72:79:)(+)(:492:72:79:)(+)(:493:72:79:)(+)(:494:72:79:)(+)(:495:72:79:)(+)(:496:72:79:)(+)(:497:72:79:)(+)(:498:72:79:)(+)(:499:72:79:)(+)(:500:72:79:)(+)(:501:72:79:)(+)(:502:72:79:)(+)(:503:72:79:)(+)(:504:72:79:)(+)(:505:72:79:)(+)(:506:72:79:)(+)(:507:72:79:)(+)(:508:72:79:)(+)(:509:72:79:)(+)(:510:72:79:)(+)(:511:72:79:)(+)(:512:72:79:)(+)(:513:72:79:)(+)(:514:72:79:)(+)(:515:72:79:)(+)(:516:72:79:)(+)(:517:72:79:)(+)(:518:72:79:)(+)(:519:72:79:)(+)(:520:72:79:)(+)(:521:72:79:)(+)(:522:72:79:)(+)(:523:72:79:)(+)(:524:72:79:)(+)(:525:72:79:)(+)(:526:72:79:)(+)(:527:72:79:)(+)(:528:72:79:)(+)(:529:72:79:)(+)(:530:72:79:)(+)(:531:72:79:)(+)(:532:72:79:)(+)(:533:72:79:)(+)(:534:72:79:)(+)(:535:72:79:)(+)(:536:72:79:)(+)(:537:72:79:)(+)(:538:72:79:)(+)(:539:72:79:)(+)(:540:72:79:)(+)(:541:72:79:)(+)(:542:72:79:)(+)(:543:72:79:)(+)(:544:72:79:)(+)(:545:72:79:)(+)(:546:72:79:)(+)(:547:72:79:)(+)(:548:72:79:)(+)(:549:72:79:)(+)(:550:72:79:)(+)(:551:72:79:)(+)(:552:72:79:)(+)(:553:72:79:)(+)(:554:72:79:)(+)(:555:72:79:)(+)(:556:72:79:)(+)(:557:72:79:)(+)(:558:72:79:)(+)(:559:72:79:)(+)(:560:72:79:)(+)(:561:72:79:)(+)(:562:72:79:)(+)(:563:72:79:)(+)(:564:72:79:)(+)(:565:72:79:)(+)(:566:72:79:)(+)(:567:72:79:)(+)(:568:72:79:)(+)(:569:72:79:)(+)(:570:72:79:)(+)(:571:72:79:)(+)(:572:72:79:)(+)(:573:72:79:)(+)(:574:72:79:)(+)(:575:72:79:)(+)(:576:72:79:)(+)(:577:72:79:)(+)(:578:72:79:)(+)(:579:72:79:)(+)(:580:72:79:)(+)(:581:72:79:)(+)(:582:72:79:)(+)(:583:72:79:)(+)(:584:72:79:)(+)(:585:72:79:)(+)(:586:72:79:)(+)(:587:72:79:)(+)(:588:72:79:)(+)(:589:72:79:)(+)(:590:72:79:)(+)(:591:72:79:)(+)(:592:72:79:)(+)(:593:72:79:)(+)(:594:72:79:)(+)(:595:72:79:)(+)(:596:72:79:)(+)(:597:72:79:)(+)(:598:72:79:)(+)(:599:72:79:)(+)(:600:72:79:)(+)(:601:72:79:)(+)(:602:72:79:)(+)(:603:72:79:)(+)(:604:72:79:)(+)(:605:72:79:)(+)(:606:72:79:)(+)(:607:72:79:)(+)(:608:72:79:)(+)(:609:72:79:)(+)(:610:72:79:)(+)(:611:72:79:)(+)(:612:72:79:)(+)(:613:72:79:)(+)(:614:72:79:)(+)(:615:72:79:)(+)(:616:72:79:)(+)(:617:72:79:)(+)(:618:72:79:)(+)(:619:72:79:)(+)(:620:72:79:)(+)(:621:72:79:)(+)(:622:72:79:)(+)(:623:72:79:)(+)(:624:72:79:)(+)(:625:72:79:)(+)(:626:72:79:)(+)(:627:72:79:)(+)(:628:72:79:)(+)(:629:72:79:)(+)(:630:72:79:)(+)(:631:72:79:)(+)(:632:72:79:)(+)(:633:72:79:)(+)(:634:72:79:)(+)(:635:72:79:)(+)(:636:72:79:)(+)(:637:72:79:)(+)(:638:72:79:)(+)(:639:72:79:)(+)(:640:72:79:)(+)(:641:72:79:)(+)(:642:72:79:)(+)(:643:72:79:)(+)(:644:72:79:)(+)(:645:72:79:)(+)(:646:72:79:)(+)(:647:72:79:)(+)(:648:72:79:)(+)(:649:72:79:)(+)(:650:72:79:)(+)(:651:72:79:)(+)(:652:72:79:)(+)(:653:72:79:)(+)(:654:72:79:)(+)(:655:72:79:)(+)(:656:72:79:)(+)(:657:72:79:)(+)(:658:72:79:)(+)(:659:72:79:)(+)(:660:72:79:)(+)(:661:72:79:)(+)(:662:72:79:)(+)(:663:72:79:)(+)(:664:72:79:)(+)(:665:72:79:)(+)(:666:72:79:)(+)(:667:72:79:)(+)(:668:72:79:)(+)(:669:72:79:)(+)(:670:72:79:)(+)(:671:72:79:)(+)(:672:72:79:)(+)(:673:72:79:)(+)(:674:72:79:)(+)(:675:72:79:)(+)(:676:72:79:)(+)(:677:72:79:)(+)(:678:72:79:)(+)(:679:72:79:)(+)(:680:72:79:)(+)(:681:72:79:)(+)(:682:72:79:)(+)(:683:72:79:)(+)(:684:72:79:)(+)(:685:72:79:)(+)(:686:72:79:)(+)(:687:72:79:)(+)(:688:72:79:)(+)(:689:72:79:)(+)(:690:72:79:)(+)(:691:72:79:)(+)(:692:72:79:)(+)(:693:72:79:)(+)(:694:72:79:)(+)(:695:72:79:)(+)(:696:72:79:)(+)(:697:72:79:)(+)(:698:72:79:)(+)(:699:72:79:)(+)(:700:72:79:)(+)(:701:72:79:)(+)(:702:72:79:)(+)(:703:72:79:)(+)(:704:72:79:)(+)(:705:72:79:)(+)(:706:72:79:)(+)(:707:72:79:)(+)(:708:72:79:)(+)(:709:72:79:)(+)(:710:72:79:)(+)(:711:72:79:)(+)(:712:72:79:)(+)(:713:72:79:)(+)(:714:72:79:)(+)(:715:72:79:)(+)(:716:72:79:)(+)(:717:72:79:)(+)(:718:72:79:)(+)(:719:72:79:)(+)(:720:72:79:)(+)(:721:72:79:)(+)(:722:72:79:)(+)(:723:72:79:)(+)(:724:72:79:)(+)(:725:72:79:)(+)(:726:72:79:)(+)(:727:72:79:)(+)(:728:72:79:)(+)(:729:72:79:)(+)(:730:72:79:)(+)(:731:72:79:)(+)(:732:72:79:)(+)(:733:72:79:)(+)(:734:72:79:)(+)(:735:72:79:)(+)(:736:72:79:)(+)(:737:72:79:)(+)(:73
```

KOMPUTAR WORKS

Figure 2.

P.O.Box 483
Electric City
Washington
99123

(509) 633-2653

SALES ORDER ACKNOWLEDGEMENT

Date- APRIL 1 1981

S.O.number- 0001

Sold to- MR. CHIP BUGGS
APT. # 102
9900 SEAMOSS AVE.
ELECTRIC CITY
WASHINGTON 99123

Ship to- SAME
SAME

Ship via- U.P.S. BLUE LABEL

Stock #	Description	Scheduled ship date	Quantity	Unit price	Amount
PHC004	!TI 99/4 HOME COMPUTER CONSOLE	!05-2-81	1	!499.00	499.00
PHA2000	!DUAL CASSETTE CABLE	!05-2-81	1	!11.00	11.00
PHA4100	!10" COLOR MONITOR	!05-2-81	1	!327.98	327.98
PHA2605	!BLANK OVERLAYS (4 PACK)	!04-2-81	2	!5.45	10.90

SUBTOTAL = 859.88
TAX = 46.28
FREIGHT = 00.00
TOTAL = \$ 897.16

Thank you for the order. Remember "word of mouth" advertising keeps our costs down.... So help spread the word!

is the whole FORM DATA FILE produced by my "editor" program for my Sales Order Acknowledgement form. Figure 2 shows the results of running IFG with the Sales Order Acknowledgement FORM DATA FILE. The above repeat sequence is from lines 60-62 of that Sales Order Acknowledgement FORM DATA FILE.

Finally, Listing 6 is the INTERACTIVE FORMS GENERATOR program. I recommend loading it without all the comment lines to save memory. If you use the disk drive system, you should use CALL FILES(1) and NEW prior to loading IFG. That will give space for about a 70-line FORM DATA FILE.

If you have problems or questions with the IFG please write me in care of 99'er Magazine. And assuming everybody

doesn't swamp me with letters, I will get right back to you. Good luck and have fun!

The "IFG" program is setup for use with an EPSON MX-80 printer connected as device:

"RS232/2,DA=8,BA=9600"

If you are connected to the other port or using a different baud rate, the OPEN statements for the printer on line number 2010 must be changed.

You can use a different "RS232" printer with the "IFG" program but first check line numbers 190-360 to make sure these character sequences are compatible with your printer. Especially check RESETEPSON that initializes the printer.

```

100 REM *****
110 REM * INTERACTIVE FORMS GENERATOR *
120 REM *****
130 REM 99'ER VERSION 9.81.1
140 REM BY DAVID G. BRADER
150 REM
160 REM
170 OPTION BASE 1
180 DIM A$(70)
190 RESETEPSON$=CHR$(18)&CHR$(20)&CHR$(27)
    &CHR$(70)&CHR$(27)&CHR$(72)&CHR$(13)
200 BLANK$=""
210 QUOTE$=CHR$(34)
220 BANG$=CHR$(33)&CHR$(33)
230 COLON$=CHR$(58)
240 SEMICOLON$=CHR$(59)
250 RIGHTARROW$=CHR$(62)
260 AMPERSAND$=CHR$(64)
270 OPENPAREN$=CHR$(40)
280 CLOSEPAREN$=CHR$(41)
290 DECIMAL$=CHR$(46)
300 ZERO$=CHR$(48)
310 PLUS$=CHR$(43)
320 MINUS$=CHR$(45)
330 MULTIPLY$=CHR$(42)
340 DIVIDE$=CHR$(47)
350 EQUAL$=CHR$(61)
360 SPACE$=CHR$(32)
    
```

Continued on p. 38

Listing 6.

```

THIS PROGRAM SCANS THE FILE FOR LINES THAT START WITH !! . THEN THESE "INTERACTIVE COMMAND LINES" ARE PARSED FOR FOUR TYPES OF COMMANDS. 1. "COMMENTS" OR MESSAGES TO PROMPT THE INTERACTIVE USER. THIS TYPE OF COMMAND IS IN THE FORM OF TEXT PRECEDED BY A QUOTE AND FOLLOWED BY A QUOTE. 2. "FIELD-DEFINITION" TYPE COMMANDS DEFINE THE PHYSICAL FIELD INTO WHICH THE USER'S KEYBOARD INPUT WILL BE STORED. IT HAS THE FORM--

!<LINE NUMBER>:<START POSITION><END POSITION>:
EXAMPLE-- !23:5:22:

A SAMPLE COMMAND LINE IS:
!!"ENTER THE SERIAL NUMBER-" !19:7:22:

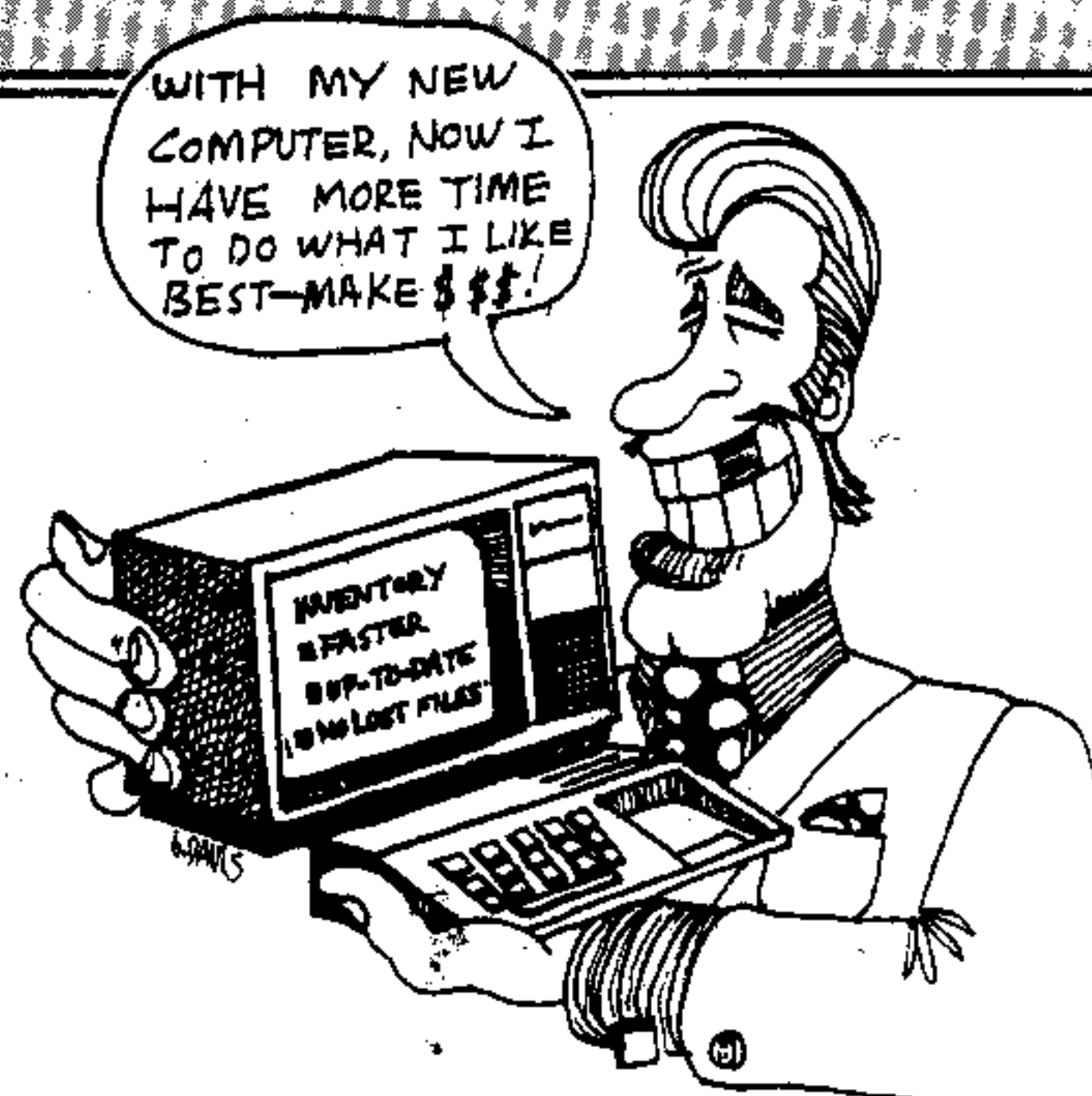
3. "REPEAT COMMAND SEQUENCE" START WITH- !!<NUMERIC VALUE>- AND MUST END WITH A @ . EVERYTHING IN BETWEEN WILL BE REPEATED THE NUMBER OF TIMES SPECIFIED BY THE NUMERIC VALUE. A SAMPLE MIGHT BE:

(LINE 20) SERIAL NUMBER-          MODEL-
(LINE 21) SERIAL NUMBER-          MODEL-
(LINE 22) SERIAL NUMBER-          MODEL-
(LINE 77) !!FILL IN THE TABLE VALUES THAT FOLLOW:
(LINE7+1) !!@3:"ENTER SERIAL NUMBER:" !20:15:24:
(LINE7+2) !!"ENTER THE MODEL NUMBER:" !20:31:40: @

4. "MATH TRANSFORMATIONS" ARE MADE UP OF TERMS AND OPERATORS. TERMS MAY BE "FIELD-DEFINITION" OR CONSTANT TYPES. OPERATORS ARE "*", "/", "+", "=", AND "-". ALL TERMS AND OPERATORS MUST EACH BE ENCLOSED IN PARENTHESES.
EXAMPLE:

!!(!23:5:22:) (8) (.0544) (=) (!>23:17:35:)

NOTE THE ">" IN THE LAST TERM WHICH CAUSES THE ANSWER TO BE RIGHT JUSTIFIED IN THE FIELD.
    
```



Getting Down to Business

By George Struble

In the last issue I promised that this time around we'd look at some commercial software packages. In keeping with that promise, this article will review two very different pieces of software. Since my general advice on evaluation criteria in the last issue was necessarily abstract, the present analyses should be useful to you as examples of how you might go about evaluating a piece of software. So here goes.



General Ledger

TI Extended BASIC, 16K; Disk, \$99.95

Futura Software
Ehninger Associates, Inc.
P. O. Box 5581
Fort Worth, Texas 76108

The first package we'll look at is Futura Software's General Ledger system, available from Ehninger Associates.

Page 1 of the documentation contains (1) a short statement of purpose, (2) an indication of size limitations (approximately 900 entries—i.e., chart entries plus general ledger entries—per month), and (3) the equipment configuration needed by the system. This is exactly what you should want to know

About the Author

George Struble, a professor of computer and information science at the University of Oregon, is author of *Business Information Processing with Basic*, Addison-Wesley Publishing Co., 1980.

on page 1. The required configuration is as follows:

- TI-99/4 Console
- TI Extended Basic Command Module
- Color Monitor
- PHP 1800 Disk Drive Controller
- Two PHP 1850 Disk Memory Drives
- PHP 1700 RS232 Interface
- TI Omni 800 Model 820 terminal printer with compressed print option (10 and 16.5 char/in) or equivalent
- A box of minidisks

An Overview of System Design

The Futura system implements standard double-entry bookkeeping, and produces the full range of standard reports: journal proof and posting reports, trial balance, plus balance sheet and income statements. And the software lets you define your individual "chart of accounts"—the set of accounts lines you want in these reports. There are, however, some mandatory accounts.

I tested the system by creating a skeletal chart of accounts, entering a few transactions into the journal, and going through the proofing/posting/month-end process. Some entries were made wrong on purpose, so that validity checks and error correction procedures, if provided, could be inspected.

The first thing that struck me about the software was that this General Ledger system was indeed well designed: The hierarchy of menus made sense naturally, and the functions that a user would want to perform were provided in a logical way. In particular, information that I needed to see on the screen in order to perform an intelligent action was usually there. (Note: It is not easy for a designer to achieve this objective, since the screen does not display a great deal of information at a time. Furthermore, cramming the screen makes it hard to find the information you want anyway. So it takes rather sensitive judgement for a system designer to show what users want. Unfortunately, some system de-

signers seem not to consider the problem at all . . .)

Another aspect of good design is that careful attention is paid to error correction. A few centuries of bookkeeping practice have developed some standards for how errors should be corrected. These standards are adhered to in the Futura package. For example, *after* a general ledger entry is "posted," any corrections must be made by entering an *adjusting* entry; but *before* an entry is posted, corrections can be made *directly* to the entry. And good facilities are provided for doing this as well.

The documentation is well written; it is clear and neatly presented. The directions were reasonably unambiguous, and I was able to follow them without getting lost. (The good writers among you know how difficult it is to write clear, unambiguous instructions.) The things I needed to know were there, and organized so I could find them again. Later, I will, however, make some suggestions for possible improvements in the documentation. These may suggest points for you to look for when evaluating any system.

Performance

The Futura General Ledger system does indeed work: It does what it says it will, and it does what it should (not necessarily the same thing). I did find one bug, however: When I had no current liability detail accounts in my chart of accounts, the system *made up* liabilities to match my assets. This is certainly an out-of-the-way bug—one that a user shouldn't ever encounter (except on a practice run to gain familiarity with the system, prior to actual use). In real life, I can hardly imagine anyone's actual chart of accounts *not* including any current liabilities. [A telephone call to Mr. Ehninger, the program's author, resulted in an immediate correction to this minor defect—Ed.]

In regard to speed, the system is *not* instantaneous since the Extended BASIC

programs are chained: When you go from one menu function to another, a program must be loaded from your program disk, and it is likely that data must be read from your data disk too. But *during* a particular function, however, the operation is prompt. Data entry is perhaps the major place where you want fast responsiveness, and the system performs well here.

Required Level of User Knowledge

To use this system correctly, some accounting knowledge is needed. You must know what are "debits" and what are "credits," and how to create balancing entries for each transaction. You must also be able to analyze an out-of-balance trial balance and create correct adjusting entries. The system provides the information you need and the facilities to make your actions, but you must know *how* to use them!

This brings me back to the documentation. The very sensible recommendation made in the documentation is that you practice with a few journal entries and reports before you start entering your actual data. I believe that this is absolutely necessary. If for no other reason, you must learn what must be entered as positive and what as negative. For example, one of my practice transactions was a sale of \$500 whose proceeds were directly deposited in my bank account. It turned out that the \$500 bank deposit had to be entered as a *positive* number, but the sale as a *negative* number. No doubt, if I knew more about accounting, this would have been obvious from the start, but I would wager that some of you would "discover" it during practice as I did. Some advice on points like this would be a useful addition to the documentation.

Some of us like to read a manual or program documentation first, assimilate and understand the information, then try it out. The documentation for this system is not sufficient for this kind of "armchair" understanding. Several additions would have been helpful to me. First, a chart showing the hierarchy of menus would have shown the structure of the system in an easily understandable form. Second, pictures of some of the screen displays would have been quite informative—especially the data entry screens; they would have told me what was expected. As a part of this, some examples of data entry also would have helped me to relate data entry, the actions of the system, and the reports produced. The documentation does include nice copies of all sample reports, but an explanation of the data entered to achieve those reports would have helped to tie everything together. [My personal preference is for documentation that is so loaded with examples and screen displays that a computer is not needed *initially* to obtain a good grasp and understanding of how a system is supposed

to operate. Serious software developers would do well to study the *format* of Kenneth Bowles' *Beginners Guide for the UCSD Pascal System*. For that matter, programmers interested in purchasing TI's Pascal Development System can get a real "feel" for it by studying the *content* of the Bowles' book. See Mr. Berkey's related Pascal article in this issue—Ed.]

System Summary

In summary, the system is well designed and well implemented; it should be graciously usable by one who is familiar with bookkeeping procedures. The documentation is well written, and though my personal taste would ask for selected additions, the necessary information to permit me to complete my understanding with "hands-on" practice

is certainly present. I can highly recommend the system.

Additional Linked Packages

We have also received a preliminary version of Futura's Accounts Payable system that can be used interactively with the General Ledger. Some initial testing revealed a few operational problems in this early release. At the time of this writing, the developer was at work correcting them. The system design did appear to be quite good—all the functions I could think of were included, with some very nice usability features I did not expect. You will undoubtedly be hearing more about this system [and other linked record systems in Futura's expanding line of business/professional software—Ed.] in a forthcoming issue.



The Project Boss

TI BASIC, 16K; Disk \$49.95

Charles Mann & Associates
7594 San Remo Trail
Yucca Valley, CA 92284

The second piece of software reviewed for this issue is *The Project Boss*, available from Charles Mann & Associates.

To quote from the System Overview:

The Project Boss is a simple management tool to use in the financial control of any small business or personal project. The system allows you to estimate costs for a project budget or project bid, keep an accurate running total of actual costs for the project and periodically estimate the amount of additional financial effort that will be required to complete the project.

The documentation provided describes rather clearly and accurately the operation of the software, and clearly lists the required equipment—i.e., TI-99/4 console with 16K memory (resident TI BASIC), one disk drive and controller, plus the TI Thermal Printer or TI RS232 Interface and compatible printer if a printed copy is required.

Testing of the software revealed that the program worked—all except for printing on the thermal printer. (An easy-to-correct logic bug was discovered). Overall, the software did what is described in the documentation. Not every package can meet these performance criteria. And yet, this is *not* a satisfying package to use—because of a number of shortcomings in the "human engineering." Let us examine some of them with the hope that perhaps this software will be improved; perhaps the analysis will help you evaluate other software; and perhaps we can help other software designers pay more attention to human engineering as they build their systems.

1. Each project is broken into twenty budget categories. You can re-label the categories, but you will *always* have exactly twenty of them. You cannot delete or add any. Furthermore, any that you are not using are listed in every report anyway, and the screen functions that review or change the amounts for each category, in turn, always take you past those you may not be using. This takes up a user's time in a way that appears silly and wasteful. A design that causes user frustration isn't very good human engineering.

2. This leads to a related shortcoming: The function of displaying ("reviewing") budgets and costs on the screen is *only* available for each of the 20 categories *in turn*. There is no way to select one category for review, or to escape from the review once you have seen what you wanted. Most other functions permit the user to specify a particular category; this one should too.

3. On the display screen, the entry of numbers immediately follows the prompting messages; there's no space or other punctuation in between to aid readability.

4. The report format is rather unintelligent; it is limited in width so it can be used on the little 32-column thermal

GAMES & SIMULATIONS by FUTURA

ALL*STAR BASEBALL*	\$19.95
Pitching, fielding and base-running all under player control	
ALL*STAR BOWLING**	\$19.95
Tournament bowling at its best for up to eight contestants	
CHALLENGE I	\$16.95
Leaping frogs in two mind-challenging games	
CHALLENGE II	\$16.95
Nim and Tic-Tac-Toe with a new twist	
'CHUTES & SHARKS**	\$19.95
Rescue parachuters as they abandon ship over shark-infested ocean	
GALACTIC WAR**	\$19.95
Fast action space battle	
SAM DEFENSE	\$19.95
Command on authentic surface-to-air missile site	
WALL STREET*	\$19.95
Play the stock market and become a millionaire or a pauper	

*Also in Extended Basic
**Extended Basic only

ANNOUNCING A SERIES OF EDUCATIONAL PROGRAMS
Use the power your computer has to focus
***SIGHT *SOUND *ATTENTION *MOVEMENT**
To help your child learn to read.
Write for full details and availability.

HOUSEHOLD INVENTORY SYSTEM

\$34.95

With this award-winning program you can accurately inventory personal property, calculate actual replacement value, and maintain records on a current basis



BUSINESS SOFTWARE

General Ledger	\$99.95
Accounts Payable	\$99.95
Accounts Receivable	\$99.95
Payroll	\$99.95
Inventory Management	\$99.95
Word Processing	\$99.95
Mailing List	\$49.95

Fully interactive programs offer a totally integrated accounting system.

Loaded with innovative features:

- Completely menu-driven
- Random access of data
- Choice of retrieval of account records by name or number
- Fully prompted input
- Guaranteed to be bug-free and operator-proof!
- Hot-line and enhancement update service available.

Purchase units individually or the entire package for \$500.00.

Write for full details.

ORDERING INFORMATION: All orders shipped within 48 hours. Please enclose payment with order. Add \$1.00 for postage. (Add \$2.00 for postage on orders outside U.S.)

Ask for FUTURA Software at your local software dealer. Write for detailed description of these and other programs from FUTURA.

Dealer Inquiries Invited

FUTURA

S O F T W A R E

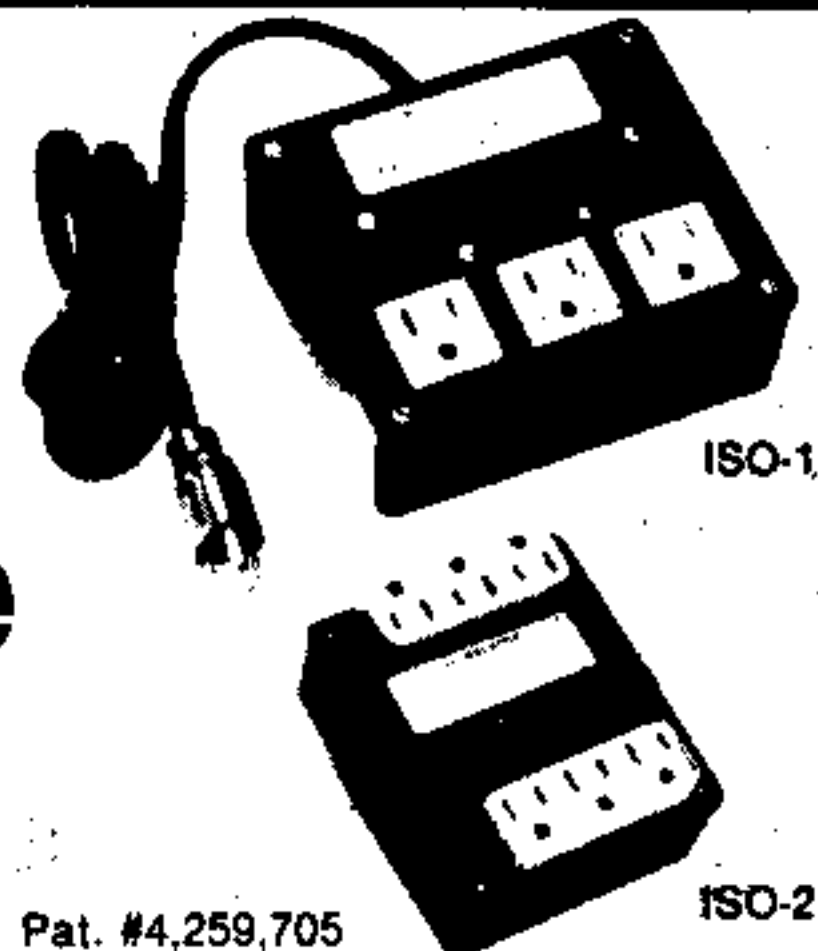


Ehninger Associates, Inc.

P.O. Box 5581
Fort Worth, Texas 76108
817-243-6536

**DISK DRIVE WOES?
PRINTER INTERACTION?
MEMORY LOSS?
ERRATIC OPERATION?**

Don't Blame The Software!



Power Line Spikes, Surges & Hash could be the culprit!

Pat. #4,259,705

Floppies, printers, memory & processor often interact! Our patented ISOLATORS eliminate equipment interaction AND curb damaging Power Line Spikes, Surges and Hash.

- ISOLATOR (ISO-1) 3 filter isolated 3-prong sockets; integral Surge/Spike Suppression; 1875 W Maximum load, 1 KW load any socket \$62.95
- ISOLATOR (ISO-2) 2 filter isolated 3-prong socket banks; (6 sockets total); Integral Spike/Surge Suppression; 1875 W Max load, 1-KW either bank \$62.95
- SUPER ISOLATOR (ISO-3), similar to ISO-1 except double filtering & Suppression \$94.95
- ISOLATOR (ISO-4), similar to ISO-1 except unit has 6 individually filtered sockets \$106.95
- SUPER ISOLATOR (ISO-11) similar to ISO-2 except double filtering & Suppression \$94.95
- CIRCUIT BREAKER, any model (add-CB) Add \$ 8.00
- CKT BRKR/SWITCH/PILOT (-CBS) Add \$16.00

AT YOUR DEALERS

Master-Card, Visa, American Express
Order Toll Free 1-800-225-4876
(except AK, HI, PR & Canada)

Electronic Specialists, Inc.

171 South Main Street, Natick, Mass. 01760
Technical & Non-800: 1-617-655-1532

E-X-T-E-N-D-E-R CABLES

Now Available
Connects TI-99/4 Computer
To Peripherals

- Shielded Cable
- Comes In 2 & 3 Foot Lengths
- Other Lengths Available on Request

2 ft. cable \$39.95
3 ft. cable \$42.95

Send Check or Money Order to:

UMI

P. O. BOX 4511
PITTSBURGH, PA 15205
Tel. 412-771-2863

Distributor or Dealer Inquiries Welcome

printer even though the system asks you to specify the bit/ baud parameters if you select the option for an RS232-compatible printer. The software doesn't provide a print format that can take advantage of more characters per line and print, for example, the budget, costs, variance, and completion cost in four columns. Furthermore, even in the single-column format, decimal points should have been lined up.

5. There are a few unclear instructions. For example, when I wanted to store my budget to disk, the system asked me which was my "MASTER." Is that the disk with the program on it? No, we were really being asked which disk the data should be stored on. So why not say so?

6. In the long run, what's perhaps most troubling is what the system *doesn't* do. Although it keeps track of total costs entered to date in each of the 20 categories, it *doesn't* print or keep any record of the detailed costs you have entered, so as to avoid double entry or overlooking an entry. And if *you* have to do that, is this software much more than a 20-accumulator adding machine?

7. I have saved the most amusing for last—primarily because it isn't serious. One of the functions you can choose is "INSTRUCTIONS." The system displays for you a condensed set of instructions for the software—a screenful at a time. The instructions are reasonably clearly written, but no attempt has been made to divide the screenfuls intelligently: The next screen often begins in the middle of a sentence! Moreover, there is one screen that starts "THIS IS REALLY IMPORTANT." What was important? It was on the previous screen. How can I get back? Only by stepping through *all* the rest of the instructions, then stepping through them all *again!!!*

Software Summary

I would not buy this software in its present form. Although the intended purpose for the product is a good idea, the product itself does too little, and does it too awkwardly. And yet, it has promise. Never underestimate the achievement of a piece of software that actually does what it claims . . .



Comments
and
Requests
Invited

Again, I invite your comments and requests. Do you disagree with something? Or at least think I should have done it better? What would you like me to do in future issues? I am much more likely to write something that is useful to you if you tell me what that could be.

Want to Get Published?

99'er Magazine is looking for articles in all areas of personal computing that concern the Texas Instruments TI-99/4 and other TMS9900-family systems (e.g., Marinchips, Technico, and the TM990/189). Here are the kinds of articles that we want you to write for us:

- Are you a businessman, professional, hobbyist, scientist, or engineer with an interesting microcomputer application? Tell us how it works, what problems you've had to overcome, and what recommendations you have for others. We're especially interested in sharing user-written software with our readers.
- Have you recently purchased a piece of hardware or software that hasn't quite come up to your expectations, or has, on the other hand, impressed you with its performance? We're looking for comprehensive product and book reviews from different perspectives.
- Are you an educator or parent with something to contribute to computer-assisted instruction (CAI)? We're always looking for new ideas and fresh approaches to educational problems.
- Have you created any unusual computer games or simulations? Let our readers experience your excitement and pleasure.
- Perhaps you've modified your microcomputer or have interfaced it with some unique or useful hardware. Send us your how-to-do-it story.

These are just some ideas. Perhaps you have others. Don't worry if you're not a professional writer. Our editorial staff stands ready to help polish up your manuscripts. And we'll be more than happy to send you a copy of our author's guidelines.

Send your manuscripts and correspondence to:

99'er Magazine / Editorial Dept.
2715 Terrace View Drive,
Eugene, Oregon 97405

Nine

SUPER DEALS FROM THE BACH COMPANY

1. Texas Instruments TI 99/4A Home Computer. Powerful TI Basic. Up to 72K total memory capacity. NEW typewriter keyboard. We carry *all* accessories. **\$369.95.**
2. Texas Instruments TI 59C. Programmable Calculator. With magnetic card storage. **\$189.95.** \$20 Rebate Program.
3. Texas Instruments TI 58C. Programmable Calculator. **\$89.95.**
4. Texas Instruments TI 100C. Printer/Plotter for TI 58C, TI 59C. **\$159.95.**
5. Texas Instruments TI Thermal Paper for 100C. 3 pack. **\$9.95.**
6. Hewlett-Packard Quad Memory Module. Converts HP-41C to HP-41CV capabilities. **\$89.95.**
7. Hewlett-Packard HP-41CV Alphanumeric Full Performance Programmable with Expandable Continuous Memory. **\$269.95.**
8. Hewlett-Packard HP-34C. Advanced Programmable Scientific Calculator with Continuous Memory. **\$119.95.**
9. Hewlett-Packard HP-38C. Advanced Financial Programmable Calculator with Continuous Memory. **\$119.95.**

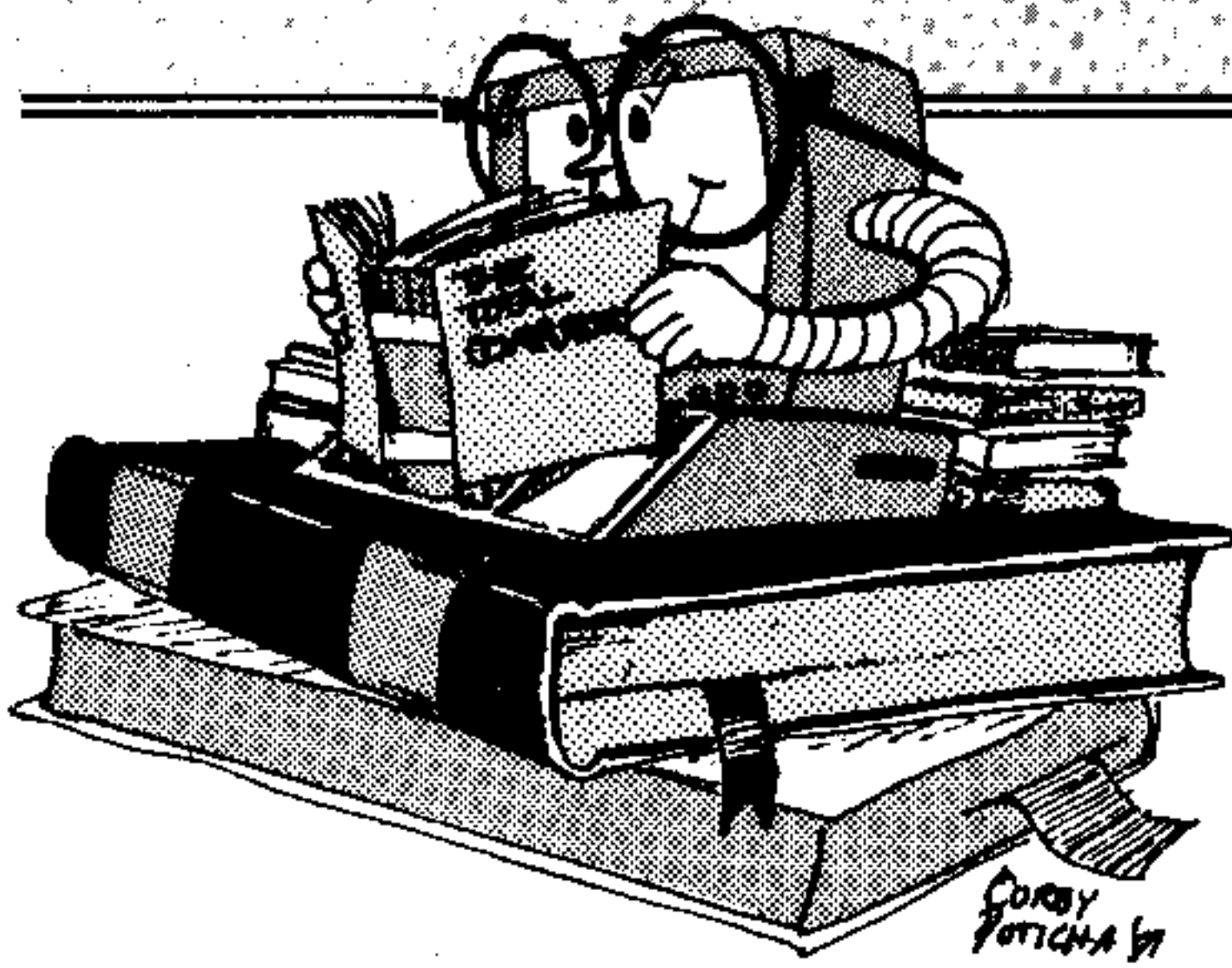
ORDER TOLL FREE
800-227-8292

California Residents call 415-494-1995. Use VISA, Mastercharge, Check or money order. Indicate card number and expiration date. Add \$2.50 for shipping.



The **BACH**
Company

P.O. Box 51178
Palo Alto, CA 94303



BASIC Computer Programs for the Home

By Charles D. Sternberg
(Hayden Books) paper, \$9.95

Practical BASIC Programs

Edited by Lon Poole
(Osborne/McGraw-Hill) paper, \$15.99

BASIC Computer Programs for Business (Vol. 1)

By Charles D. Sternberg
(Hayden Books) paper, \$10.95

By John Clulow

BASIC Computer Programs for the Home by Charles D. Sternberg (Hayden Books) is an excellent collection of 81 practical programs—each documented with a description, user instructions, and sample run. Program modifications are easily made since all programs are well documented internally with REM statements and an accompanying symbol table which describes each variable and function used.

The changes required in converting to TI BASIC are minimal. The MID\$ function must be replaced with SEG\$ which does the same thing. Occasionally, DIMension statements use variable names instead of numbers. In this case, you need only examine the initialization portion of the program and substitute the appropriate number for the variable name:

```
Given 100 Let N=20
      110 DIM A(N)
Change 110 DIM A(20)
```

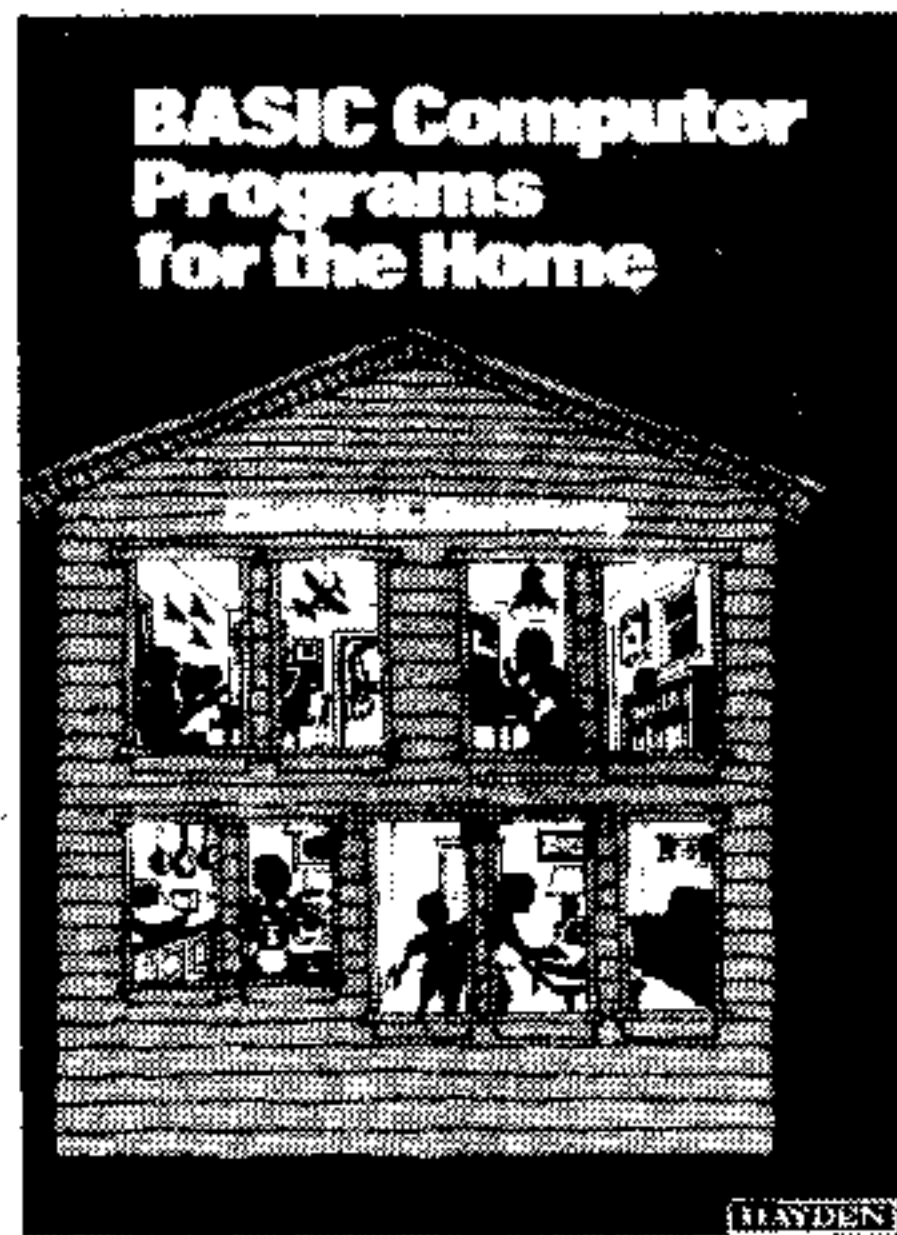
The OPTION BASE 1 statement should be included in the initialization portion of the program. Finally, PRINT statements sometimes require revision for an acceptable display with TI's 28 character screen line or appropriate modification for use with a printer.

No problems in file-handling conversion are encountered since none of the programs use external data files: In most cases, data are entered and retained in DATA statements. Although many of the programs in the book will prove useful "as is," converting them to accept external disk or tape cassette data files should enhance their utility considerably. The thorough internal documentation that is provided makes such changes easy to accomplish.

The only hardware required to RUN any program in the book is a console, monitor or TV, and cassette recorder for program storage. However, effective use of many of the programs does require a printer.

The programs are grouped in ten categories covering a wide range of home applications. The 11 **Home Financial Programs** include Household Budget and Household Expenses. The first program allows you to project a budget for up to six months; the second provides a monthly analysis of actual versus budgeted expenses.

If you are considering purchase of a car, **Automobile Comparisons**, one of 6 **Automobile Related Programs**, can help you take all operating costs into account and compare different models. If you already have a car, **Trip Planning** can help you prepare an itinerary, project daily trip costs, and compare various routes.



If you like to cook, you're sure to find a palatable program or two among the 9 **Kitchen Helpmates**. **Meal Planning**, for instance, prints a menu and categorized grocery checklist from your meal selections. And if you like to eat, there are a couple of diet programs to help you keep track of those calories.

With the 16 **List Programs for Every Purpose**, you can keep track of everything from addresses to 99'er Magazine articles, as well as coin, book, record, and even beer can collections. **Miscellaneous Programs for Home Use** includes

Utility Bill Analysis, Household Inventory, Weight Control, Paper Route, and seven other programs. The other program categories are **Tutorial Programs**, **Conversion Programs**, **Recreational Programs** and **Hobbyist Diaries**.

Practical BASIC Programs edited by Lon Poole (Osborne/McGraw-Hill), is by far the best single source of financial and technical programs I have seen. Thorough user documentation is provided for each of the 40 programs, replete with general description, examples, practice problems, and even references for further study.

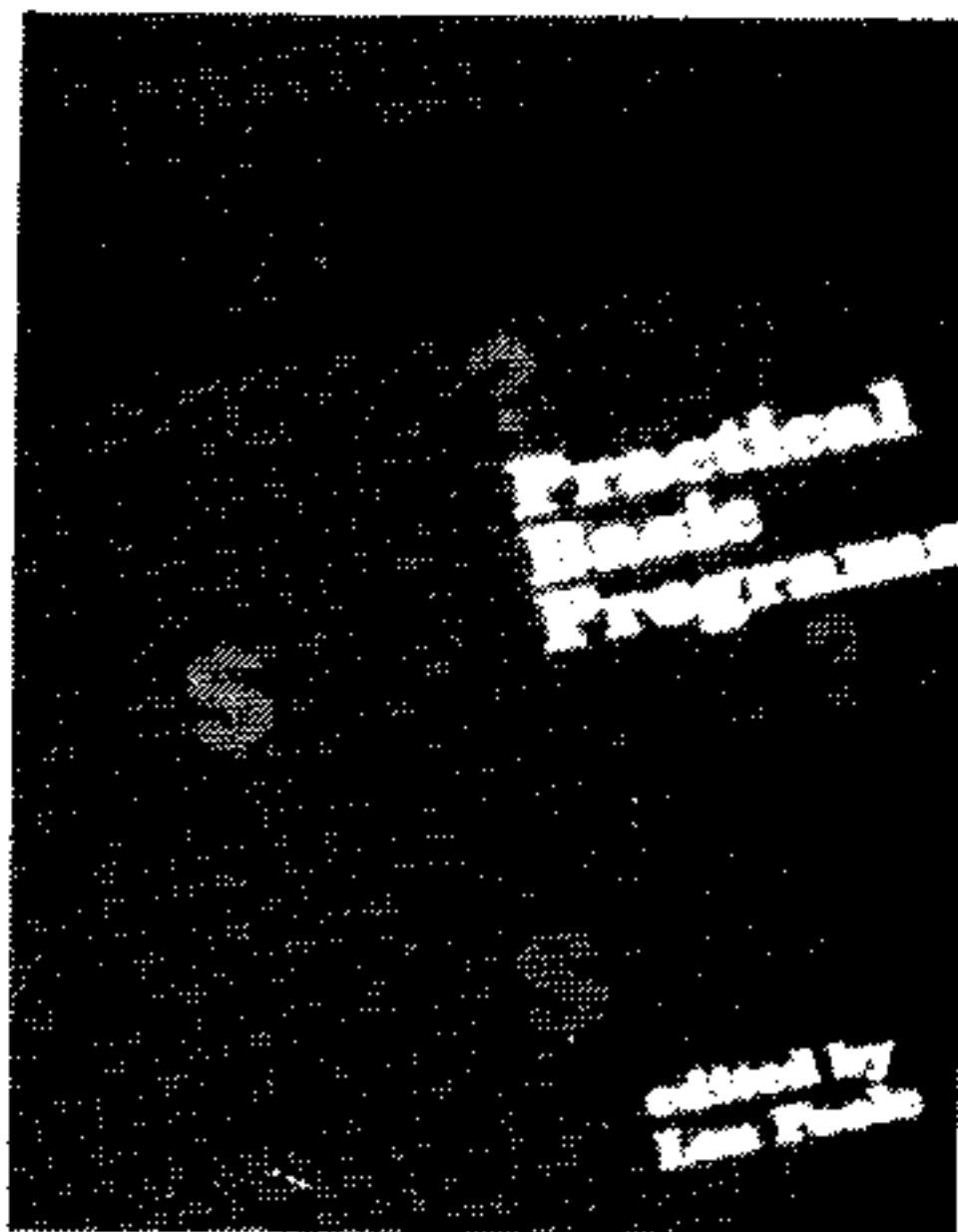
Conversion to TI BASIC presents no real problems. You must use SEG\$ to replace the MID\$ function, and "&" instead of "+" in string concatenation:

```
Given 100 A$=B$+C$
Change 100 A$=B$&C$
```

In some cases, PRINT statements require modification to fit the TI 28 character display line. In addition to required language conversions, the appendix provides a subroutine which can be used to suspend output when the screen is full.

The programs fall into four general areas of application: (1) **Financial**, (2) **Management Decision**, (3) **Statistics**, and (4) **Mathematics/Scientific**. Some of the financial programs are **Income Averaging** (form 1040 G), **Present Value of a Tax Deduction**, **Financial Statement Ratio Analysis**, **Checkbook Reconciliation**, and **Home Budgeting**. As an example of the quality of these programs, consider **Home Budgeting**: The user is first questioned extensively as to sources of income, outstanding loans, credit cards, and expense items. Cash flow is then projected on a monthly basis showing the day, date, and amount of each individual transaction, together with opening cash balance and a cash-in/cash-out summary.

The management decision programs cover a number of operations research models including Markov Analysis, Queuing Theory, PERT, Critical Path Method, Bayesian Decision Analysis, Economic Order and Production Quantities, Nonlinear Breakeven Analysis, and Payoff Matrix Analysis. I think these are called "management decision" programs because before you start typing them in you have to decide if you can manage to figure out what in the world they do . . .



If you want to analyze some data and need a good descriptive statistics program, *Statistics* provides 26 measures of central tendency, dispersion, skewness, kurtosis, and correlation for grouped and ungrouped data. There is also a program which provides an unbiased estimator of the standard deviation based on the gamma function, as well as good chi-square and confidence intervals programs.

The mathematics and scientific programs include Newtonian and Lagrangian Interpolation, Sums of Powers, Factorials, Numeric Base Conversions, and Temperature Conversions (Fahrenheit, Celsius, Reaumur, Kelvin, and Rankine). There is even a musical key transposition program! Any of these could be easily adapted for inclusion in a program you may be writing.

In summary, if you want to explore the area of practical applications for your computer, both of these books should prove most valuable; they are inexpensive and present virtually no software conversion problems.

In contrast to the previous two books, *BASIC Computer Programs for Business* (Vol. 1) by Charles D. Sternberg (Hayden Books) contains *integrated sets* of programs (business application systems) as well as individual business appli-

cation programs. The programs are grouped under the headings (1) **Financial Control and Analysis**, (2) **Inventory Control and Analysis**, and (3) **Production Planning and Control**.

They are intended for the typical business computer system which uses disk storage and a printer. Their modular design will allow selective application for specific business needs.

The documentation provided is outstanding. In addition to liberal use of REMark statements within a program and a symbol table which describes all major variable names and BASIC functions, the reader is given (a) a general description of the business process, (b) a description of the systems operation complete with flow charts, and (c) a detailed layout of all files used. One or more examples following each program allow you to observe its operation. This documentation will facilitate program extension and modification as well as construction of new programs from available program modules (since within each application area there is consistency with respect to use of variable names and processing methods).

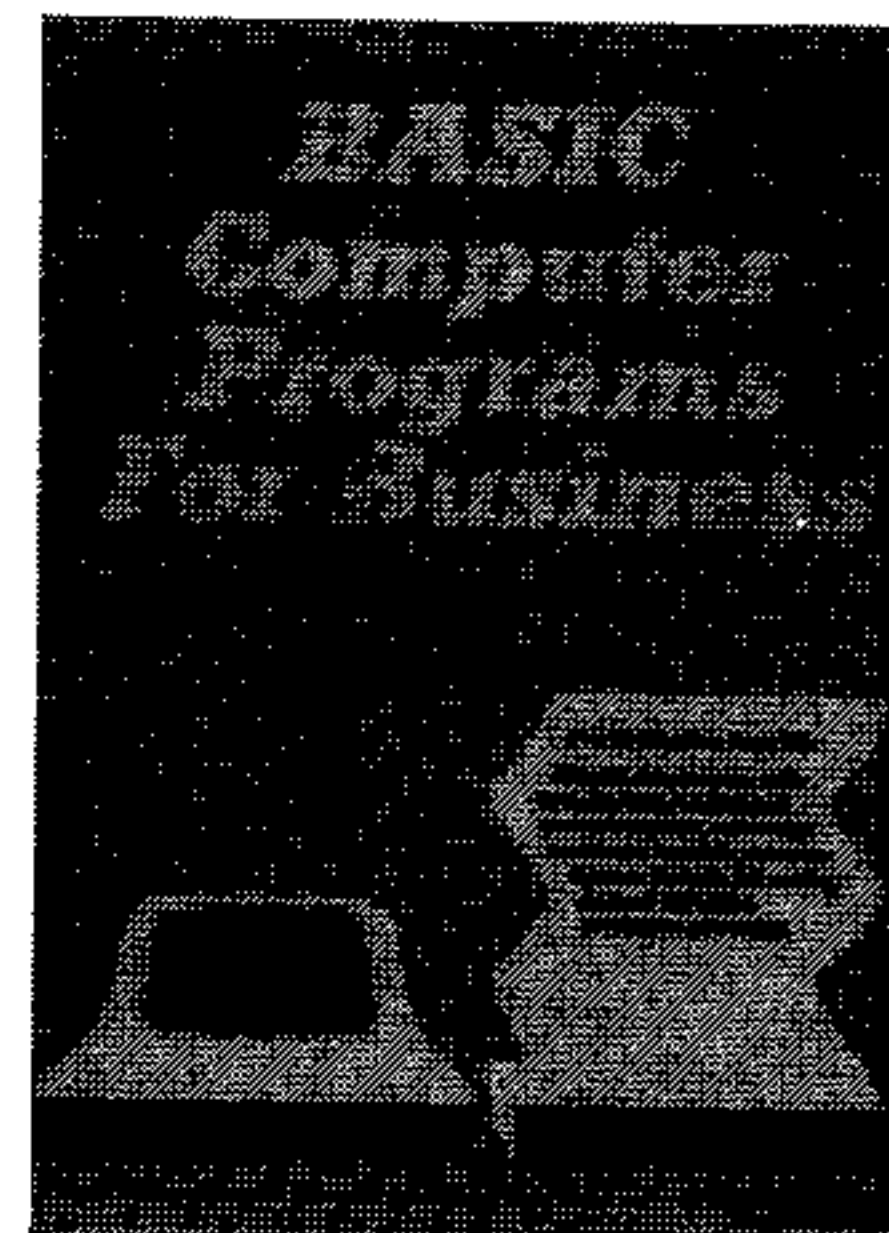
The financial programs include a bookkeeping system, an accounts receivable system, and 21 general financial analysis programs. The bookkeeping system is similar to a manual method: At the beginning of the accounting period, required accounts are determined and files initialized. Throughout the period as journal entries are posted to the file, account file contents can be listed, accounts displayed, account information corrected, and journal entries recreated as required. At the end of the accounting period the system will prepare a trial balance and income statement. After posting net income/loss to capital accounts, a balance sheet is prepared, closing journal entries are made, and accounts closed. The system uses one file for operation and contains 10 programs.

The accounts receivable system allows a user to add, change, and update accounts with payment and charge transactions. In addition, monthly accounts receivable reports are prepared together with monthly statements for customers. Additional capabilities include copying the file for recovery purposes and closing the file at the end of the accounting period with a summary report by account. One disk file is required for program operation.

The general financial programs include breakeven analysis, balance sheet preparation, cash flow and budget analysis, three types of forecasting, depreciation, amortization, return on investment, property and equipment comparisons, and several others.

Inventory control and analysis includes a perpetual inventory system, a periodic inventory system, and five inventory analysis programs. The perpetual inventory system allows a user to update the inventory file as each transaction is entered. The status of specific items can be determined in response to customer request. Monthly status reports summarize transactions in the inventory period. At any time during the period, new inventory items can be added, and data for existing items corrected.

The periodic inventory system allows monitoring and control of an inventory on a periodic basis (e.g., weekly, monthly, etc.). At the end of a period, a computer-printed recording log is used to take a physical inventory. Then, records of incoming stock and inventory-on-hand data are processed, files are updated, and a report for the period is printed. Optional reports (by location or class code) and an inventory analysis program (providing projected use and reorder point information) are also available. The system uses four files: an index file, an inventory master file, and two temporary transaction files. The general inventory programs include reorder point computation, inventory turnover analysis, inventory use projections, asset control/accounting and material locator.



Finally, the production planning and control category includes 10 programs. For assistance in product pricing, *JOB COST* accepts overhead, fixed, and variable costs to compute component and overall costs for various production quantities. A bill-of-materials program maintains a disk file with material requirements—including individual components and assemblies for several products. With the production scheduling program, an updated schedule file is maintained and can be reviewed to determine available times for using critical resources. Job routing maintains a file containing individual processes or tasks

Continued on p. 37

How To Write A BASIC Program That Writes BASIC Programs



PART 1:

A SURPRISING DISCOVERY WITH TI'S PROGRAMMING AIDS III

By John Clulow

Ti's *Programming Aids III* opens the door to some powerful programming techniques. The Cross Reference and Editor capabilities of this software will be appreciated by the serious Extended BASIC programmer. But the excitement really begins when you realize *how* this software does its thing.

PA III can provide (1) a tabular, line number cross reference for all variables, arrays, keywords, functions, and line number references in a program and (2) the ability to delete, move, or resequence specified groups of lines within a program much more quickly than could be done manually at the keyboard.

Required Hardware

Programming Aids III is a set of four Extended BASIC programs (LINPUT, CREF, CREFPRINT, and EDITOR) available on disk at a suggested retail price of \$19.95. In addition to a disk controller, disk drive, and the Extended BASIC Command Module, a printer is a practical necessity; either the TI Thermal Printer or an RS232-compatible printer may be used. In fact, there is no provision for screen display of the output from the Cross Reference procedure. (I use the inexpensive "Paper-and-Pencil Printer," however, and so modified the CREFPRINT program to display the cross reference table on the screen, using the crude SHIFT C - CONTINUE method to stop and start the output. These simple changes are given following this review.)

EDITOR

The EDITOR program makes possible virtually any desired modification of

line numbers in a BASIC or Extended BASIC Program. Heretofore, the only way to resequence a program was to use the RESEQUENCE (RES) *command* which affects all line numbers within a program. By way of contrast, EDITOR allows one to resequence specified sections of a program without affecting others.

If, for instance, you have numbered subroutine statements in a manner which is easy to remember (1000, 2000, 3000, etc.), you can retain this numbering and "open up" a previous part of the program for insertion of additional lines. An even more useful application would be the rearrangement of sections of BASIC code. Suppose, for example, you want to merge several programs, each of which contains subroutines. Without EDITOR, you would be faced with the time consuming chore of moving all subroutines to the end of the merged program. With EDITOR, this procedure can be completed very simply and quickly by re-numbering all subroutine lines.

Finally, the EDITOR program allows deletion of sections of BASIC code. If you want to get a subroutine out of one program to use in another, it's no problem.

How EDITOR Works

If you are wondering how a BASIC program can alter another BASIC program, be assured that no mirrors are involved. It is a relatively simple procedure which anyone with Extended BASIC can use to write their own custom utility programs and even BASIC programs which write other BASIC programs!

The technique is based upon what happens to a program when it is saved with the MERGE option (see page 163,

TI Extended BASIC manual). If you have ever cataloged a disk containing a file saved with the MERGE option, you may have noticed that unlike an ordinary program which carries the Type description PROGRAM, a program saved with MERGE is actually a data file consisting of display code with variable length records having a maximum length of 163 bytes. A BASIC program can access this sequential file like any other data file.

In addition to creating a data file form, saving a program with MERGE makes two other important changes. First, the order of program lines corresponds to the order of program line numbers. (By contrast, when a program is saved without MERGE, the file is a program memory image, and lines are placed in program memory in the order in which they were entered—not according to line number.) Second, the content of each line is represented in "condensed format": All non-essential information is deleted in a coding process. When a program saved with the MERGE option is loaded into memory with the MERGE *Command* and LISTed (see TI Extended BASIC, page 122), the coding process is reversed and each program line is reconstructed.

In order to understand how the EDITOR program works, it is necessary to know how line numbers are represented in condensed format. The first two bytes of each record contain the line number represented in ASCII code. Table 1 shows how the line numbers "80" and "9020" are represented in ASCII characters. Starting with the line number 80, the first step involves representing the base 10 number in binary. Two bytes (8 bits) are available for this representation. Next, the base 10 representation of each byte is determined

Table 1
ASCII Coding of Line Numbers

Line Number	80	
Binary	Byte 1	Byte 2
Base 10	00000000	01010000
ASCII	0	80
		P

Line Number	9020	
Binary	Byte 1	Byte 2
Base 10	00100011	00111100
ASCII	35	60
	#	<

Table 2
Sample Cross Reference Output

```

MUSIC 2/1
PROGRAM UNIT (MAIN)

STRING ARRAYS      BASIC KEYWORDS      REM
NS ( )            CALL
100                130
120                230
140                240
                  250

NUMERIC ARRAYS    DATA
NT ( )            190
100                200
120                DIM
240                100

NUMERIC VARIABLES FOR
I 110              110
120                150
140                GOSUB
180                160
240                NEXT
J 150              170
170                180
                  PRINT
                  140
                  READ
                  120

SUBPROGRAMS
CLEAR
130
SOUND
230
240
250

LINE REFERENCES
220
160

```

and the corresponding ASCII symbol produced. In this case, the character with an ASCII code of 80 is "P". Applying this process to the number 9020 gives the ASCII representation "#<".

In condensed code format, when the left-most bit of a byte is "on," the software which reconstructs a program from the code is signaled that some special action will be required in the reconstruction process. In the case of line numbers, this principle applies to the first bit of the first of the two line-number bytes. When all bits in both bytes except this left-most one are "on," the number represented in base 10 is 32767 (in binary, 01111111 11111111), the highest allowable line number in a program. When the left-most bit is added, the two-byte combination becomes an end-of-file mark. Thus the first two bytes of the last condensed format record must be CHR\$(255)&CHR\$(255) or equivalently 65535 in base 10.

With this information, you should be able to understand the basic operation of the EDITOR program. The program to be edited is first saved with the MERGE option, and then the EDITOR

program is loaded and run. Upon entry of the "OLD" command provided, EDITOR inputs each record in the condensed format file and constructs the line number from the ASCII codes of the first two bytes. Program line numbers thus obtained are stored in an array—array position corresponding to record number. After the user has altered these numbers using the DELETE (DEL) and RESEQUENCE (RES) commands provided, the SAVE command initiates the process in which altered numbers are reassigned to records in the file. As each record is read a second time, the corresponding line number in the array is translated into two ASCII characters which are substituted for those on the record, and the new record is written to a new file (after making the necessary changes to any line references). At the end of this process, the end-of-file mark is written as the last record on the new file. After initializing program memory with the NEW command, all you need to do is load the new file with the MERGE command. The program will then be reconstructed and can be SAVED in the usual way.

SAVE

Texas Instruments TI-99/4A

TI-99/4A Console	\$379.00
10" Color Monitor	329.00
Disk Controller	223.00
Disk Memory Drive	369.00
Speech Synthesizer	111.00
RS232 Interface	167.00
Telephone Modem	167.00
RF Modulator	37.50
32K Memory Expansion	298.00
Solid State Printer	298.00
Joysticks (pair)	27.00
Dual Cassette Cable	11.95

Command Modules

Extended BASIC	76.00
TI LOGO	209.95
Terminal Emulator II	39.95
Speech Editor	36.95
Household Budget Management	34.95
Personal Record Keeping	39.95
Video Chess	54.95
Football	24.95
Hunt the Wumpus	21.95
The Attack	34.95
A-Maze-Ing	20.95
Connect Four	16.95
Addition & Subtraction 1	34.95
Multiplication 1	34.95
Statistics	37.95
Diagnostic	24.95

Printers

TI 840RO Basic	895.00
TI 810RO Basic	1480.00
TI 820RO Basic	1795.00
TI 825RO Basic	1408.00

Call for Prices
on other TI, Image,
Futura and Instant Software
for the TI 99/4 & TI 99/4A.

Prices FOB Lexington, KY
Add \$4 for shipping
KY Residents add 5% tax
VISA & MC Orders add 3%

ORDER TOLL FREE
(outside Kentucky)
1-800-354-9099
Kentucky Calls
606-276-1519

CBM INC.

198 Moore Dr
Lexington, KY 40503
1-800-354-9099

Simple Techniques Provide **SPEED**

- DYNAMIC FULL SCREEN GRAPHICS
- REALISTIC SOUND EFFECTS

* 3 FAST FUN GAMES *

- 1) LASER WARS—Zap alien saucers with a laser beam—for one player.
- 2) RACING—You drive the speedster. If you reach the checkered flag, did you beat the high score? Every game is different and you can customize the track for even more variety.
- 3) WRAP—(requires Joysticks)—you'll enjoy this fast paced game. One, two, or "three" players.

17 years combined experience
went into analysis, design, programming,
testing, and documenting these games.

SPECIAL OFFER

All 3 games \$9.95 cassette California orders
\$13.95 disk add 6% tax

Send cash, check, or money order to

Kemp Software, TI Games Group
3292 Mission Blvd.
San Diego, CA 92109

PROFESSIONAL SOFTWARE

for the TI-99/4

Accounts Receivable	\$ 99.95
Accounts Payable	99.95
Inventory	99.95
General Ledger	99.95
Package (All 4 Systems)	\$350.00

Extended BASIC

Inventory - Invoice	\$129.95
Payroll	\$129.95

- Disk-Based, 1 or More Drives
- Supports RS-232
- Full Documentation

For More Information Contact :

W. R. Wilson Co.
928 North Apple St.
Greenfield, IN 46140
Tel. (317) 462-1099

CROSS REFERENCE

The remaining three programs (LINPUT, CREF, and CREFPRINT) are used to produce a complete tabulation of all lines in which each variable, array, keyword, function, and line number reference occurs. An independent tabulation is provided for each subprogram. The cross reference table will give you detailed documentation for use in program development, and would also seem to be a useful tool in analyzing a poorly documented program. (See Table 2)

As in the case of the EDITOR program, the first step involves saving the program to be cross referenced by using the MERGE option. The LINPUT program converts the DISPLAY records of the merged file to INTERNAL code, presumably to speed subsequent execution. The CREF program then reads in each record of the file and analyzes its contents for the presence of all keywords, functions, etc. which occur in TI Extended BASIC, as well as in the user's variable names, arrays, line references, and subprograms. The output, a list of the line numbers in which each element is found, is written to a disk file. The file is then printed by the CREFPRINT program.

The instructions recommend that the CREF program be run in TI BASIC, rather than Extended BASIC, to speed execution. Even with this advantage, however, the cross referencing of a large

program should be planned such that you can be doing something else—like taking a trip to Switzerland. Actually, it doesn't take *quite* that long: cross referencing a program of moderate size (270 lines) took 35 minutes.

I am sure of that time because I used the CREF program six times before I discovered the source of a bug which caused a file error half way through the output of the cross reference table. It turned out that the program was attempting to print a record 255 bytes long in a VARIABLE length record format where maximum length is 254 bytes. Because the problem occurred as a result of the characteristics of the program I chose to cross reference, chances are it will never happen to you, but the fix shown following this article is recommended just in case.

How CREF Works

Although a detailed analysis of the cross reference program is beyond the scope of this article, generalization of the principles involved presumes an understanding of the structure of condensed code. As mentioned previously, the method used to signal the reconstruction software that it is encountering an "instruction" byte involves an "on" condition in the left-most bit. In contrast to line numbers, most "instructions" in condensed code consist of a single byte. When the left-most bit is

"on" (i.e., 10000000) the base 10 representation is 128. Instructions thus begin with the number 10000001 or 129.

ASCII byte codes used by the reconstruction software to generate BASIC keywords, punctuation, etc. are translatable with the program "Condensed Format Code Table." This program generates a file called DSK1.FILENAME which is in condensed format. Each record in the file contains a single byte in the third position beginning with ASCII 129 and ending with ASCII 254. This byte will be interpreted as an "instruction" by the reconstruction software. Preceding the byte, a two-byte line number is written; following it is an end-of-line mark, ASCII 0. Line numbers have been set equal to the ASCII code for ease in subsequent interpretation of the results.

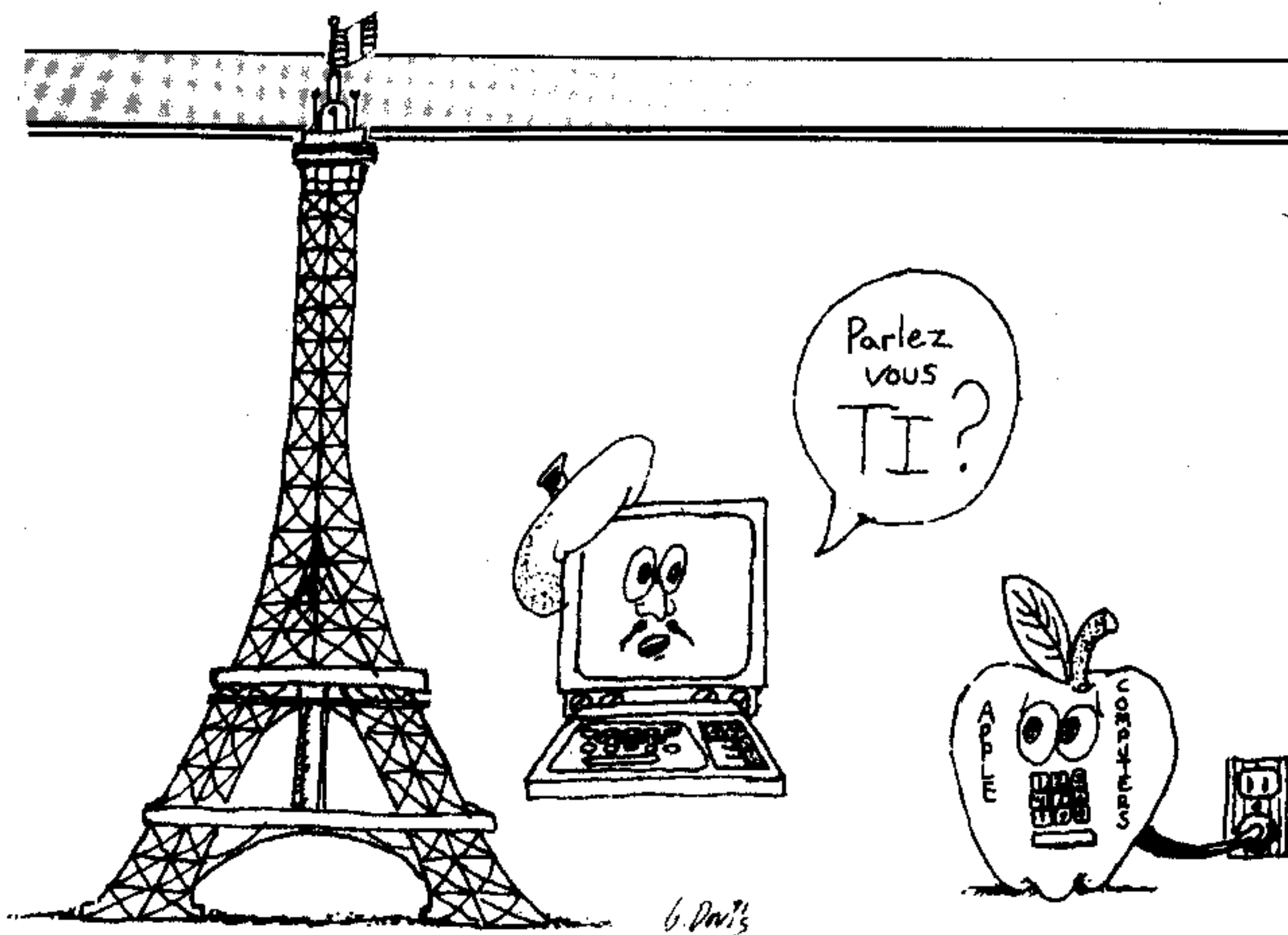
In order to view the reconstruction of each potential BASIC element, you first initialize program memory with the NEW command, then load the output file with the MERGE command as if it were a program—i.e., MERGE DSK1.FILENAME. The result is given in Table 3. For example:

```
CHR$(129) is reconstructed as ELSE
CHR$(130) as ::
CHR$(166) as WARNING
```

Continued on p. 77

Applesoft to TI BASIC

By Samuel D. Pincus



In the previous issue (July/August 1981), Fred Forster used this column to show us how to convert TRS-80 BASIC to TI BASIC. Well, as many of you know, the TRS-80 hasn't been the only machine to spawn prodigious amounts of software. The Apple II has also generated its fair share of applications and games programs—most of them taking advantage of the Apple's color graphics capability. In this regard, the Apple is more like the TI-99/4(A) than the *non-color* TRS-80. And, as it turns out, the TI BASIC interpreter is indeed very similar to the APPLESOFT BASIC interpreter. In this first article, I'll discuss some of the major considerations when converting from APPLESOFT to TI BASIC. In a later article, I'll deal with Apple graphics.

The APPLESOFT language card has about 29 non-graphic commands which are identical to TI BASIC. These commands, shown in Table 1 below, can be copied without much concern over compatibility.

ABS	DEF	GOTO	ON...GOSUB	SQR
ASC	DIM	INT	READ	STEP
ATN	END	LEN	REM	STOP
CHR\$	EXP	LET	RETURN	STR\$
COS	FOR...TO	LOG	SGN	TAN
DATA	GOSUB	ON...GOTO	SIN	

Table 1

In the remaining 26 or so commands, the differences range from very slight to major. Most importantly, the differences though slight in format or content can cause major problems in converting code. I'll go into each command showing what to look for and how to resolve difficulties.

String Commands

APPLESOFT uses three different commands (LEFT\$, MID\$, and RIGHT\$) in place of the TI's SEG\$. The statement LEFT\$(A\$,N) references the first N characters of string A\$. This directly translates into SEG\$(A\$,1,N). MID\$(A\$,M,N) is the same as SEG\$(A\$,M,N). Right\$(A\$,N) references the last N characters in string A\$. The best way to duplicate this is to combine the LEN and SEG commands as follows: SEG\$(A\$,LEN(A\$)-N+1,N).

The VAL function acts the same way in both APPLESOFT and TI BASIC if the field being VALed is a valid numeric string. That is, both will return 45.2 as the value

of "45.2". If the string does not contain valid numeric characters, however, the results are very different. TI BASIC will stop the program if the field contains non-numeric characters. APPLESOFT, however, will return with the numeric equivalent of the numbers found in the string before the first non-numeric character. For example: VAL("123AB") will return with 123. If the first character of the string isn't numeric, APPLESOFT returns a 0.

This is important because it means that APPLESOFT does not have to edit a string prior to the VAL statement. A typical program will have code such as

```
10 INPUT A$
20 X=VAL(A$)
30 IF X=0 THEN 10
```

I've found that in most cases, I can ignore the whole issue by using TI's built-in numeric editor and coding INPUT X in place of statements 10 to 30 above. If you can't do this, use the following routine to replace the APPLESOFT VAL command:

```
10 FOR Y=1 TO LEN(A$)
20 IF(ASC(SEGS(A$,Y,1))<48)+(ASC(SEGS(A$,Y,1))>57) THEN 40
30 NEXT Y
40 Y=Y-1
50 Y=VAL(SEGS(A$,1,Y))
```

Note: This is *not* a rigorous equivalent of APPLESOFT's VAL, but it is sufficient for whole numbers greater than -1.

FOR-TO-STEP-NEXT

In the usual run of programs, the FOR-TO-STEP statement is identical in the two interpreters. There is, however, a very significant difference to look out for. The BASIC statement FOR Z=5 TO 4 will execute once in APPLESOFT but will *not* execute at all in TI BASIC! This difference is important but can easily be spotted while transcribing a program. It isn't so obvious if the statement were FOR Z=A TO B where A and B are computed variables. The safest thing is to test for A greater than B. If it is, make B equal to A before entering the loop.

Both interpreters treat the STEP statement the same way and are very similar in the format of the NEXT statement—though in APPLESOFT, NEXT may be used by itself to end a single FOR loop; if the FOR loops are nested, however, APPLESOFT needs the variable name just as in TI BASIC.

MORE

What A Bargain! Order Now!

Game Packages I & II

All 6 Games contained in our first two games packages... on one disk!

Games I Package

Space Bar Bandit—Turns your TI into a slot machine! Optional speech output.

Bio Rhythms II—Plots physical, emotional and intellectual cycles for any time period for one or two simultaneously!

Number Please—All the excitement of Master Mind.* Optional speech output.

*Master Mind is a trademark of Invicta Plastics.

Games II Package

Eliza—Your TI becomes your psychotherapist! Plain English input and reply.

TI Post—Lets you create unlimited verse on any subject you select. Change or add to subject matter.

Electron—Rays are reflected, deflected or absorbed, giving you clues to location of hidden electron. A space-age Black Box.**

**Black Box is a trademark of Parker Brothers.

Available in TI BASIC or TI Extended BASIC. The Extended BASIC version contains the same games, but is auto-loading and menu driven.

Order GD-01 (Standard BASIC) — \$19.95
Order GD-X-01 (Extended BASIC) — \$21.95

Timore Game Packages I & II are still available separately:

Order Games I—GT-01 (Tape)—\$10.95 or GD-01 (Disk)—\$14.95
Order Games II—GT-01 (Tape)—\$10.95 or GD-01 (Disk)—\$14.95

COMING SOON—The TI software most requested by our customers. See our next advertisement for the exciting details.

Please send me:

SUB-TOTAL	\$	_____
Add \$1 for shipping & handling	+\$	1.00
Florida residents add 4% sales tax	+\$	_____
TOTAL ENCLOSED	\$	_____

(All Timore disks and tapes are compatible with standard TI 99/4 systems. Games I and II are written in standard TI BASIC.)

Name _____
Address _____
City _____
State/Zip _____

Make check or money order payable to:

TIMORE PRODUCTS

A division of M-I-A Systems Corporation

Dept. 99-2 • P.O. Box 1431 • Winter Park, Florida 32790

INPUT/OUTPUT (I/O)

The APPLE II has a screen 40 characters wide with 24 rows. It has automatic tabs at 11, 22, and 33. The TI has 24 rows of 28 columns with a single tab at 14. This means that a large portion of the conversion effort will be directed toward the re-formatting of print data. A further explanation of screen formatting will be found in the SCREEN commands section later.

Both machines use very similar INPUT and PRINT statements. They differ only in the use of print separators. Both use the comma as a tab command and the semicolon as a non-space separator. APPLESOFT reserves the colon for a special use and doesn't treat it as a new line separator. When converting, always keep this in mind because it provides a powerful formatting tool when converting PRINT statements. The TAB command is similar in both interpreters, but TI machine skips to a new line if a TAB value is less than the current column location. The APPLE will ignore the TAB statement in this case.

As part of the print function, APPLESOFT has a command of the format SPC(N) which is used to print N spaces. This must be replaced with a string of N spaces in the TI PRINT statement. APPLESOFT has to be very careful with spaces because it does *not* format a number with leading and trailing spaces the way TI BASIC does. This means that it is very rare to see something like PRINT J;K in APPLESOFT—a perfectly acceptable command in TI code.

The APPLE II screen starts off with the cursor at the top and works its way down to the bottom before scrolling begins. To replace the TI colon in forcing line feeds, the APPLE uses the HTAB and VTAB statements to shift the cursor horizontally and vertically. This is used to print information at different locations on the screen. When converting, either change the print format to use line-feeds (colons), or use HCHAR to print at an equivalent location. *Note:* TI provides a full PRINT AT (using HCHAR) routine as part of its *Programming Aids I* package, but it is *very* slow. In many cases (where scrolling is acceptable), you are better off setting up a sequence of PRINT commands using the colon (PRINT :::::). If you must use the HCHAR method of print out, here's a routine to print string A\$ at row RO column CO:

```

10 FOR X=1 TO LEN(A$)
20 CALL HCHAR(RO,CO+X-1,ASC(SEGS(A$,X,1)))
30 NEXT X

```

This routine is much faster but requires you to remember to begin at column 3 (where TI BASIC begins its PRINT line) and not to allow A\$ to extend past column 30 (where TI ends its PRINT line).

The prompt for APPLESOFT input is the same as for TI BASIC except that it uses a semicolon in place of the colon to separate the prompt from the input variable.

For example:

```

10 "ENTER A NUMBER":Q
VS
10 "ENTER A NUMBER":Q

```

The last I/O difference concerns getting a single character without using the INPUT statement: APPLE uses the GET statement, while TI uses the CALL KEY statement.

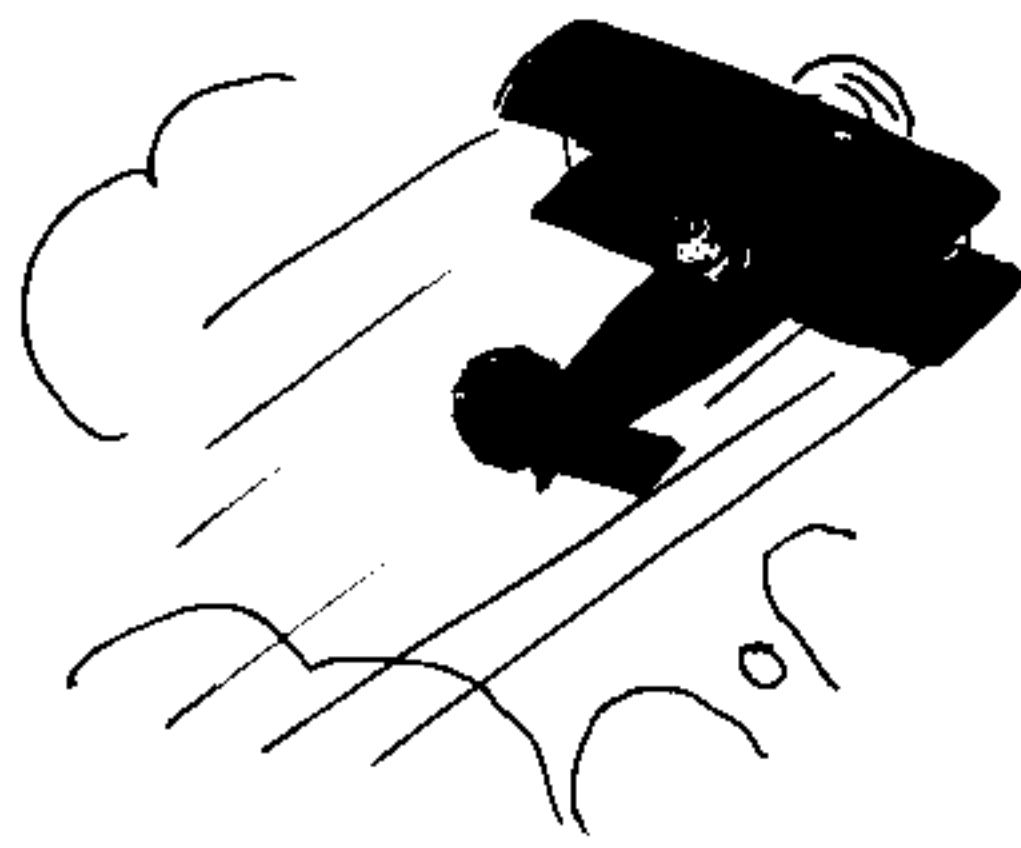
DATA-RESTORE

Both interpreters have identical DATA statements but the RESTORE is much more powerful in the TI BASIC version. APPLESOFT will only RESTORE to the *first* DATA statement in the program. For this reason, it isn't unusual to see an APPLESOFT program reading 100 or more DATA elements in order to get to the one needed by a particular routine. For TI-BASIC, a simple RESTORE NNN (where

Continued on p. 44

KELLEY'S

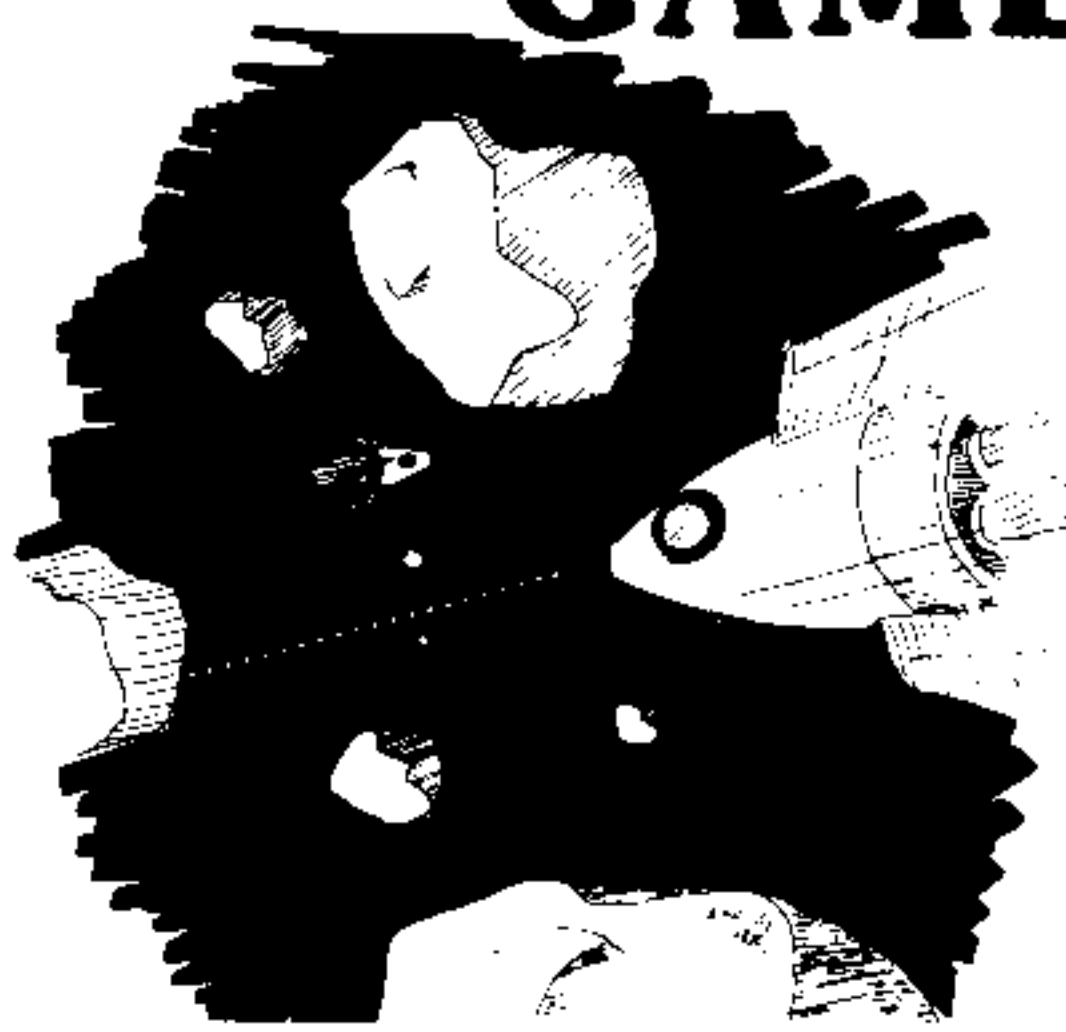
KORNER



Dogfight

By W.K. Balthrop

GAMES of AGILITY and DERRING-DO



Space War

By Mark Moseley

Listen 99'ers,
And you shall hear,
The roar of two rockets
As they blast missiles and veer.

It's an old-fashioned shootout
In deep, deep, dark space,
You're surrounded by asteroids,
So carefully give chase . . .

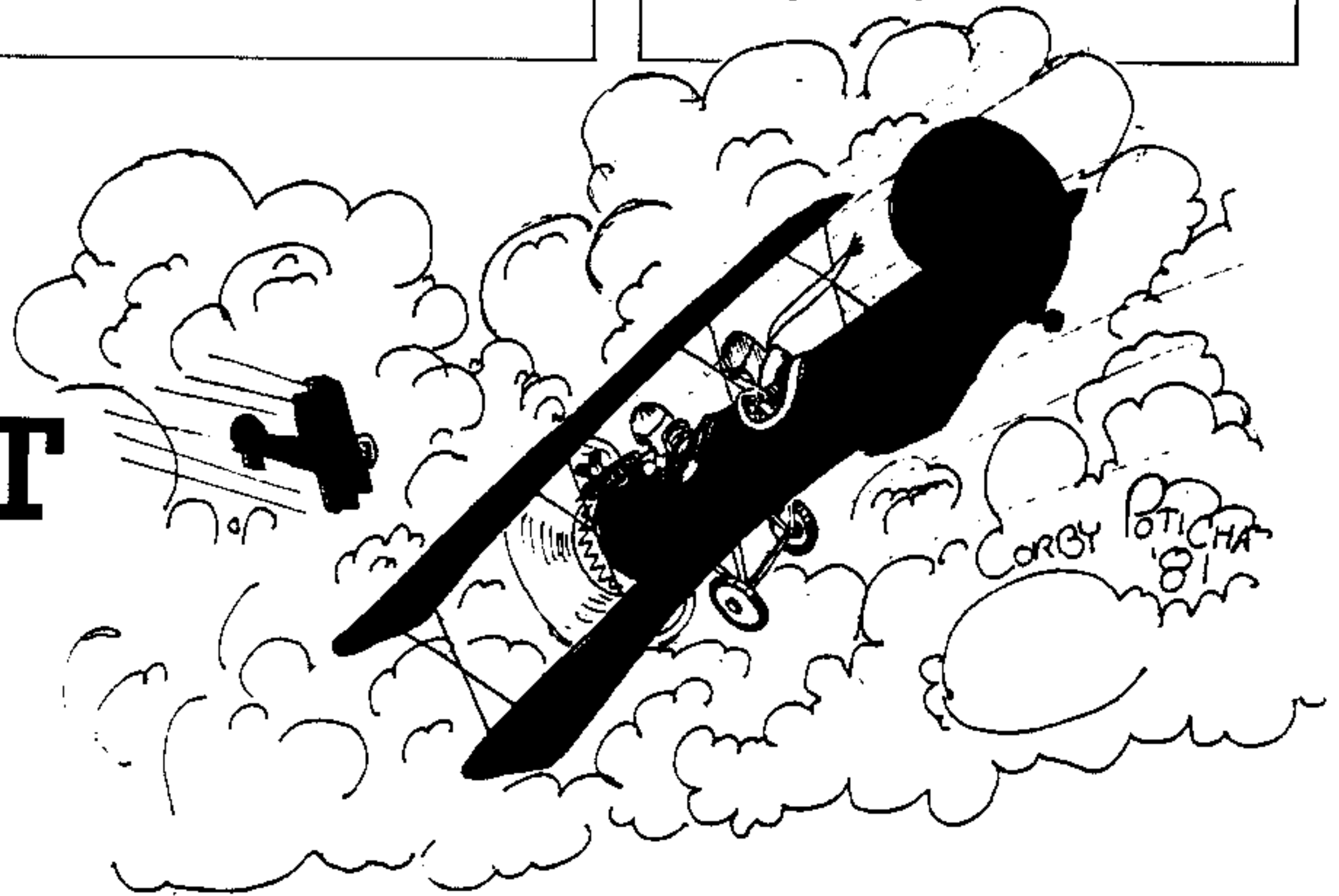
Just four centuries earlier,
In skies laced with fear,
It was a victorious dogfight,
Then stein of stout beer.

So it's your choice of combat
With missile or gun,
Just key in a program,
And get ready for FUN !

Forget all the educational and technical stuff you've been reading in the rest of this magazine. Sure, it's been interesting and informative . . . but you need some fun too! Right? Relax then. You're in my territory now: Kelley's Korner—the place for great graphic games and sensational simulations.

DOGFIGHT

By W.K. Balthrop



Dogfight is a two-player game written in Extended BASIC. Each player has control of one aircraft—a bi-plane. You must out-maneuver your opponent and shoot him down before he can do the same to you. If both planes crash into each other, the score will not change. The first player to destroy 10 enemy aircraft wins the game.

Your plane is controlled with 4 directional keys. Unlike most games, the key pressed will *not* cause your plane to *immediately* move in that particular direction. For example, if your plane is traveling down and to the right at a bearing of 135 degrees and you press "E" or the up arrow, your plane will turn its nose up and first change its heading to 90 degrees, then to 45 degrees, and finally to due North—i.e., straight up. This gives the plane a more realistic move-

ment and makes unrealistic, 180-degree *hairpin* turns impossible.

To make a 180-degree turn, you must first press a key indicating the direction of turn. For example, if you are traveling due East (left to right) and want to make a 180-degree turn to go due West, you must first press either "E" or "X" to indicate the upward or downward turn. Pressing "S," the left directional key, will have no effect. In a similar way, the player using the right-hand side of the keyboard uses I, J, K, and M to move the plane. (If you have the overlay for the *Video Games Command Module*, you might find it convenient to use.)

Pressing the "F" and "H" keys will fire the guns, but only *one* shot can be fired at a time. Each shot has a limited range and cannot be carried over the edge of the screen to the opposite edge.

You can't terminate a bad shot; it must first go off screen. The limit of only one shot at a time was placed in the program so that the computer could make accurate coincidence checks with objects moving fairly rapidly on the screen.

Four levels of difficulty make the game easy enough for beginners and challenging enough to hold the interest of experts. The higher the level of difficulty, the faster the planes and shots will move. The option to fly a day or night mission will change the screen color to either light blue or black.

Features of Extended BASIC Used in "Dogfight"

For you 99'er readers who are also programmers, Dogfight illustrates many of the features of Extended BASIC.

Continued on p. 37

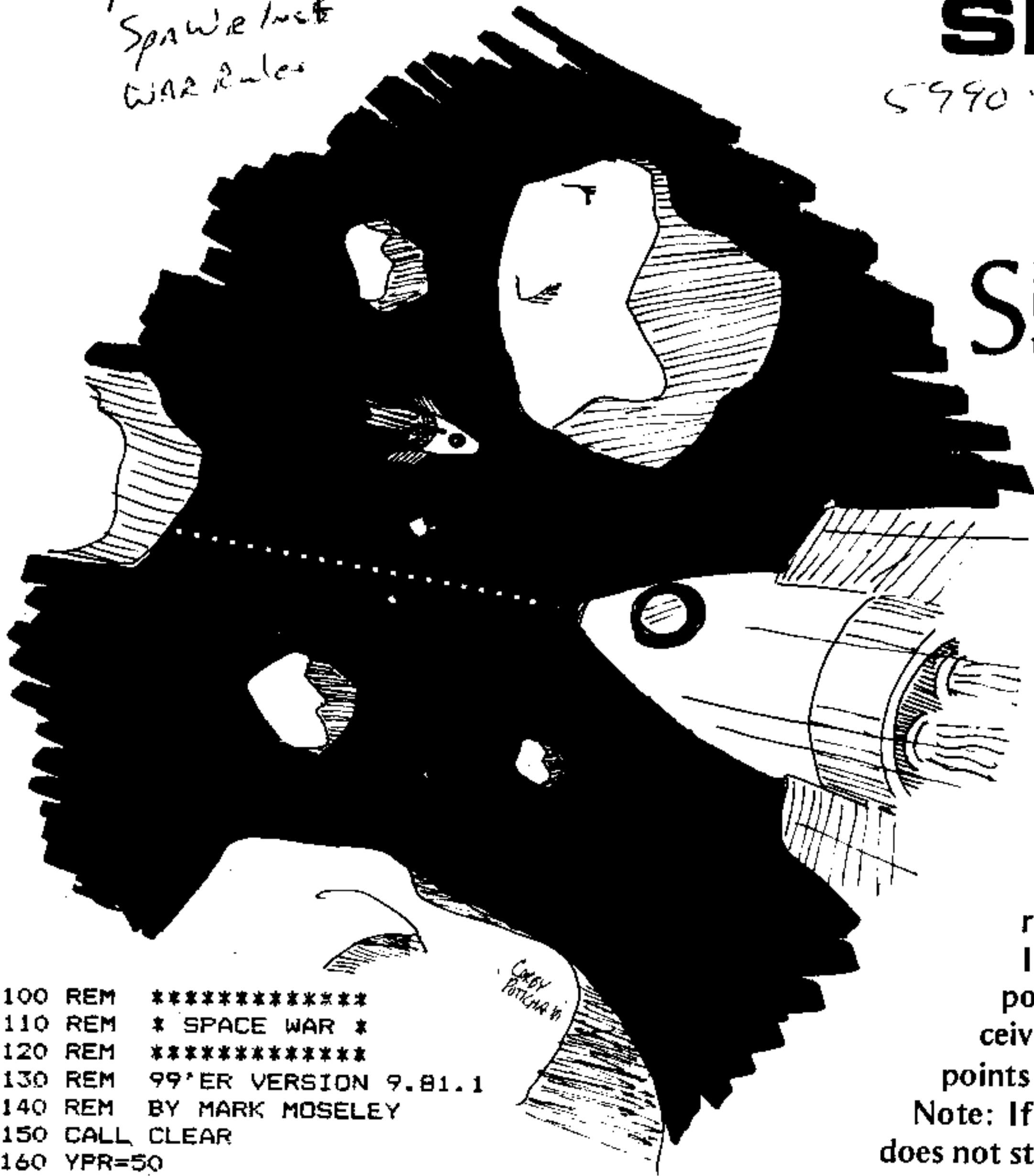
SPACE WAR Inst
 Spawer Inst
 WAR Rules

4740 add GO
 3580+5470 add space

SPACE WAR

5990+6000-150

By Mark Moseley



```

100 REM *****
110 REM * SPACE WAR *
120 REM *****
130 REM 99'ER VERSION 9.81.1
140 REM BY MARK MOSELEY
150 CALL CLEAR
160 YPR=50
170 BPR=50
180 CALL SCREEN(2)
190 CALL CHAR(128,"0303030303030303")
200 CALL CHAR(129,"C0C0C0C0C0C0C0C0")
210 CALL CHAR(120,"F0F0F0F0F0F0F0F0")
220 CALL CHAR(121,"FFFFFFFF")
230 CALL CHAR(152,"0B1C2A4908080808")
240 CALL CHAR(153,"08080808492A1C08")
250 CALL CHAR(154,"00080402FF020408")
260 CALL CHAR(155,"00102040FF40201")
270 CALL CHAR(156,"1F0305091120408")
280 CALL CHAR(157,"FBC0A09088040201")
290 CALL CHAR(158,"B04020110905031F")
300 CALL CHAR(159,"0102048890A0C0F8")
310 A1$=CHR$(157)&" "&CHR$(152)&" "&CHR$(156)
320 A2$=CHR$(159)&" "&CHR$(153)&" "&CHR$(158)
330 CALL COLOR(13,2,2)
340 CALL COLOR(15,2,2)
350 FOR I=1 TO 8
360 CALL COLOR(I,2,2)
370 NEXT I
380 PRINT TAB(4);"PRESS ANY KEY TO BEGIN":;;;
390 DATA 3,6,3,4,6,1,5,6,3,6,8,1,7,6,3,3,11,1,5,
11,1,3,15,2,5,15,2,3,19,3,7,19,3
400 DATA 3,23,3,5,24,1,7,23,3,13,14,5,9,21,2,11,
21,2,9,25,3,11,26,1,13,27,1
410 DATA 3,10,5,3,12,3,3,14,5,3,17,5,4,19,3,4,23,
3,9,14,4,11,16,2,9,18,4
420 DATA 9,20,5,9,23,5,10,25,4,10,27,2
430 FOR I=1 TO 20
440 READ A,B,C
450 CALL HCHAR(A,B,144,C)
460 NEXT I
470 FOR I=1 TO 13
480 READ A,B,C
490 CALL VCHAR(A,B,144,C)
500 NEXT I
510 CALL HCHAR(12,26,128)
520 CALL HCHAR(12,27,129)
530 CALL COLOR(15,16,16)
540 CALL COLOR(13,16,2)
550 FOR I=1 TO 8
560 CALL COLOR(I,16,2)
570 NEXT I
580 CALL HCHAR(1,2,120,29)
590 CALL HCHAR(24,2,120,29)
600 CALL VCHAR(1,30,121,24)
610 CALL VCHAR(1,2,121,24)
620 CALL COLOR(12,7,2)

```

Note:
 If using a disk system, type CALL FILES(1) prior to RUNing. Even so, you still might encounter some conditions during play when the memory will fill and the program will halt. To eliminate this, you can delete all the instructional PRINT statements.

Space War is a two-player game written in TI BASIC. Each player has one rocket. The object of the game is to destroy your opponent by missile fire, or by either forcing him to crash with an asteroid or causing him to use up his allotment of fuel.

You can fire missiles in any of the 8 directions selectable from each side of the split keyboard. Missiles emit a nerve gas that paralyzes any moving object on the screen until a hit is made or the missile goes out of range—i.e., off the screen. Firing a missile, however, does require an expenditure of fuel.

Each rocket starts out with 50 units of fuel. One unit is subtracted for each move, and a missile shot costs 5 units of fuel, so you must try to move efficiently and shoot accurately. If you run out of fuel, the game ends and the other player receives 2 points.

If your missile hits the enemy rocket, you score 5 points. If you crash into an asteroid, your opponent receives 3 points. And if you crash into each other, no points are awarded.

Note: If you shoot an asteroid you lose 1 point but the game does not stop.

EXPLANATION OF THE PROGRAM *Space War*

Line Nos.	Description
150-180	Clears screen, initializes fuel 50 units; makes black screen.
190-370	Definition of characters and colors for title screen and instructions. Characters 152-159 are arrows.
380-570	Draws title screen.
580-660	Draws border and blinks colors until user presses a key.
670-800	Asks if user wants instructions and waits for response.
810-1270	Prints instructions invisibly and makes white letters appear on black screen.
1280-1350	Clears screen, resets letters to white on black and defines colors for game.
1360-1790	Defines characters for graphics. Characters starting with R are the rockets in different directions; VS is for the missile; S indicates asteroids, and D crashing graphics.
1800-1880	Clears screen for game, initializes variables and draws rockets. A1, B1 are coordinates for crashing; A, B, and C, D are the rocket's coordinates.
1890-2150	Draws center asteroid then 7 random asteroids making sure asteroids do not overlap.
2160-2240	Receives players' input. If a key has been pressed, branches accordingly. If no key has been pressed, goes to the other player's keyboard input. Initializes variables. G indicates which player is doing something. V=1 when an asteroid has been hit.
2250-2310	Procedure when yellow rocket fires a missile.
2320-2470	Procedure when yellow rocket moves.
2480-2540	Procedure when blue rocket fires missile.
2550-2700	Procedure when blue rocket moves.
2710-3460	Routines for moving the blue rocket different directions.



The Third-Party Pascal Development System: SOME USER IMPRESSIONS

By F. T. Berkey

Anyone who has programmed in BASIC is well aware of the limitations imposed by that language in writing large applications programs. Interpreted BASIC is slow and it is virtually impossible to directly transfer BASIC code from brand X to brand Y microcomputer without tedious and time consuming modifications. Furthermore, the logic in large BASIC programs is difficult, if not impossible, to follow. In addition to BASIC, most home computer manufacturers now make available one or more structured languages. Texas Instruments intends to follow their lead and will market a UCSD Pascal system in late 1981. For several months Texas Instruments has been offering a third-party Pascal development system for individuals or firms interested in developing and marketing home computer software. The introduction of this development system appears to have marked a turning point in TI corporate attitude toward third-party software. The development system gives the third-party software developer the capability of developing software in both Pascal and TMS9900 assembly language. As explained by Gary Kaplan in a previous issue of *99'er Magazine* (May/June, P. 52 to p. 56), Pascal P-code is transportable and can be executed on any other system using the UCSD Pascal system. This opens up a market far exceeding that of TI-99/4 users alone. A further advantage is that Command Modules can be programmed in either Pascal or TMS9900 assembly language. Before reading what follows, I strongly advise you to scan the two articles which discuss UCSD Pascal and third-party development systems.

The Hardware

So what exactly comprises the UCSD Pascal development system? The major piece of hardware presently associated with the development system is a 13" by 17" by 3" black box which contains the 32Kbyte expansion RAM and auxil-

iary GRAM (Graphics Random Access Memory). In addition to this "Super Memory" box, you will need a disk controller and two disk drives. (I am told that you can get by with *one* drive but I wouldn't recommend it.) Of course, you'll also need a console and a monitor or RF modulator and TV. The memory box is connected to the system bus by a 5 inch, 44 wire cable. Due to the extremely short length of the cable, I found it necessary to place my disk controller and RS-232 interface on top of this memory box and to use the lid of the shipping carton to support the console! Increasing the length of the cable would have saved a lot of frustration. [The "Super Memory" box will not be a part of the production version. The function of this hybrid RAM expansion/P-code peripheral will be filled by two *separate* units: the 32K RAM Expansion, and a P-Code peripheral—Ed.]

programs associated with Pascal, the operating system, the compiler, and the assembler; the Command Module contains a boot utility, TIBUG (a monitor program) and a downloader. Before discussing the software, I should mention that the UCSD Pascal system was designed to be all things to all microcomputers. Thus, the UCSD P-system is equipped to support a wide variety of terminals operating with several different microcomputers.

In its *present* implementation of UCSD Pascal, Texas Instruments has seen fit *not* to include all the various bells and whistles that the original system supplies. [Version 4.0 re-implements the bells and whistles, such that a wide variety of terminals can be used. It is much more flexible in dealing with I/O.] There are both good and bad aspects to this decision: Texas Instruments has spared the third-party developer the

"In BASIC, this program executed in 340 seconds (all primes between 3 and 6000) while the Pascal version required only 22 seconds . . . some 15 times faster than BASIC."

The Software

The present UCSD Pascal software release in the hands of third-party developers is Version 2.0, but TI intends to upgrade to SofTech Microsystem's most recent release (V4.0) in the near future. TI has included sound, speech, sprites and color graphics in its implementation of UCSD Pascal. As might be expected, the Pascal procedures and functions bear a close resemblance to similar functions incorporated in TI Extended BASIC for programming these same enhancements. In many cases, the parameters in the Pascal argument list are essentially identical to those of the comparable BASIC subprogram.

The software is provided on three floppy disks and in one Command Module. The floppies contain various

effort needed to configure the system to a particular type of terminal—obviously assuming that the standard console would be used. Nor is it necessary to use the boot-strap loader utility, since this is provided in ROM by the Pascal Command Module.

Some, but not all, of the utility programs incorporated in the UCSD system are implemented by TI. The UCSD utilities which are not transparent to the user are LIBRARIAN.CODE and PATCH.CODE.

The librarian utility allows the user to add compiled code to the system library. PATCH is an extremely useful utility which enables the user to examine and/or modify a single byte from any data or code file. I have used this utility for two purposes; the first use was to modify the RS-232 parameters in one of

TI-99/4 Software

- Astronomy Programs
- Matrix Routines
- Linear Equation Solver
- Fourier Transform
- Music Analysis/Synthesis
- Curve Fitting Routines
- Hyperbolic Functions
- Two-Dimensional FFT
- Data Fitter

Programming in TI BASIC
Extended BASIC and UCSD Pascal

Programs also Available for the
TRS80 in Level 2 BASIC

For further information write:

Eastbench Software Products
1290 Cliffside Drive
Logan, Utah 84321
Tel: (801) 753-1084

THE MICRO HOUSE

Complete TI-99/4 Line

★ Discount Prices ★

Extended BASIC	\$81
The Attack	32
Line Editor (cassette)	20
Line Editor (disk)	25
Mail List Merge	15

Add \$5 for Shipping & Handling

For complete price list write:

THE MICRO HOUSE
527 Simonet Street
Green Bay, WI 54301
Tel. (414) 432-2871

TI-COUNT Small Business System

• General Ledger (professionally approved)	650
• Accounts Payable (writes checks)	300
• Accounts Receivable (prepares bills)	300
• Inventory (1400 items per disk)	300
• Mailing list (renewal letters)	650

Mini Computer Performance
At Micro Prices

Only For The 99/4

For information on dealer
program, write or call.

To order, send check to:

PIKE CREEK
COMPUTER CO., INC.
2 Galaxy Drive
Newark, Delaware 19711
(302) 239-5113

the system files. My second application was to examine and fix a BASIC disk whose directory had been inadvertently clobbered and which was therefore unreadable by either the Disk Manager module or a BASIC directory program. To fix the disk directory, it was necessary only to modify several bytes at the beginning of the disk.

Three other utilities have been written by TI for specific application to the TI-99/4 system; these are DFORMAT, X9900 and TEXTXFER. DFORMAT is a utility which is used to initialize a new disk. In their User's Document, TI recommends making back up copies of the system disks before embarking into the UCSD system, and guides users through this procedure using the *Disk Manager* Command Module to initialize the disks. Once familiar with the system, this procedure can be eliminated and you can initialize disks by executing DFORMAT.CODE and zeroing the directory file from the filer utility.

The utility X9900 converts 9900 Assembly language code to an ASCII format. To utilize Assembly language subroutines in Extended BASIC programs, they must be formatted in 256 byte blocks. The utility TEXTXFER performs the transformation from 512 byte blocks (UCSD standard) to 256 byte blocks. Note that TEXTXFER's utility is not limited to Assembly language code; using TEXTXFER I have converted a Pascal text file so that it could be read by a BASIC program.

The UCSD system provides the user with a screen editor, a file manipulation utility, a library utility, a Pascal compiler, and a TMS9900 assembler. I have found that the UCSD system is easy to learn and that it provides me with many of the features available on a large time-share computer system. An Assembly language debugger, TIBUG, can be called from the Command Module menu. It can be utilized only with a modified RS-232 interface box. The downloader utility (also selected from the Command Module) uses a simple protocol to accept data via the RS-232 interface. TI indicates that UCSD Pascal can be downloaded to the TI-99/4 . . . if one happens to have a TI series 990 minicomputer handy! I have not had any direct experience with either of these utilities. [The new PATCH utility allows you to modify the parameters of the RS-232 interface that are stored in one of the system files. In the more convenient 4.0 system, those parameters are always in RAM and are readily changed—Ed.]

The screen editor is relatively easy to use but has limitations which must be recognized. For example, the editor's buffer will hold 10,240 bytes before becoming full and not allowing you to insert any further text. This corresponds roughly to 400 lines of code, so it is not a severe constraint. And, it is not

a restriction on the size of programs which may be written in Pascal, since there is a compiler option (an *include directive*) which allows you to include another source file when compiling a large program. Only one of the two editors provided by the UCSD system is supplied by TI; YALOE, a line oriented text editor, is not included as part of the TI Pascal development system, although I believe it would be more useful in a text editing situation.

Some of you may have read of the TURTLE program which provides a graphics capability on some UCSD systems. TURTLE is not a standard feature of UCSD Pascal, but is an enhancement added to many systems. Texas Instruments has not included turtle graphics, feeling that the graphics enhancements provided by sprites and color are more powerful. (Note that the TURTLE program is incorporated in TI LOGO.)

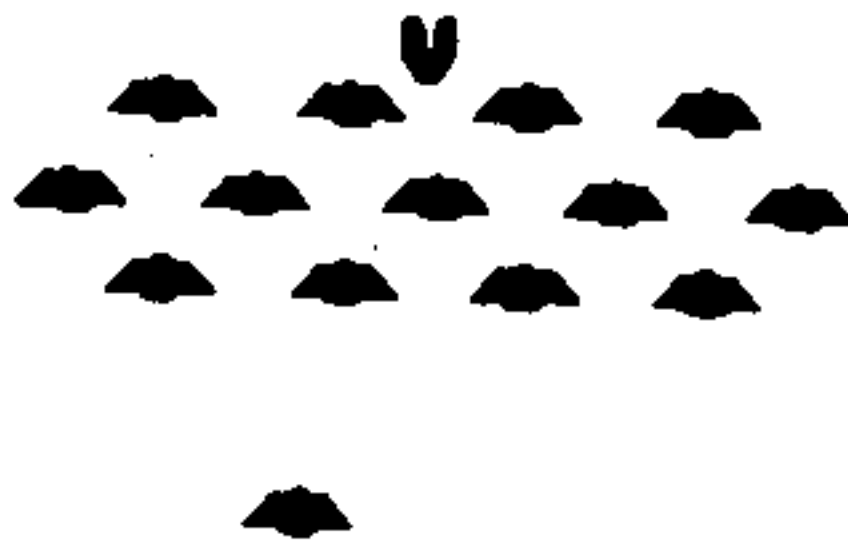
The Documentation

The large quantity of literature that accompanies the development system includes (1) *UCSD Pascal Users Manual*; (2) *TMS9900 Assembly Language Programmer's Guide*; (3) *9900 Family Systems Design and Data Book*; (4) *TI-99/4 Technical Data Manual*; (5) *The UCSD P-System Supplemental User's Document for Use with the TI-99/4 Computer*; (6) *File Management Specification for the TI-99/4 Home Computer*; (7) a Home Computer system memory, CRU and interrupt mapping specification; (8) a TIBUG manual; (9) a set of notes describing the peripheral devices such as the sound generator chip and video display chip. Conspicuous in its absence from this list is Bowles' *Guide for the UCSD Pascal System* and the *Pascal User Manual and Report* by Jensen and Wirth. In order to understand all the nuances of the UCSD system, careful reading of Bowles' book is a *must*. [The Bowles' book is available from the 99'er Bookstore—Ed.] For the ROM-based version of Pascal, however, TI is preparing five manuals which are more oriented to the home computer user than either of the books mentioned above.

The supplemental *User's Document* written by TI is a "how-to" manual which is terse at best. Its nine pages leave much unsaid, and some of what is said contains errors. It appears to be a document produced on short notice—a shortcoming that I hope will be corrected in the immediate future. An example of what has been left unsaid is the procedure that must be followed to exit from the operating system. During my initial experience with the system, I found that somehow I was clobbering the directory on the system disk. Having done so, I found it impossible to boot the system, and necessary to recopy the system disk. I then discovered a proce-

SPACE GAMES VOL. 1

AAA SCORE=00000

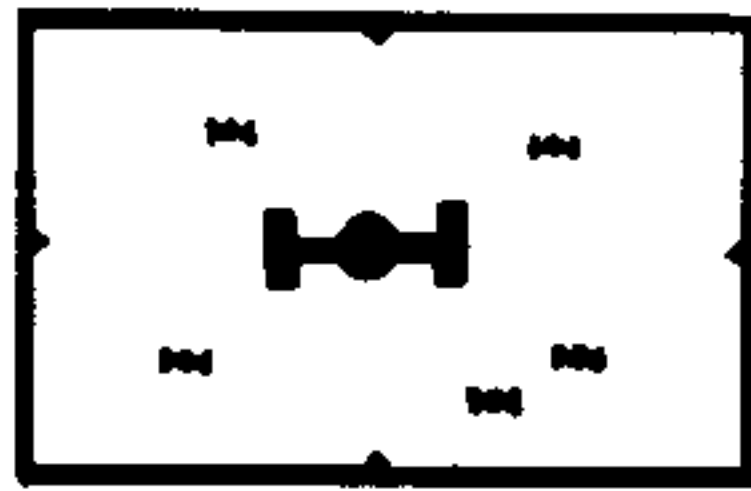


ALIEN ATTACK

Blast 'down alien saucers and the "Ultimate Alien" to score. Five lives against the endless alien army. Fast action, music, graphics. Difficulty increases with skill.

PROGRAMMED IN TI BASIC, 16K
DOCUMENTATION INCLUDED
joysticks optional

FUEL=300 SCORE=00



MAY THE FORCE BE WITH YOU

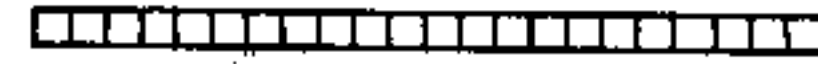
X-WING PILOT

Squads of TIE-Fighters are escaping from a raid on your rebel base. Your mission: Destroy all with limited fuel, or "Good-Bye Rebel!" Graphics, Sound, 3 skill levels.

\$12.95 - cassette
\$16.95 - diskette

Dealer Inquiries Invited

1=MOVE 2=TURN 3=SHIELD
4=TORP 5=PHASER 6=PROBE
7=ODDS 8=STATUS 9=DESTRUCT



INVADE THE EMPIRE

Commanding a starship, you must destroy 4 enemy vessels to gain control of sector ON-9. This challenging space game requires skill & strategy. Graphics, Sound, 3 skill levels.

send check or money order to:
THE SOFTWARE EXCHANGE GROUP
P O BOX 97
NORTH CHILI, NY 14514
(716) 594-9474

ture to avoid this problem: From the outer shell of the operating system I type H (for Halt) and then press SHIFT Q. This returns me to the standard TI graphic screen. I have since learned that the H command is not supported in V2.0 (where did it say that in the User's Document?), and that exiting from the filer utility should gracefully close all files! I might also note that the H command is not discussed in the *UCSD Pascal Users Manual*—perhaps because it isn't supported! [The H(alt) function is fully operational in Version 4.0, and is equivalent to SHIFT Q in TI BASIC—Ed.]

A useful feature of the development system is the ability to assign portions of the GRAM and VDP RAM as "executable disks." Since these "disks" disappear each time the system is rebooted, their primary use is in executing code already compiled. Using the system filer, a user can transfer an executable code file to either of the executable disks and have the code executed without accessing either of the floppy disk drives. [RAM Executable Disk is a feature of Version 2.0 which will be made obsolete by Version 4.0. Developers should not depend on the existence of this construct for commercial use, since the consumer version won't need or support this—Ed.]

Software developers can write assembly language programs with the third-party development system, and call

them from Pascal or Extended BASIC programs. The first stumbling block I encountered in writing an assembly language program was that the assembler directives listed in the TMS9900 Programmer's Guide cannot be used with the UCSD assembler. Instead, the UCSD system uses a very similar set of directives which must all be preceded by a period. This point is not documented in the literature supplied by TI, but becomes obvious after a careful reading of the *UCSD Users Manual* and some experimentation.

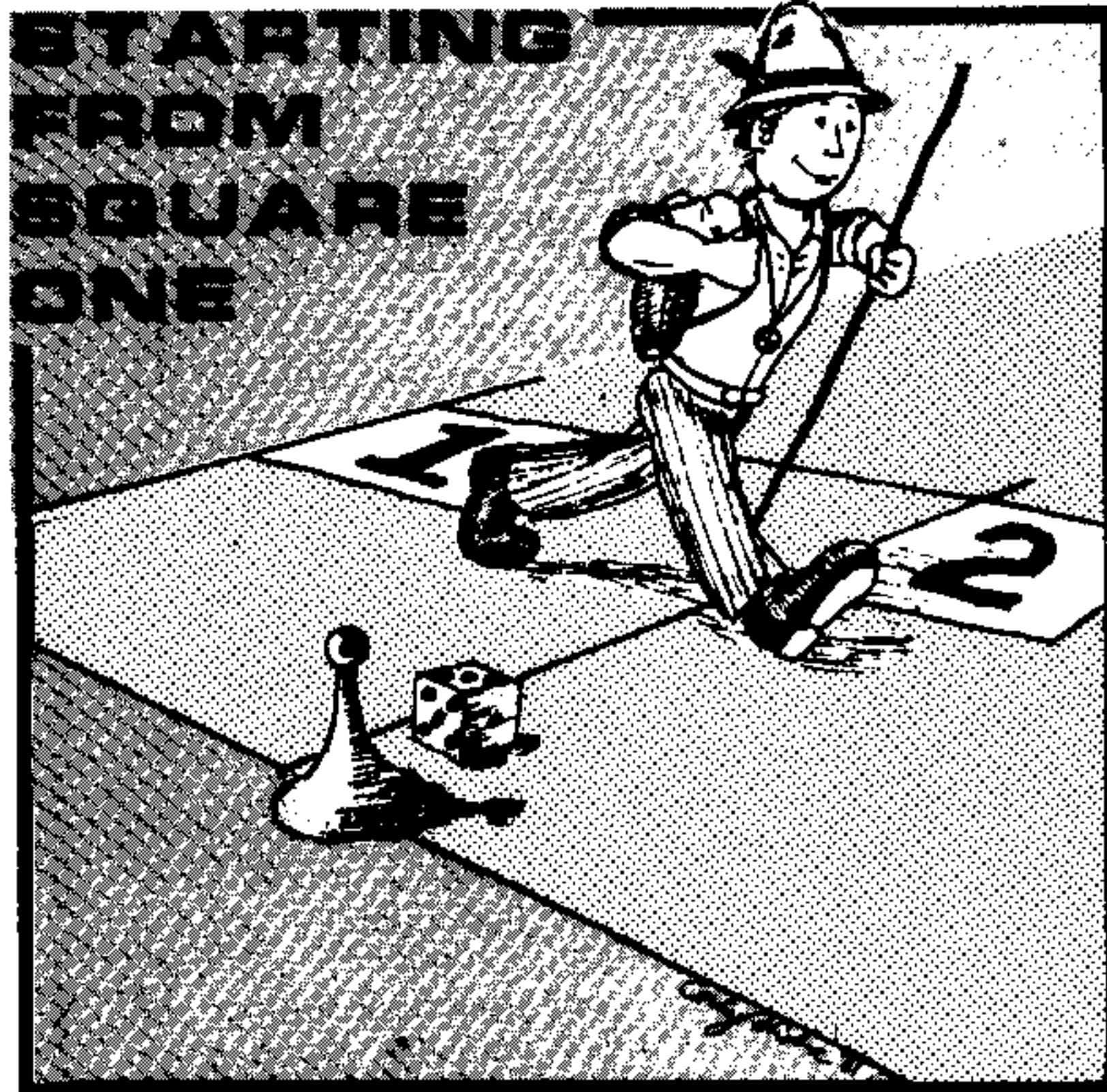
In all fairness to TI, it must be emphasized that the third-party development system is a limited production system available only through Texas Instruments and not intended for the general home computer market. In order to give third-party developers an opportunity to become familiar with the UCSD Pascal system, a developmental system was introduced onto the market in a relatively short period of time. As a result, portions of the system are still in a state of flux. [TI engineers are working overtime to complete the ROM 4.0 Pascal System. The purpose of the present hybrid system was to give developers enough lead time in which to ready their wares for release prior to the availability of consumer hardware that creates demand for those wares—Ed.]

And how does it run? Perhaps a useful indication is a comparison of the

execution time required for calculating prime numbers. In order to make such a comparison, I utilized a benchmark program that uses neither multiplication nor division (see *Byte*, September 1981, page 180). In BASIC, this program executed in 340 seconds (all primes between 3 and 6000) while the Pascal version required only 22 seconds. Thus, in this particular example, Pascal executes some 15 times faster than BASIC.

As indicated by the title, this article was not intended to be a comprehensive review of the UCSD Pascal system but rather to convey user impressions after several weeks of experience. In using the system, I have often been frustrated by the lack of disk space, particularly during program compilation. The compiler, linker, editor and library are large chunks of code which must be accessible to the system and therefore occupy considerable disk space. Solutions to this problem are to add a third disk drive, to double the density of the existing drives or to use eight-inch floppy drives. Another frustration I have encountered is an inability to stop a program once it has started executing. Neither the 'break' nor 'stop' keys have any effect on program execution. Turning the system off during execution tends to leave one with open files! I have thus far found no method to stop program execution once begun.

Continued on p. 37



Precautions & Programming Ideas FOR BEGINNERS:

PART 1-

Murphy's Law and the Home Computer

By Samuel D. Pincus

PART 2-

NOW WHAT?

By Regena

Murphy's Law and the Home Computer

By Samuel D. Pincus

"ANYTHING THAT CAN GO WRONG, WILL GO WRONG" – Murphy's Law

"And at the worst possible time." – Pincus's corollary to Murphy's Law

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

environment. Consequently, your programs, your mental health, and your wife's (or husband's) peace of mind will all be that much better for it . . .

Rule # 1 – Name that Variable !

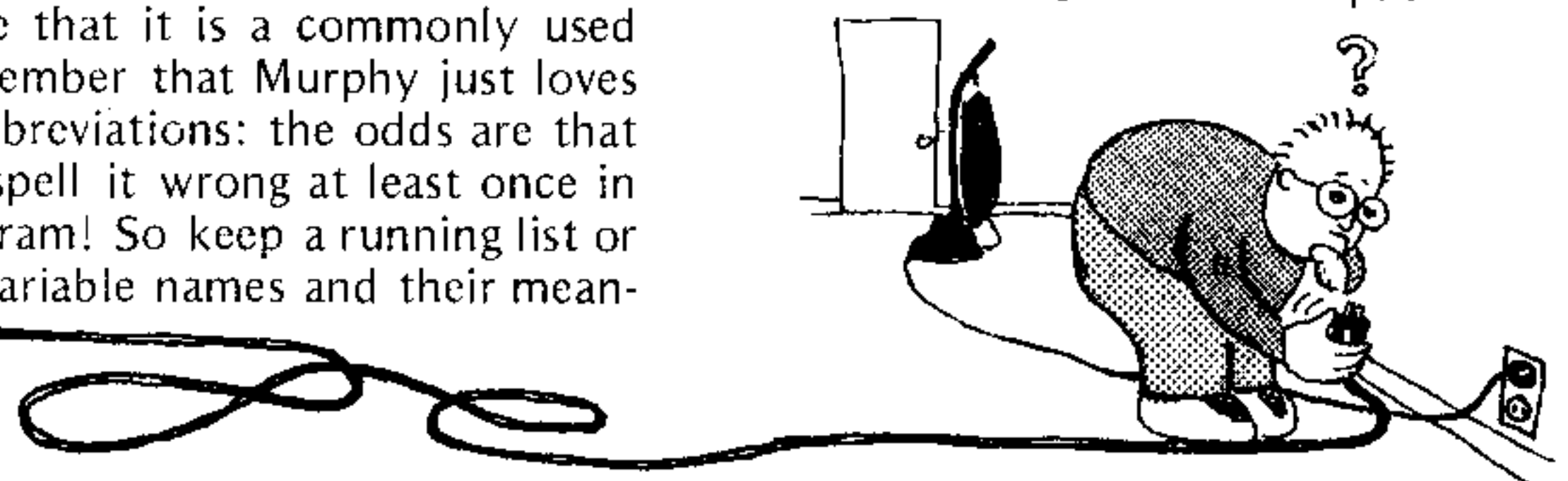
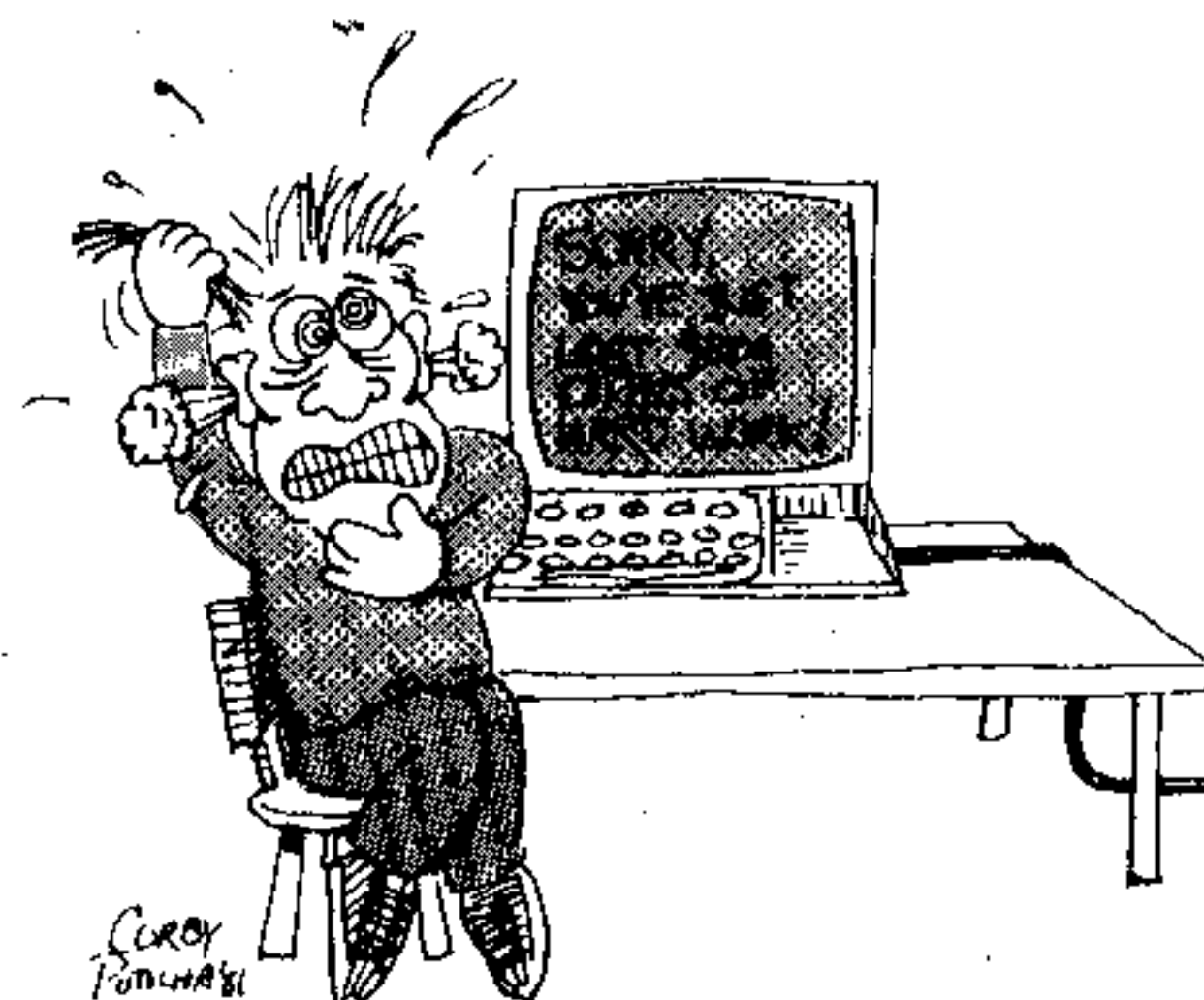
Most beginning programmers select simple, two-letter variable names like X1, X2, X3, etc. The major problem with this kind of naming scheme is that you tend to forget what each one represents. Fortunately, both of TI's BASIC languages allow you to use long variable names. Unfortunately, using long names tends to slow down a BASIC language program. The solution is to pick a 4- or 5-letter name that adequately identifies the variable it represents. For example, a program that needs a loop counter could have the name "LOOP" for the counter. If you pick an abbreviation, make sure that it is a commonly used one. Remember that Murphy just loves strange abbreviations: the odds are that you will spell it wrong at least once in your program! So keep a running list or chart of variable names and their mean-

ings in front of you while you code. Avoid the temptation to add a variable to your program without updating the chart. A poorly maintained chart is worse than no chart at all because it can lead to a false sense of security—having you believe that a "new" variable name hasn't been used before. [When picking names for variables, you should also be careful to avoid reserved words used in BASIC. So if in doubt, check your *User's Reference Guide* – Ed.]

Rule #2 – Save that Program !

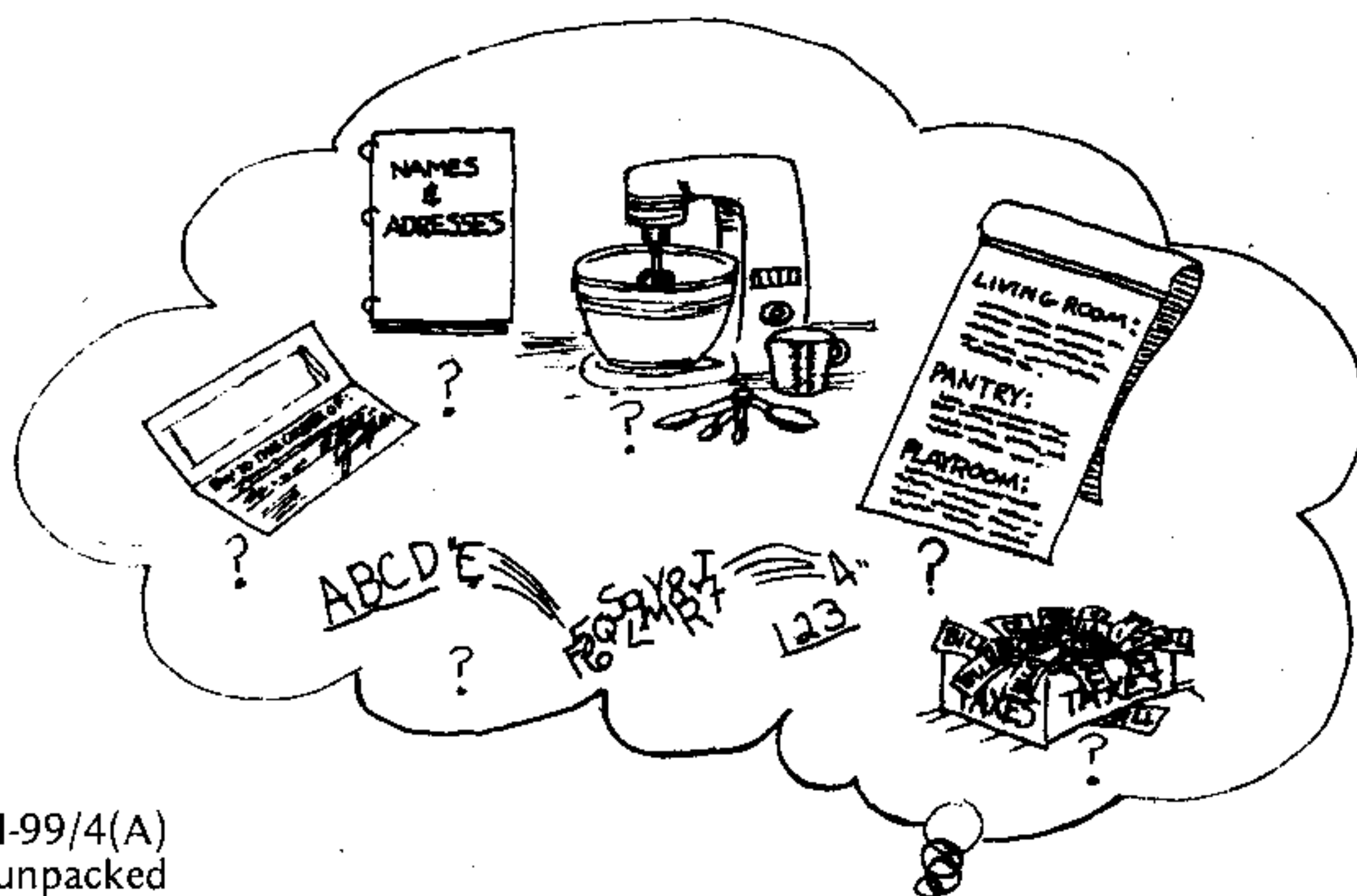
Get used to SAVEing your program after typing in about 50 lines of code if your storage medium is a floppy disk, or 75 lines if cassette tape. Keep your labels up-to-date or else you may get confused in case of trouble. I always place the version number "V.XX" on the cassette before recording on it. When recording is complete, I do *not* verify (with the "DO YOU WANT TO CHECK?" feature) the cassette recording. Instead I just flip the cassette over and cross out the version number on the new top side. It is not worth verifying the recording until a significant number of lines have been coded in. Flipping the tape over as soon as the program has been SAVEd makes sure that I don't accidentally record over my last SAVE. For disk users, I suggest that they SAVE their programs under the names V01 and V02 *alternately* so that they don't fill up a disk.

Continued on p. 56



NOW WHAT?

By Regena



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

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

By the way, be sure you have the dual cassette cables and a good cassette recorder (or the disk system if you're really into computers) because you'll want to save your valuable programs.

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

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

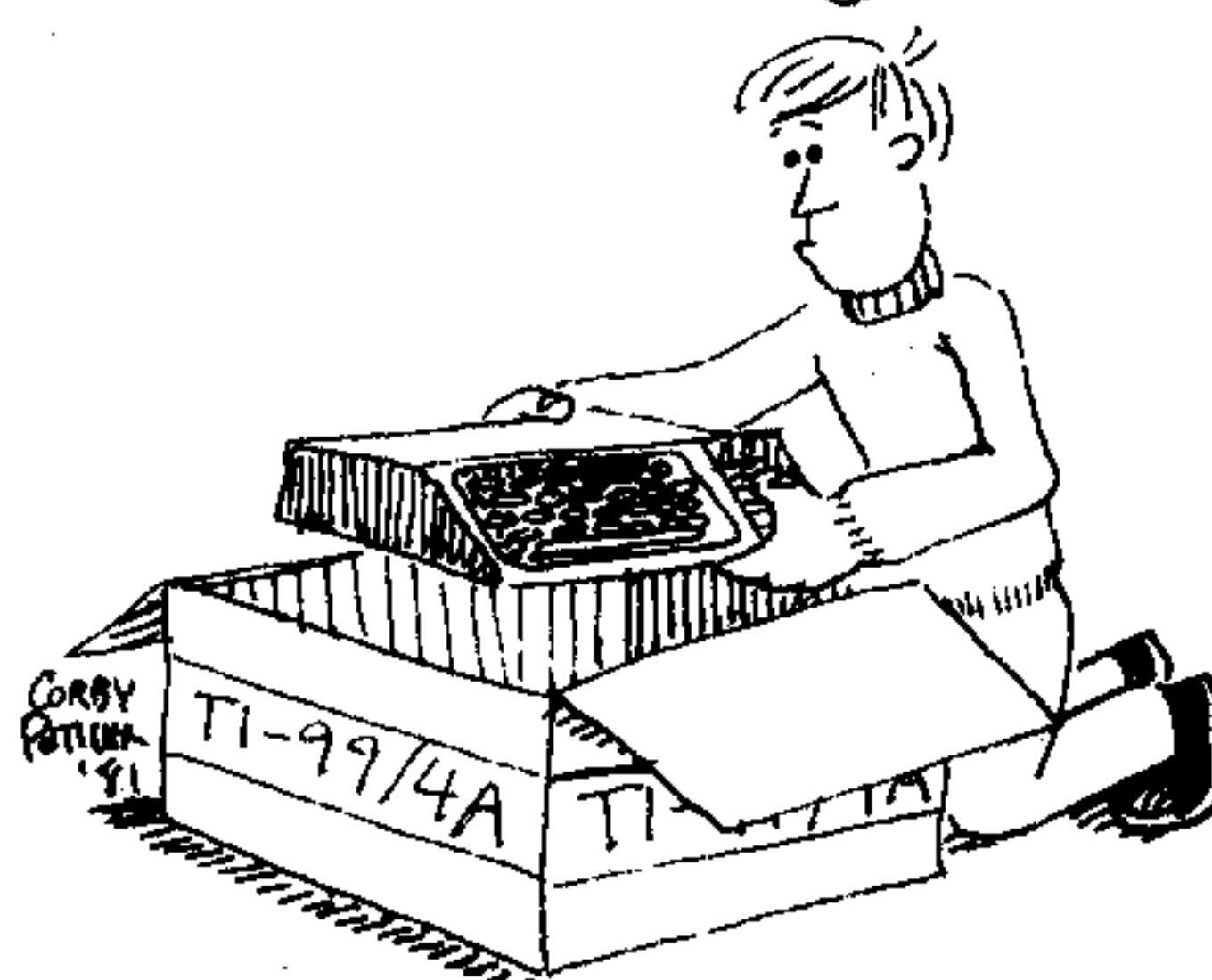
Let me just give you a few ideas for programming and then you'll be on your own.

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

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

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

I usually start a program by clearing the screen. Line 110 prints a message. You'll notice the line actually prints then moves up one line. The first colon in Line 120 says "go to the next line" then print HELLO. Another colon—so HOW ARE YOU? starts on the next line, then you "go to the next line" four times. The number of blank lines is the number of



colons at the end of the line, minus one. If the colons are at the beginning of a statement, the number of blank lines is equal to the number of colons. Don't get confused—just RUN this program and experiment a little to learn how to use the spacing effectively.

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

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

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

INPUT also allows a prompting message:

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

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

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

```
400 PRINT "ANSWER Y OR N"
410 CALL KEY(0, KEY, STATUS)
420 IF KEY=78 THEN 500
430 IF KEY<>89 THEN 410
440 Continue here for "Yes" answer
```

```
500 Continue here for "No" answer
```

3 NEW RELEASES

from

PrP computergraphics

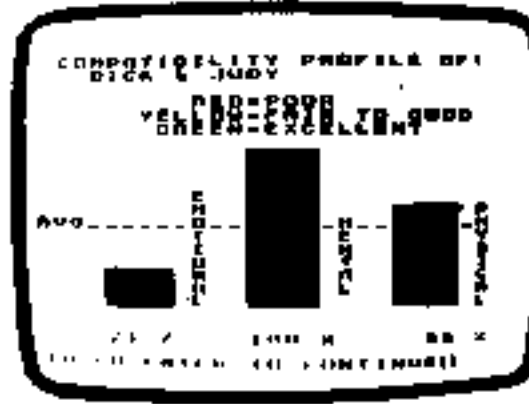
All Programs
TI 99/4 CASSETTE FORMAT

SCRIBBLE



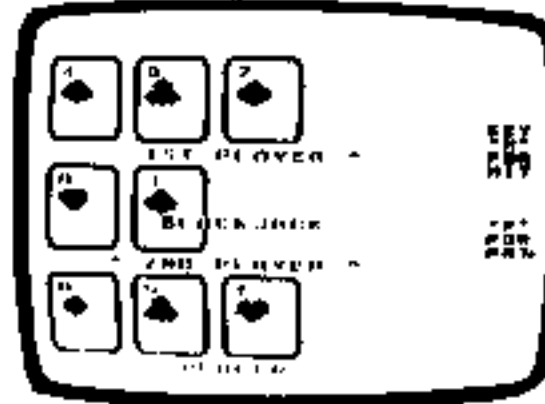
Fascinating computer word game. Players form cross words from 8 random letters offered by computer. Colorful Graphic playing board displayed. Scores are calculated and shown in personalized color Box Scores. Extra Bonus Color Squares in game. Exciting and easy to play. A PrP favorite.

BIORYTHYM



Outstanding use of Color Graphics show computer analysis of Bio Rhythms based on any birthdate since 1801. Displays mental, physical, and emotional status for any given day. Graphics dramatically change to show status of succeeding days. Same program contains a compatibility analysis for any two people. Results shown in spectacular color graphics with sound. Humorous comments. Exciting program from PrP.

BLACKJACK



Computer, as Dealer, plays against 1 or 2 Players. Excellent Color Graphics actually display the dealt cards on the screen. Dealer's second card remains hidden until players pass or take hits. Up to 5 cards displayed for dealer and both players, all at the same time. Cards are dealt without repeats from 52 card deck. Fun, realistic. A PrP winner.

Mail Coupon and Check or
Money Order To:

PrP computergraphics

901 Cherrydale Street
Lake Charles, LA. 70605

Mark An X By Programs Ordered
Scribble Biorhythm Blackjack

Single Program _____ 7.95 each
Any 2 Programs _____ 6.95 each
Any 3 Programs _____ 5.95 each

Please Send Programs Marked Above To:

Name _____

Street _____

City _____ St. _____ Zip _____

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

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

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

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

Checkbook Balancing

Ah-ha! That's already in your *User's Reference Guide*, page 182 [page III-22 in the new TI-99/4A *User's Reference Guide*]. Just key that program in and add your own embellishments to make it *your* program. I like to take advantage of TI's color and sound to enhance a program, so let's add a little color at the beginning. Add

```
115 GOSUB 500
```

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

```
100 REM CHECKBOOK BALANCE
110 CALL CLEAR
115 GOSUB 500
120 INPUT "BANK BALANCE: ";BALANCE
130 DISPLAY "ENTER EACH OUTSTANDING"
140 DISPLAY "CHECK NUMBER AND AMOUNT."
150 DISPLAY
160 DISPLAY "ENTER A ZERO FOR THE"
170 DISPLAY "CHECK NUMBER WHEN FINISHED."
180 DISPLAY
200 INPUT "CHECK NUMBER: ";CNUM
210 IF CNUM=0 THEN 250
220 INPUT "CHECK AMOUNT: ";CAMT
230 CTOTAL=CTOTAL+CAMT
240 GOTO 200
250 DISPLAY "ENTER EACH OUTSTANDING"
260 DISPLAY "DEPOSIT AMOUNT."
270 DISPLAY
280 DISPLAY "ENTER A ZERO AMOUNT"
290 DISPLAY "WHEN FINISHED."
300 DISPLAY
320 INPUT "DEPOSIT AMOUNT? ";DAMT
330 IF DAMT=0 THEN 360
340 DTOTAL=DTOTAL+DAMT
350 GOTO 320
360 NBAL=BALANCE-CTOTAL+DTOTAL
370 DISPLAY "NEW BALANCE= ";NBAL
380 INPUT "CHECKBOOK BALANCE: ";CBAL
390 DISPLAY "CORRECTION= ";NBAL-CBAL
400 END
500 CALL CHAR(96,"0")
510 CALL CHAR(97,"0000FF")
520 CALL CHAR(98,"8618433402CC61")
530 CALL COLOR(9,13,11)
540 CALL HCHAR(10,6,96,18)
550 CALL VCHAR(11,6,96,6)
560 CALL HCHAR(11,7,98,5)
570 CALL HCHAR(11,12,96,4)
580 CALL HCHAR(11,16,97,8)
590 CALL HCHAR(12,7,96,17)
600 CALL HCHAR(13,7,97,17)
610 CALL HCHAR(13,18,96,2)
620 CALL HCHAR(14,7,97,17)
630 CALL HCHAR(15,7,96,17)
640 CALL HCHAR(16,7,96,9)
650 CALL HCHAR(16,16,97,8)
660 RETURN
```

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

Dogfight ... from p. 26

1500-1560 Procedure if Plane #2 is shot.
1570-1630 Increments score for Player 1 and prints scores.
1640 Ends game if either player's score is 10.
1650-1680 If score is less than 10, players may choose to try again or stop.
1690-1720 Prints final score and ending remarks.

```
660 IF K1=12 THEN GOSUB 990
670 IF S2=0 THEN 820
680 IF K2<>5 THEN 720 ELSE IF P1>3 AND P1<8 THEN
    P1=P1-1 ELSE IF P1<3 OR P1=8 THEN P1=P1+1

690 IF P1=0 THEN P1=8
700 IF P1=9 THEN P1=1
710 GOTO 800
720 IF K2<>0 THEN 760 ELSE IF P1>3 AND P1<7 THEN
    P1=P1+1 ELSE IF P1<3 OR P1=8 TH
EN P1=P1-1
730 IF P1=0 THEN P1=8
740 IF P1=9 THEN P1=1
750 GOTO 800
760 IF K2<>2 THEN 780 ELSE IF P1>5 THEN P1=P1-1
    ELSE IF P1<5 AND P1>1 THEN P1=P1+1

770 GOTO 800
780 IF K2<>3 THEN 810 ELSE IF P1<5 AND P1>1 THEN
    P1=P1-1 ELSE IF P1>5 THEN P1=P1+1
```

```
790 IF P1=9 THEN P1=1
800 CALL PATTERN(#2,(P1*4)+92):: ON P1 GOSUB 1110,
    1120,1130,1140,1150,1160,1170,
1180 :: GOTO 820
810 IF K2=1 THEN GOSUB 1190
820 CALL COINC(#2,#3,3*V,PC):: IF PC=-1 THEN 1500
830 CALL COINC(#1,#4,3*V,PC):: IF PC=-1 THEN 1350
840 CALL COINC(#1,#2,2.5*V,PC):: IF PC=-1 THEN
    1220
850 CALL SOUND(-4250,110,2,220,2,-8,2)
860 IF SH1=0 THEN 880
870 CALL POSITION(#3,B3,B4):: IF B3<6 OR B3>186 OR
    B4<8 OR B4>250 THEN CALL DELSPRITE(#3)
    :: SH1=0
880 IF SH2=0 THEN 900
890 CALL POSITION(#4,C3,C4):: IF C3<6 OR C3>186 OR
    C4<8 OR C4>250 THEN CALL DELS
PRITE(#4):: SH2=0
900 GOTO 500
910 CALL MOTION(#1,0,V):: RETURN
920 CALL MOTION(#1,-V*.6,V*.6):: RETURN
930 CALL MOTION(#1,-V,0):: RETURN
940 CALL MOTION(#1,-V*.6,-V*.6):: RETURN
950 CALL MOTION(#1,0,-V):: RETURN
960 CALL MOTION(#1,V*.6,-V*.6):: RETURN
970 CALL MOTION(#1,V,0):: RETURN
980 CALL MOTION(#1,V*.6,V*.6):: RETURN
990 IF SH1=1 THEN RETURN
```

Continued on p. 36

Line 190 and change Line 240 to GOTO 200; and delete Line 310 and change Line 350 to GOTO 320.

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

Name and Address File

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

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

Recipe Conversions

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

```
100 REM RECIPE CONVERSION
110 CALL CLEAR
120 DIM AMT(20),M$(20),ING$(20)
130 PRINT "ENTER NUMERICAL AMOUNT"
140 PRINT "(USE DECIMAL IF FRACTION),"
150 PRINT "THEN ENTER MEASURE (C,TSP),"
160 PRINT "THEN ENTER INGREDIENT."
170 PRINT "ENTER '0' TO END RECIPE.":
180 INPUT "AMOUNT: ":AMT(I)
190 IF AMT(I)<=0 THEN 250
200 INPUT "MEASURE: ":M$(I)
210 INPUT "INGREDIENT: ":ING$(I)
220 I=I+1
230 PRINT
240 GOTO 180
250 CALL CLEAR
260 FOR J=0 TO I-1
270 PRINT AMT(J);M$(J);" ";ING$(J)
280 NEXT J
290 PRINT "MULTIPLY BY WHAT NUMBER"
300 INPUT "OR DECIMAL FRACTION? ":F
310 CALL CLEAR
320 PRINT F;"TIMES ORIGINAL RECIPE":
330 FOR J=0 TO I-1
340 PRINT F*AMT(J);M$(J);" ";ING$(J)
350 NEXT J
360 PRINT "CONVERT AGAIN? (Y/N)"
370 CALL KEY(0,K,B)
380 IF K=B9 THEN 290
390 IF K<>7B THEN 370
400 END
```

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

AMOUNT: 2
MEASURE: CUPS
INGREDIENT: SHORTENING

AMOUNT: 2
MEASURE: CUPS
INGREDIENT: SUGAR

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

AMOUNT: 1.5
MEASURE: TSP.
INGREDIENT: ALMOND EXTRACT

AMOUNT: 2
MEASURE: TSP.
INGREDIENT: BAKING POWDER

AMOUNT: 4
MEASURE: CUPS
INGREDIENT: FLOUR

AMOUNT: 4
MEASURE: DOZ.
INGREDIENT: ALMONDS

AMOUNT: 0

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

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

Inventory

There are all types of ways to approach an inventory program. Ten programmers will come up with ten different programs. One possibility is to use the "Electronic Home Secretary" program. Here is one Regena method of a household

Dogfight . . . from p. 35

```

1000 CALL POSITION(#1,B1,B2)
1010 P3=P :: SP=3 :: A1=B1 :: A2=B2 :: SH1=1
1020 ON P3 GOTO 1030,1040,1050,1060,
1070,1080,1090,1100
1030 CALL SPRITE(#SP,128,CO2,A1,A2,0,V*2):: RETURN
1040 CALL SPRITE(#SP,128,CO2,A1,A2,-V*1.6,V*1.6)
:: RETURN
1050 CALL SPRITE(#SP,128,CO2,A1,A2,-V*2,0):: RETURN
1060 CALL SPRITE(#SP,128,CO2,A1,A2,-V*1.6,-V*1.6)
:: RETURN
1070 CALL SPRITE(#SP,128,CO2,A1,A2,0,-V*2):: RETURN
1080 CALL SPRITE(#SP,128,CO2,A1,A2,V*1.6,-V*1.6)
:: RETURN
1090 CALL SPRITE(#SP,128,CO2,A1,A2,V*2,0):: RETURN
1100 CALL SPRITE(#SP,128,CO2,A1,A2,V*1.6,V*1.6)
:: RETURN
1110 CALL MOTION(#2,0,V):: RETURN
1120 CALL MOTION(#2,-V*.6,V*.6):: RETURN
1130 CALL MOTION(#2,-V,0):: RETURN
1140 CALL MOTION(#2,-V*.6,-V*.6):: RETURN
1150 CALL MOTION(#2,0,-V):: RETURN
1160 CALL MOTION(#2,V*.6,-V*.6):: RETURN
1170 CALL MOTION(#2,V,0):: RETURN
1180 CALL MOTION(#2,V*.6,V*.6):: RETURN
1190 IF SH2=1 THEN RETURN
1200 CALL POSITION(#2,C1,C2)
1210 A1=C1 :: A2=C2 :: P3=P1 :: SP=4 :: SH2=1
:: GOTO 1020
1220 CALL MOTION(#1,0,0,#2,0,0):: CALL POSITION(#1,
PO1,PO2,#2,PO3,PO4)
1230 CALL SPRITE(#1,132,10,PO1,PO2,20,0,#2,132,10,
PO3,PO4,20,0)
1240 CALL POSITION(#1,CR1,CR2)
1250 CALL POSITION(#2,CR3,CR4)
1260 IF CR1>186 OR CR3>186 THEN CALL DELSPRITE
(#1,#2)
1270 CALL SOUND(-1000,110,2,220,2,-7,0)
1280 IF CR1<>0 OR CR3<>0 THEN 1240
1290 CALL DELSPRITE(#1,#2,#3,#4)
1300 CALL SCREEN(6)
1310 DISPLAY AT(5,13):"CRAAAASSSH"
1320 DISPLAY AT(10,5):"BOTH TEAMS CRASHED."
1330 DISPLAY AT(12,5):"NO POINTS AWARDED."
1340 GOTO 1640
1350 CALL DELSPRITE(#3,#4)
1360 CALL MOTION(#1,0,0):: CALL POSITION(#1,
PO1,PO2)
1370 CALL SPRITE(#1,132,10,PO1,PO2,20,0)
1380 CALL SOUND(-1000,440,2,220,2,-4,0)
1390 CALL POSITION(#1,CR1,CR2)
1400 IF CR1<186 THEN 1380
1410 CALL DELSPRITE(#1,#2)
1420 RED=RED+1
1430 CALL SCREEN(6)
1440 DISPLAY AT(5,1):PLA2*;" HAS WON THE BATTLE."
1450 DISPLAY AT(9,3):PLA2*;" SCORE IS :"
1460 DISPLAY AT(9,26):RED

```

Continued on p. 37

inventory. Use DATA statements and enter each item in the following order: room number, item, cost.

```

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

```

Only a few items in a few rooms are shown here to illustrate the logic of the program. You will probably want to include more rooms, the year purchased, and perhaps depreciation, replacement value, and a few other remarks. And don't forget to add titles to make the information more meaningful. You can use this program idea for any kind of inventory from food storage to retail products. Extended

BASIC allows nice formatting of output (with USING or IMAGE) so the numbers line up. It is also possible in regular BASIC by testing the length or the size of the numbers and printing accordingly.

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

```

100 REM SORTING
110 REM N=NUMBER OF ITEMS
120 DIM A(50)
130 CALL CLEAR
140 INPUT "N= ";N
150 PRINT
160 FOR I=1 TO N
170 INPUT "A"&STR*(I)&" = ";A(I)
180 NEXT I
190 PRINT :::
200 LIM=N-1
210 SW=0
220 FOR I=1 TO LIM
230 IF A(I)<=A(I+1) THEN 290
240 AA=A(I)
250 A(I)=A(I+1)
260 A(I+1)=AA
270 SW=1
280 LIM=I
290 NEXT I
300 IF SW=1 THEN 210
310 FOR I=1 TO N
320 PRINT A(I)
330 NEXT I
340 PRINT
350 END

```

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

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

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

Continued on p. 83

Dogfight . . . from p. 25

Lines 190-280 define special graphics characters four at a time. Then Line 290, CALL MAGNIFY(3) indicates that sprites will actually consist of four regular size characters, and only the first character number needs to be specified.

DISPLAY AT(x, y) prints messages on the screen starting at row x and column y. ACCEPT AT(x1, y1) receives input at a certain place. SIZE(8) limits the players' names to 8 characters to prevent a messy score chart later.

CALL SPRITE(#1, 96, 2, 120, 20, 0, 5) defined Sprite #1 (the first plane) as characters 96, 97, 98, and 99, black, starting in dot-row 120 and dot-column 20, and going zero velocity up or down and to the right at velocity 5. More than one sprite may be defined at a time, as

in Line 310.

CALL DELSPRITE(#1) deletes Sprite #1, and more than one sprite may be deleted at a time. This statement is used when the planes are hit or crash or when the bullet leaves the edge of the screen.

Complex IF-THEN-ELSE statements are used in Lines 530-780 to determine what direction the plane is headed. CALL PATTERN then draws the plane depending on its heading; all other characteristics of the sprite remain the same.

CALL POSITION(#1, B1, B2) in Line 1000 returns the row and column position of Sprite #1 so the bullet can be shot from that same position.

CALL MOTION can change the motion of a sprite without affecting other characteristics.

CALL COINC(#2, #3, 3*V, PC) determines if Sprite #2 (second plane) and

Sprite #3 (bullet from first plane) are coincident within a tolerance of 3*V. If so, a value of -1 is returned for PC. Coincidence is reported only when the CALL COINC statement is actually executed during the program, so at greater velocities of the sprites coincidence will not be detected if the program is busy somewhere else.

One solution to this problem is to CALL COINC more often (which would slow down the response time in turning the plane or shooting). Another solution is to increase the tolerance level. In this program the tolerance level is increased as the velocity increases, but if the tolerance level is too great, the planes will crash even if they are not touching. Another solution is to decrease the time the program is executing other statements. One way to decrease execution time is to minimize the logic. This is done by allowing only one bullet at a time per plane—sorry, no machine guns—so the program only needs to keep track of four sprites. Using multi-statement lines also increases the speed of execution. CALL MOTION (#1, 0, V):: RETURN is faster than using two separate lines for the statements.

Although kept to a minimum, the coincidence problem does still exist in this program: You may notice that once in a while a bullet will pass undetected right through a plane. You'll just have to accept it as one of the hazards of the game, or visualize a three-dimensional situation—a bullet passing directly over or under the plane but at a different altitude. Also, once in a while a bullet won't disappear at the edge of the screen, but will "fwrap." Just consider it a stray bullet—a frequent occurrence in a real dogfight.

Dogfight . . . from p. 36

```
1470 DISPLAY AT(11,3):PLA1$;" SCORE IS :"  
1480 DISPLAY AT(11,26):BLUE  
1490 GOTO 1640  
1500 CALL DELSPRITE(#3,#4)  
1510 CALL MOTION(#2,0,0):: CALL POSITION(#2,PD1,PD2)  
1520 CALL SPRITE(#2,132,10,PD1,PD2,20,0)  
1530 CALL SOUND(-1000,440,2,220,2,-4,0)  
1540 CALL POSITION(#2,CR3,CR4)  
1550 IF CR3<186 THEN 1530  
1560 CALL DELSPRITE(#2,#1)  
1570 BLUE=BLUE+1  
1580 CALL SCREEN(6)  
1590 DISPLAY AT(5,1):PLA1$;" HAS WON THE BATTLE."  
1600 DISPLAY AT(8,3):PLA1$;" SCORE IS :"  
1610 DISPLAY AT(8,26):BLUE  
1620 DISPLAY AT(10,3):PLA2$;" SCORE IS :"  
1630 DISPLAY AT(10,26):RED  
1640 IF RED=10 OR BLUE=10 THEN 1690  
1650 DISPLAY AT(20,3):"WANT TO TRY AGAIN? (Y/N)"  
1660 PC=0 :: SH1=0 :: SH2=0 :: P=1 :: P1=5  
1670 CALL KEY(0,KEY,S)  
1680 IF KEY=89 THEN 370 ELSE IF KEY<>78 THEN 1670  
1690 IF BLUE>RED THEN PRINT PLA1$;" HAS WON THE  
WAR.":"SORRY, ":"PLA2$ :: GOTO 1720  
1700 IF RED>BLUE THEN PRINT PLA2$;" HAS WON THE  
WAR.":"SORRY, ":"PLA1$ :: GOTO 1720  
1710 PRINT "THE WAR HAS ENDED IN A TIE. BOTH SIDES  
ARE NOW AT PEACE."  
1720 END
```

Third-Party Pascal . . . from p. 31

To summarize, the UCSD system is a powerful, yet easy-to-use operating system that brings the home computer user much sophistication. Although I have pointed out several relatively minor flaws in the third-party development system, I consider them to be related to "teething problems." My feeling is that the development system is not for the neophyte computer user; exposure to and familiarity with other operating systems and text editors facilitates understanding of the UCSD system. However, it is my opinion that the expenditure of time and effort required to learn the UCSD system is more than worthwhile. Those users who opt for the production version of UCSD Pascal will find it difficult to revert back to BASIC! The incorporation of UCSD Pascal into the TI-99/4 is a positive step and it seems highly probable that more such steps will follow as the UCSD P-System supports not only Pascal, but FORTRAN, COBOL and a BASIC compiler.

Book Review . . . from p. 19

required for multiple products. An equipment maintenance scheduling program employs two sequential files for recording, updating, and displaying equipment maintenance schedules. Other programs include computations for production lot sizes and costs, plus comparison and analysis of production alternatives.

If all this sounds too good to be true for \$11.00, bear in mind that program conversion is a more difficult task than in the case of the previous two books, and will require some ingenuity on the reader's part.

Note:

Space in this issue did not permit a full treatment of the conversions necessary. Mr. Clulow's complete set of conversion subroutines will appear in the Nov/Dec issue. We suggest that interested readers first get acquainted with the book.

MOVING?



Don't
Miss Out
On Any
Issues Of
99'er Magazine.

Send us a Change-of-Address Card
(available at any Post Office)
6-8 weeks prior to the move.

Be sure to include both the old &
new address, plus the alphanumeric
code above your name on the mail-
ing label.



SOFTWARE FOR THE TI 99/4 COMPUTER

Fast, well, structured programs which fully exploit the potential of the TI computer for less than you expect to pay. Here is just a short list of what we have:

Interactive graphics
-a fast way to draw high resolution graphics

Speech adder
-expands the TI's speech capacity to include most of the English language

Music synthesizer

High resolution games which test the limits of your skills

Send for free brochure-

Norton Software
Box 575
Picton, Ontario
K0K 2T0

Inventory, Order Entry, Sales, & Mailing List For Your TI-99/4

Interactive Programming
Allows Maximum Flexibility for Customizing
the Software to *Your* Specific Needs.

Each program available separately:

- Inventory \$44.95 cassette; \$49.95 disk
- Order Entry " "
- Sales " "
- Mailing List " "

Interactive packages — disk-based only:

- Inventory - Sales \$89.95
- Inventory - Order Entry 89.95
- Order Entry - Mailing List 89.95

Versions in TI BASIC & Extended BASIC

All 4 programs \$189.95

(FREE library storage case included)

Add \$2.50 for postage/handling.
California orders, add 6% tax.

To order, or for more information:

Paul Yates Computer Services
4037 Johnson Drive
Oceanside, CA 92054
Tel. (714)-758-1633

Interactive Forms ... from p. 13

```

370 YES=1
380 NO=0
390 REM CENTRAL CONTROL MENU
400 CALL CLEAR
410 PRINT "MAKE A CHOICE--"
420 PRINT
430 PRINT " 1 - LOAD NEW FORM FILE"
440 PRINT " 2 - FILL OUT SAME FORM"
450 PRINT " 3 - PRINT COPIES"
460 PRINT " 4 - TERMINATE"
470 PRINT
480 INPUT CHOICE
490 IF 0<CHOICE<5 THEN 520
500 PRINT "ERROR, TRY AGAIN..."
510 GOTO 410
520 ON CHOICE GOSUB 550,840,1340,3010
530 GOTO 400
540 REM 1 - LOAD NEW FORM FILE SUBROUTINE
550 CALL CLEAR
560 PRINT "ENTER DEVICE:"
570 PRINT "      1 FOR CS1"
580 PRINT "      2 FOR DISK"
590 INPUT DEVICE
600 IF (DEVICE<1)+(DEVICE>2)=-1 THEN 560
610 IF DEVICE=2 THEN 690
620 OPEN #4:"CS1",INTERNAL,INPUT,FIXED 192
630 INPUT #4:X
640 FOR I=1 TO X STEP 2
650 INPUT #4:A$(I),A$(I+1)
660 NEXT I
670 CLOSE #4
680 RETURN
690 REM -- READ FROM DISK
700 PRINT "ENTER WHICH DISK, 1-3?"
710 INPUT DISKNO
720 IF (DISKNO<1)+(DISKNO>3)=-1 THEN 700
730 PRINT "ENTER FILENAME:"
740 INPUT FILENAME$
750 IF (LEN(FILENAME$)<1)+(LEN(FILENAME$)>10)=-1
    THEN 730
760 OPEN #5:"DSK"&STR$(DISKNO)&". "&FILENAME$,
    INPUT,SEQUENTIAL,INTERNAL,VARIABLE 132
770 INPUT #5:X
780 FOR I=1 TO X

```

```

790 INPUT #5:A$(I)
800 NEXT I
810 CLOSE #5
820 RETURN
830 REM 2 - FILL OUT SAME FORM
840 REM MAIN SCANNER- LOOKS FOR COMMANDS
850 CALL CLEAR
860 TERM=1
870 FOR I=1 TO X
880 REM >>> COMMAND LINE?>>>
890 IF SEG$(A$(I),1,2)=BANG$ THEN 910 ELSE 1310
900 REM >>> IF SO, START OF REPEAT SEQUENCE?>>>
910 IF SEG$(A$(I),3,1)=AMPERSAND$ THEN 920 ELSE 990
920 IF REPEAT=YES THEN 990
930 REM >>> INITIALIZE THE REPEAT SEQUENCE.>>>
940 REPEAT=YES
950 MAXREPS=VAL(SEG$(A$(I),4,PDS(A$(I),SEMICOLON$,
    4)-4))
960 REPSTART=I
970 REPS=0
980 REM LINE PARSER
990 FOR J=3 TO LEN(A$(I))
1000 P$=SEG$(A$(I),J,1)
1010 IF P$=QUOTE$ THEN 1030 ELSE 1060
1020 REM >>> GO SUBROUTINE "COMMENT" >>>
1030 GOSUB 32767
1040 J=K
1050 GOTO 1300
1060 IF P$=COLON$ THEN 1080 ELSE 1140
1070 REM >>> GO SUBROUTINE "FIELDINPUT" >>>
1080 IF SEG$(A$(I),J+1,1)=RIGHTARROW$ THEN 1090
    ELSE 1110
1090 RIGHTJUSTIFY=YES
1100 J=J+1
1110 GOSUB 32767
1120 J=K
1130 GOTO 1300
1140 IF P$=OPENPAREN$ THEN 1160 ELSE 1190
1150 REM >>> GO SUBROUTINE "MATH TERM" >>>
1160 GOSUB 1860
1170 J=K
1180 GOTO 1300
1190 IF P$=AMPERSAND$ THEN 1210 ELSE 1300
1200 REM >>> IS IT THE 1ST AMPERSAND? >>>
1210 IF J=3 THEN 1300

```

Continued on p. 75

DATA SYSTEMS

Ringwielders Lair	\$20.00 (disk)
(Extended BASIC and Memory Expansion needed)	
Denali Data Adaptor	\$20.00
Atari Joysticks	\$16.50
Le Shhhtick	\$34.00
MX70/80 ribbons	\$10.50
Disks	\$ 2.85

For a catalog of all of our products,
send \$1.00 (refundable on any order).

New Programs Coming from DATA SYSTEMS

- Legal Will Writer
- Empirical Earth
- Treasure Hunt

Minimum order is \$15.00

DATA SYSTEMS
2214 W. IOWA
Chicago, IL 60622
Tel. (312) 235-2699

QUALITY SOFTWARE FOR TI 99/4

BASIC LANGUAGE PROGRAMS

SPACE BATTLE -2056 You are the Earthfleets last hope, outnumbered you must rely on cunning and steely nerves to outwit the enemy!

\$13.95

SKI - An exciting downhill race game. Hundreds of courses and different skiing conditions test your skill.

\$14.95

LASER SHIELD - Shield the cities from incoming missiles with your laser. Fast reflexes and presence of mind are required.

\$14.95

LAND ON MARS - Landing is only half the fun. You must take off and dock with the orbiter. Hundreds of landscapes to challenge you. Dragon option included!

\$12.95

BARNYARD FUN - A memory building game for children ages four and up. LOTS OF FUN!!

\$14.95

EXTENDED BASIC PROGRAMS

MR. FROG - Three educational games for ages 3 and up. Great graphics, music, and even a race against the computer.

\$24.95

METEOR SHOWER - Only the best pilots can survive for long. An arcade type game of fast action.

\$9.95

All programs on cassettes, complete instructions included.
Plus \$1.50 shipping/handling, Mn. residents add 5% sales tax.
To order or for more information:

American Software Design & Distribution Co.
P.O. Box 46
Cottage Grove, Mn. 55016

Space War ... from p. 28

```

2230 IF S<>0 THEN 2480
2240 GOTO 2160
2250 IF K1<>11 THEN 2320
2260 YPR=YPR-5
2270 IF YPR<0 THEN 6270
2280 E=C
2290 F=D
2300 G=2
2310 GOTO 3470
2320 YPR=YPR-1
2330 IF YPR<0 THEN 6270
2340 CALL GCHAR(C,D,P)
2350 CALL HCHAR(C,D,32)
2360 CALL SOUND(200,-6,0,440,0)
2370 G=1
2380 IF K1+1=1 THEN 4750
2390 IF K1=5 THEN 4830
2400 IF K1=3 THEN 4910
2410 IF K1=2 THEN 4990
2420 IF K1=6 THEN 5070
2430 IF K1=4 THEN 5180
2440 IF K1=14 THEN 5290
2450 IF K1=15 THEN 5400
2460 CALL HCHAR(C,D,P)
2470 GOTO 2200
2480 IF K<>18 THEN 2550
2490 BPR=BPR-5
2500 IF BPR<0 THEN 6300
2510 E=A
2520 F=B
2530 G=1
2540 GOTO 3470
2550 BPR=BPR-1
2560 IF BPR<0 THEN 6300
2570 CALL GCHAR(A,B,P)
2580 CALL HCHAR(A,B,32)
2590 CALL SOUND(200,-6,0,440,0)
2600 G=2
2610 IF K=5 THEN 2710
2620 IF K+1=1 THEN 2790
2630 IF K=3 THEN 2870
2640 IF K=2 THEN 2950
2650 IF K=6 THEN 3030

```

```

2660 IF K=4 THEN 3140
2670 IF K=14 THEN 3250
2680 IF K=15 THEN 3360
2690 CALL HCHAR(A,B,P)
2700 GOTO 2200
2710 A=A-1
2720 IF A<>0 THEN 2740
2730 A=24
2740 CALL GCHAR(A,B,P)
2750 GOSUB 6140
2760 IF V=1 THEN 5510
2770 CALL HCHAR(A,B,96)
2780 GOTO 2160
2790 A=A+1
2800 IF A<>25 THEN 2820
2810 A=1
2820 CALL GCHAR(A,B,P)
2830 GOSUB 6140
2840 IF V=1 THEN 5510
2850 CALL HCHAR(A,B,97)
2860 GOTO 2160
2870 B=B+1
2880 IF B<>33 THEN 2900
2890 B=1
2900 CALL GCHAR(A,B,P)
2910 GOSUB 6140
2920 IF V=1 THEN 5510
2930 CALL HCHAR(A,B,98)
2940 GOTO 2160
2950 B=B-1
2960 IF B<>0 THEN 2980
2970 B=32
2980 CALL GCHAR(A,B,P)
2990 GOSUB 6140
3000 IF V=1 THEN 5510
3010 CALL HCHAR(A,B,99)
3020 GOTO 2160
3030 A=A-1
3040 B=B+1
3050 IF A<>0 THEN 3070
3060 A=24
3070 IF B<>33 THEN 3090
3080 B=1
3090 CALL GCHAR(A,B,P)
3100 GOSUB 6140

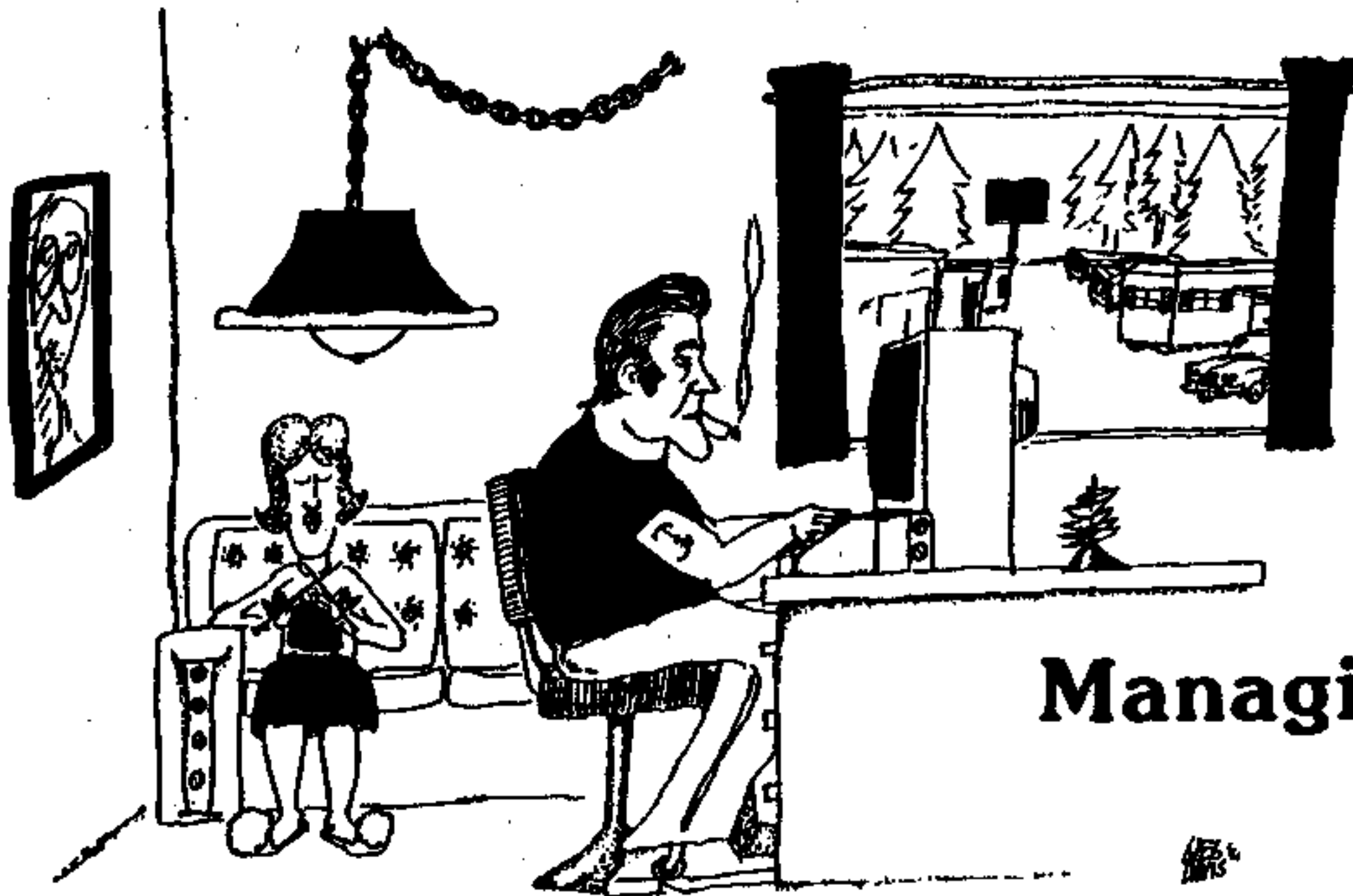
```

```

3110 IF V=1 THEN 5510
3120 CALL HCHAR(A,B,100)
3130 GOTO 2160
3140 A=A-1
3150 B=B-1
3160 IF A<>0 THEN 3180
3170 A=24
3180 IF B<>0 THEN 3200
3190 B=32
3200 CALL GCHAR(A,B,P)
3210 GOSUB 6140
3220 IF V=1 THEN 5510
3230 CALL HCHAR(A,B,101)
3240 GOTO 2160
3250 A=A+1
3260 B=B+1
3270 IF A<>25 THEN 3290
3280 A=1
3290 IF B<>33 THEN 3310
3300 B=1
3310 CALL GCHAR(A,B,P)
3320 GOSUB 6140
3330 IF V=1 THEN 5510
3340 CALL HCHAR(A,B,102)
3350 GOTO 2160
3360 A=A+1
3370 B=B-1
3380 IF A<>25 THEN 3400
3390 A=1
3400 IF B<>0 THEN 3420
3410 B=32
3420 CALL GCHAR(A,B,P)
3430 GOSUB 6140
3440 IF V=1 THEN 5510
3450 CALL HCHAR(A,B,103)
3460 GOTO 2160
3470 CALL GCHAR(E,F,Z)
3480 CALL SOUND(100,440,0,880,0)
3490 IF Z=96 THEN 3510
3500 IF Z<>104 THEN 3640
3510 FOR I=E-1 TO 1 STEP -1
3520 CALL GCHAR(I,F,P)
3530 GOSUB 6050
3540 CALL HCHAR(I,F,112)

```

Continued on p. 47



Personal Record Keeping: Managing A Mobile Home Park

By David G. Brader

First of all, this true story has a moral to it. So might as well get it out of the way now:

“Before Going To All The Work Of Writing A Program To Do A Job, Find Out If A TI Command Module Can Do The Job For You.”

The TI Command Modules are well written, almost totally error free, and have been engineered for ease of use by non-programmers. Let's talk about one of these little jewels:

The *Personal Record Keeping (PRK)* Command Module combined with your imagination is a very flexible and powerful tool. To fully utilize this power, however, your TI-99/4 system should include a printer. The TI Thermal Printer works well and is probably the easiest and least expensive to use, but I chose the more expensive route of an Epson MX-80 printer through the RS232 Interface. This gives me a bit more power (e.g., longer print lines) to the PRK's report formatting. For most applications, you will also need either a cassette recorder or the disk system to keep your data files.

Before trying to set up and work with a data file using the PRK, you should carefully read the manual that comes with the module and all the examples. When you have done that, take a break, come back a little later, and do it all again. That mild-mannered little PRK manual contains the answers to questions that will surely pop up when you start designing the solution to *your* problem. So keep it handy!

OK, now comes the real challenge. How do we decide that a problem can be solved using the PRK module? First, we must completely describe the problem. Second, we must break the prob-

lem down into subproblems or tasks. Third, we identify the tasks that can be performed by the PRK module. Fourth, we see if enough of the tasks can be handled by the PRK module to make it a reasonable solution to the overall problem. The rest of this article is about such a challenge and a way to solve it with the PRK module.

And Now For Our Story . . .

Recently I had a customer ask me how much it would cost to write a program for him. I told him that it is difficult to determine until the program is completed, but he could figure on \$15.00 per hour and a minimum of 10 hours. At this point, he decided to be brave and tackle it himself. Of course, I was curious, so I asked him what he wanted the TI Basic program to do.

He told me that he was the owner of a mobile home park. Each month he had to figure out the bill for each individual renter in the park. He wanted the TI-99/4 to do the work—saving his

time and decreasing the chance for errors. After thinking over this problem for a minute, I asked him for details of what he did to accomplish the job himself.

First, he walked around to each trailer space and copied the electric meter and gas meter readings into his notebook. A computer system could be designed to do this task but it would take extremely expensive peripheral hardware . . .

I asked him what else was in the notebook that he used for this job. He said the notebook contained all the previous electric and gas meter readings. It also contained miscellaneous charges for each renter, the electric and gas meter rental fees, the electric and gas rates, and the actual space rental fees. (It was obvious to me at this point that the computer could easily act as a notebook and store all that data on cassette tape or floppy disk.)

Next, he sat down at his desk with the notebook, pencil, paper, and a calculator. For each trailer space, he performed the following math calculations:

GAS BILL	=	(CUR. GAS METER - PREV. GAS METER)	X	GAS RATE
		+ GAS METER USE FEE		
ELECTRIC BILL	=	(CUR. KWH METER - PREV. KWH METER)	X	KWH RATE
		+ KWH METER USE FEE		
TOTAL BILL	=	GAS BILL + ELECTRIC BILL + MISC. CHARGES		
		+ SPACE RENTAL FEE		

Each of the bill items is then recorded in the notebook for bookkeeping purposes, and a statement is made out for each tenant. Finally, he figures the total gas bill, the total electric bill, and the total income for the trailer park. (The computer could also record all the bill items, perform the calculations, and make out statements.)

After reading the PRK manual a couple of times, I noted it offered the following capabilities:

1. Maintains files of data in a structured fashion.
2. Allows adding to and updating data within the files.
3. Data structures can be rearranged by sorting in different ways.
4. Any data structure of a numerical nature could have math performed on it or between it and another data structure.
5. The data structures could be arranged in various ways and then printed as reports, lists, or what have you.

After further consideration, it appeared to me that most of the tasks related to the "trailer park monthly billing problem" could be solved using the TI-99/4 with the PRK module and a printer.

With my TI-99/4 fired up, PRK module installed, and manual in hand, I started toying around setting up the data structure of the file to use on this problem. I finally settled on the structure shown in Table 1.

Once a structure has been defined, you can't go back and change it without redefining the entire file structure. The best policy to follow to minimize this problem is to try out the file structure with a small amount of test data. It is a real pain to spend 4 hours entering real data into a file and then discover one odd-ball piece of data is too big! By the way, the smaller you define the width of a data item, the more data items you can keep in memory. As you can see, some care must be given to the design of the file structure.

Look at Table 2. It shows my three sample file "pages" of test data. This is the way the data would look after putting in the initial values. Now look at Table 3. The current utility meter readings and any miscellaneous charges have now been entered as the trailer park operator would do once a month.

At this point, I realized I had to figure out how to use the PRK module's "math transformations." That sounds pretty ominous doesn't it. But study of the manual revealed that it is nothing more than a set of simple equation "templates." These are shown on page 25 of the PRK manual and included here in Table 4. By substituting an item name for the appropriate "A, B, or C" in the equations, I built up a set of math transformations to figure out the electric, gas, and total bills. The PRK module guides you through this process nicely. The tailored set of math transformations is shown in Table 5 (in the order of execution).

Notice that the tailored math transformations set up the next month's "LAST GAS, CUR. GAS, LAST KWH, CUR. KWH" item fields after their data was utilized. This means that next month the user won't have to worry about moving the old "current" values to the "last" fields. I feel it is less confusing to the user to have zero in the "current" fields for the next month too. (That ought to get your imagination working!)

Now for the big test. Run the tailored math transformations on the file of test data and see if it works. The results are shown in Table 6. It is interesting to compare Table 6 to Table 3. The comparison better illustrates the work of these tailored equation templates.

With all the real data in the file, it probably takes about a half hour to an hour to process all the math. Sure that is slow, but it is *accurate*—and the manager can be eating dinner while the pro-

FILE STRUCTURE

ITEM	TYPE	WIDTH	DEC	DESCRIPTION OF ITEM
1 SPACE #	CHAR	4	0	The trailer space number
2 RENTER	CHAR	15	0	Name of the trailer space renter
3 LAST GAS	DEC	10	3	The previous gas meter reading
4 CUR. GAS	DEC	10	3	The current gas meter reading
5 GAS RATE	DEC	6	4	The cost of the gas per unit volume
6 LAST KWH	DEC	10	3	The previous electric meter reading
7 CUR. KWH	DEC	10	3	The current electric meter reading
8 RATE/KWH	DEC	6	4	The cost of the electricity per KWH
9 G.M. CHRG	DEC	5	2	The monthly gas meter charge
10 GAS TOTAL	DEC	6	2	The cost of gas used plus the meter charge
11 E.M. CHRG	DEC	6	2	The monthly electric meter charge
12 ELEC. TOTL	DEC	6	2	The cost of KWH used plus the meter charge
13 RENT/MO.	DEC	6	2	The trailer space monthly rental cost
14 MISC. CHRG	DEC	7	2	Any other charge (maybe damage to lot...)
15 MO. TOTAL	DEC	8	2	Grand total of gas, KWH, rent, and misc.

TABLE 1

FILE: RENTALS
DATE: 6/20/81
TITLE: TABLE 2

PAGE #	1	PAGE #	2	PAGE #	3
1. SPACE #	A-23	1. SPACE #	B-44	1. SPACE #	B-45
2. RENTER	SMITH, C.W.	2. RENTER	JONES, SAM	2. RENTER	HEIM, WILLIAM
3. LAST GAS	799992.465	3. LAST GAS	830.592	3. LAST GAS	990498.328
4. CUR. GAS	0.000	4. CUR. GAS	0.000	4. CUR. GAS	0.000
5. GAS RATE	.1130	5. GAS RATE	.1130	5. GAS RATE	.1130
6. LAST KWH	128176.263	6. LAST KWH	18841.212	6. LAST KWH	130392.249
7. CUR. KWH	0.000	7. CUR. KWH	0.000	7. CUR. KWH	0.000
8. RATE/KWH	.0231	8. RATE/KWH	.0231	8. RATE/KWH	.0231
9. G.M. CHRG	2.50	9. G.M. CHRG	2.50	9. G.M. CHRG	2.50
10. GAS TOTAL	0.00	10. GAS TOTAL	0.00	10. GAS TOTAL	0.00
11. E.M. CHRG	5.00	11. E.M. CHRG	5.00	11. E.M. CHRG	5.00
12. ELEC. TOTL	0.00	12. ELEC. TOTL	0.00	12. ELEC. TOTL	0.00
13. RENT/MO.	98.00	13. RENT/MO.	105.00	13. RENT/MO.	105.00
14. MISC. CHRG	0.00	14. MISC. CHRG	0.00	14. MISC. CHRG	0.00
15. MO. TOTAL	0.00	15. MO. TOTAL	0.00	15. MO. TOTAL	0.00

TABLE 2

FILE: RENTALS
 DATE: 6/20/81
 TITLE: TABLE 3

PAGE #	1	PAGE #	2	PAGE #	3
1. SPACE #	A-23	1. SPACE #	B-44	1. SPACE #	B-45
2. RENTER	SMITH, C.W.	2. RENTER	JONES, SAM	2. RENTER	HEIM, WILLIAM
3. LAST GAS	799992.465	3. LAST GAS	830.592	3. LAST GAS	990498.328
4. CUR. GAS	800124.732	4. CUR. GAS	891.947	4. CUR. GAS	990674.998
5. GAS RATE	.1130	5. GAS RATE	.1130	5. GAS RATE	.1130
6. LAST KWH	128176.263	6. LAST KWH	18841.212	6. LAST KWH	130392.249
7. CUR. KWH	131002.097	7. CUR. KWH	23622.609	7. CUR. KWH	134305.045
8. RATE/KWH	.0231	8. RATE/KWH	.0231	8. RATE/KWH	.0231
9. G.M. CHRG	2.50	9. G.M. CHRG	2.50	9. G.M. CHRG	2.50
10. GAS TOTAL	0.00	10. GAS TOTAL	0.00	10. GAS TOTAL	0.00
11. E.M. CHRG	5.00	11. E.M. CHRG	5.00	11. E.M. CHRG	5.00
12. ELEC. TOTL	0.00	12. ELEC. TOTL	0.00	12. ELEC. TOTL	0.00
13. RENT/MO.	98.00	13. RENT/MO.	105.00	13. RENT/MO.	105.00
14. MISC. CHRG	0.00	14. MISC. CHRG	17.50	14. MISC. CHRG	0.00
15. MO. TOTAL	0.00	15. MO. TOTAL	0.00	15. MO. TOTAL	0.00

TABLE 3

ITEM TRANSFORMATIONS

1. A = B
2. A = B + C
3. A = B - C
4. A = B x C
5. A = B / C
6. A = B ^ C
7. A = ABS(B)
8. A = LOG10(B)
9. A = LOGE(B)
10. A = EXP(B)
11. A = ATAN(B)
12. A = TAN(B)
13. A = SIN(B)
14. A = COS(B)
15. A = INT(B)
16. A = SGN(B)
17. A = PI
18. A = RND

(See the User's Reference Guide for a discussion of these functions.)

TABLE 4

TAILORED MATH TRANSFORMATIONS FOR TRAILER PARK BILLING

- LAST GAS = CUR. GAS - LAST GAS
- GAS TOTAL = LAST GAS X GAS RATE
- GAS TOTAL = G.M. CHRG + GAS TOTAL
- LAST GAS = CUR. GAS
- CUR. GAS = 0.000
- LAST KWH = CUR. KWH - LAST KWH
- ELEC. TOTL = LAST KWH X RATE/KWH
- ELEC. TOTL = E.M. CHRG + ELEC. TOTL
- LAST KWH = CUR. KWH
- CUR. KWH = 0.000
- MO. TOTAL = GAS TOTAL + ELEC. TOTL
- MO. TOTAL = MO. TOTAL + RENT/MO.
- MO. TOTAL = MO. TOTAL + MISC. CHRG

TABLE 5

FILE: RENTALS
 DATE: 6/20/81
 TITLE: TABLE 6

PAGE #	1	PAGE #	2	PAGE #	3
1. SPACE #	A-23	1. SPACE #	B-44	1. SPACE #	B-45
2. RENTER	SMITH, C.W.	2. RENTER	JONES, SAM	2. RENTER	HEIM, WILLIAM
3. LAST GAS	800124.732	3. LAST GAS	891.947	3. LAST GAS	990674.998
4. CUR. GAS	0.000	4. CUR. GAS	0.000	4. CUR. GAS	0.000
5. GAS RATE	.1130	5. GAS RATE	.1130	5. GAS RATE	.1130
6. LAST KWH	131002.097	6. LAST KWH	23622.609	6. LAST KWH	134305.045
7. CUR. KWH	0.000	7. CUR. KWH	0.000	7. CUR. KWH	0.000
8. RATE/KWH	.0231	8. RATE/KWH	.0231	8. RATE/KWH	.0231
9. G.M. CHRG	2.50	9. G.M. CHRG	2.50	9. G.M. CHRG	2.50
10. GAS TOTAL	17.45	10. GAS TOTAL	9.43	10. GAS TOTAL	22.46
11. E.M. CHRG	5.00	11. E.M. CHRG	5.00	11. E.M. CHRG	5.00
12. ELEC. TOTL	70.28	12. ELEC. TOTL	115.45	12. ELEC. TOTL	95.39
13. RENT/MO.	98.00	13. RENT/MO.	105.00	13. RENT/MO.	105.00
14. MISC. CHRG	0.00	14. MISC. CHRG	17.50	14. MISC. CHRG	0.00
15. MO. TOTAL	185.72	15. MO. TOTAL	247.38	15. MO. TOTAL	222.85

TABLE 6

cessing is being done. After dinner, he can start the PRK module printing out a report for each "file page" like shown in Table 6. Finally, after a nice relaxed dessert or brandy, he can cut apart the pages of the report and tape them in the appropriate spot of the form shown in Figure 1. There is a separate form page for each space in the trailer park. By using tape only at the top of the little PRK page, he can flip through previous months data (since the little pages are overlapping).

An Automatic "Manual" Feature

By using the "ANALYZE PAGES" mode of the PRK module, the total gas, electric, and monthly income can be read. After selecting the mode, select "5 SEE ITEM STATISTICS." Then chose the item such as "GAS TOTAL" and a display will appear like in Figure 2. The gas total for the entire trailer park is contained in the value of SUM. See what reading the manual reveals to you...

Before getting out of the PRK module, the data file must be saved on cassette tape or floppy disk for next time. Yes, the math transformations are also saved automatically at the same time.

Well, that's the story. I guess the only thing to add is that the *Personal Record Keeping* Command Module isn't the solution to *all* problems. But if you study it and experiment enough, you will be ready to wield this valuable and flexible tool when the appropriate situation arises. So go ahead... give the module a try, and I'll bet that soon you too will be witnessing a "Command" performance...

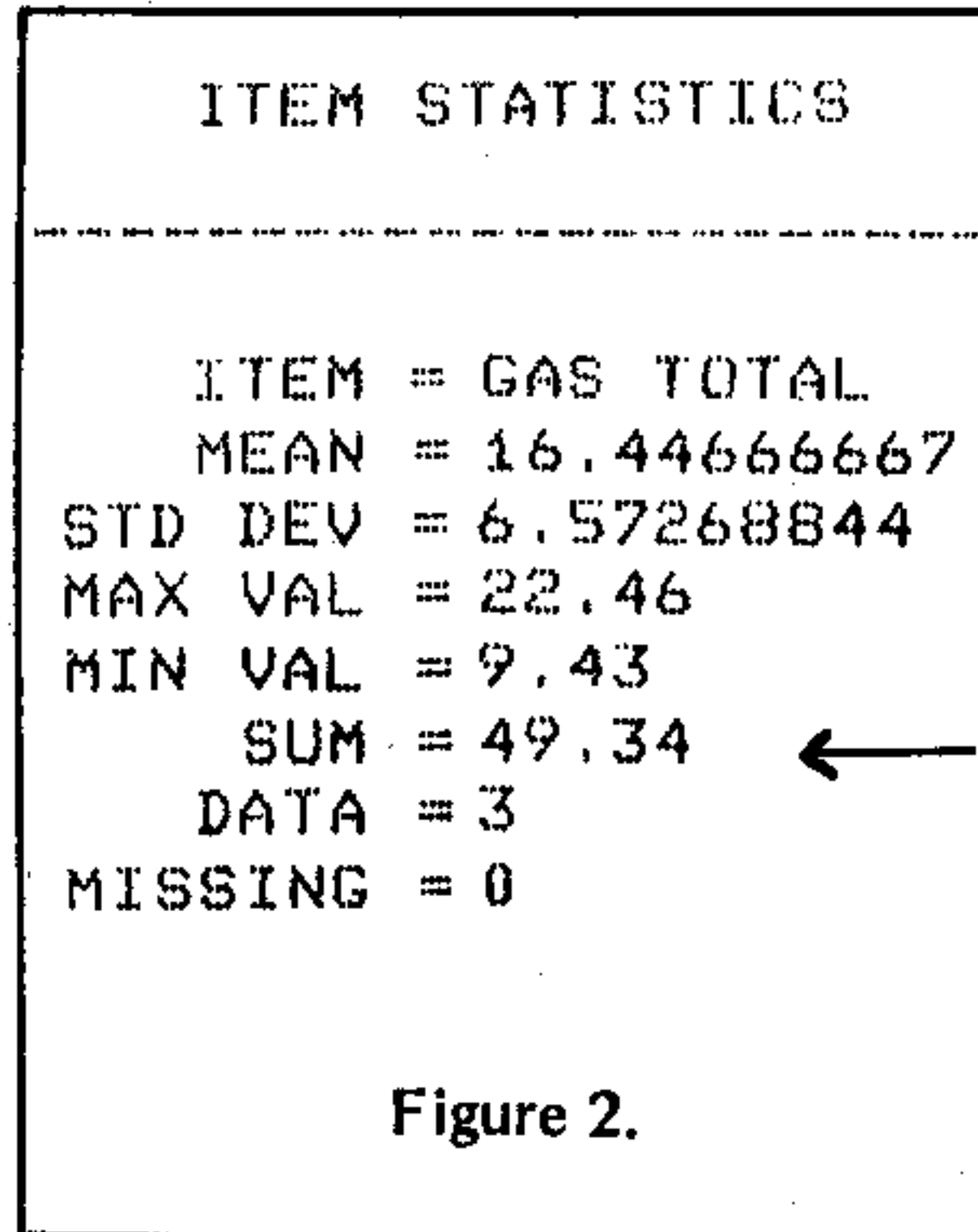


Figure 2.

Business & Professional Software

File Management System

300 records per disk, full upper and lower case, user-defined fields & printed formats. RAM sort (Shell-Hibbard), multi-key sub-file selection. **\$60**

Statistics Package

Create data files & selective sub-files. Generate descriptive statistics, parametric & non-parametric tests, linear regressions, & curve fitting. Tabular & graphic data output. **\$40**

For Order & Inquiries Contact:

Frederick Marin

Small Business/Professional
Computer Services

87 Mayfair Road
Warwick, RI 02888
Tel. (401) 461-6308

ATTENTION DEALERS!

If you don't offer Third-Party Software, you may be missing important sales. A customer's decision to purchase a 99/4 may be influenced by the availability of such supporting software. These are the type of sales that you cannot afford to lose!

C.S.C. is now offering their services to dealers. C.S.C. is a marketing firm specializing in the evaluation of Third-Party Software. Let C.S.C. "weed out" unsatisfactory software and provide a complete line of supporting software for the 99/4 and 99/4A. Over 50 programs are available in the areas of entertainment, education, business and home use.

Catalogs are available. Contact (405) 947-7402 today or write to:

C.S.C.
P.O. Box 1492
Bethany, OK 73008

MOBILE HOME PARK MONTHLY RECORD

Space number A-23

Year 1981

Dec -----
Nov -----
Oct -----
Sep -----
Aug -----
Jul -----

Month	PAGE #	1	
Jun			JUNE
May	1. SPACE #	A-23	
	2. RENTER	SMITH, C.W.	MAY
Apr	3. LAST GAS	800124.732	
	4. CUR. GAS	0.000	4, C.W.
Mar	5. GAS RATE	.1130	2.465
	6. LAST KWH	131002.097	
Feb	7. CUR. KWH	0.000	
	8. RATE/KWH	.0231	6.263
Jan	9. G.M. CHRG	2.50	
	10. GAS TOTAL	17.45	
	11. E.M. CHRG	5.00	
	12. ELEC. TOTL	70.28	
	13. RENT/MO.	98.00	
	14. MISC. CHRG	0.00	
	15. MO. TOTAL	185.72 ←	

15. MO. TOTAL 142.94 ←

Figure 1.

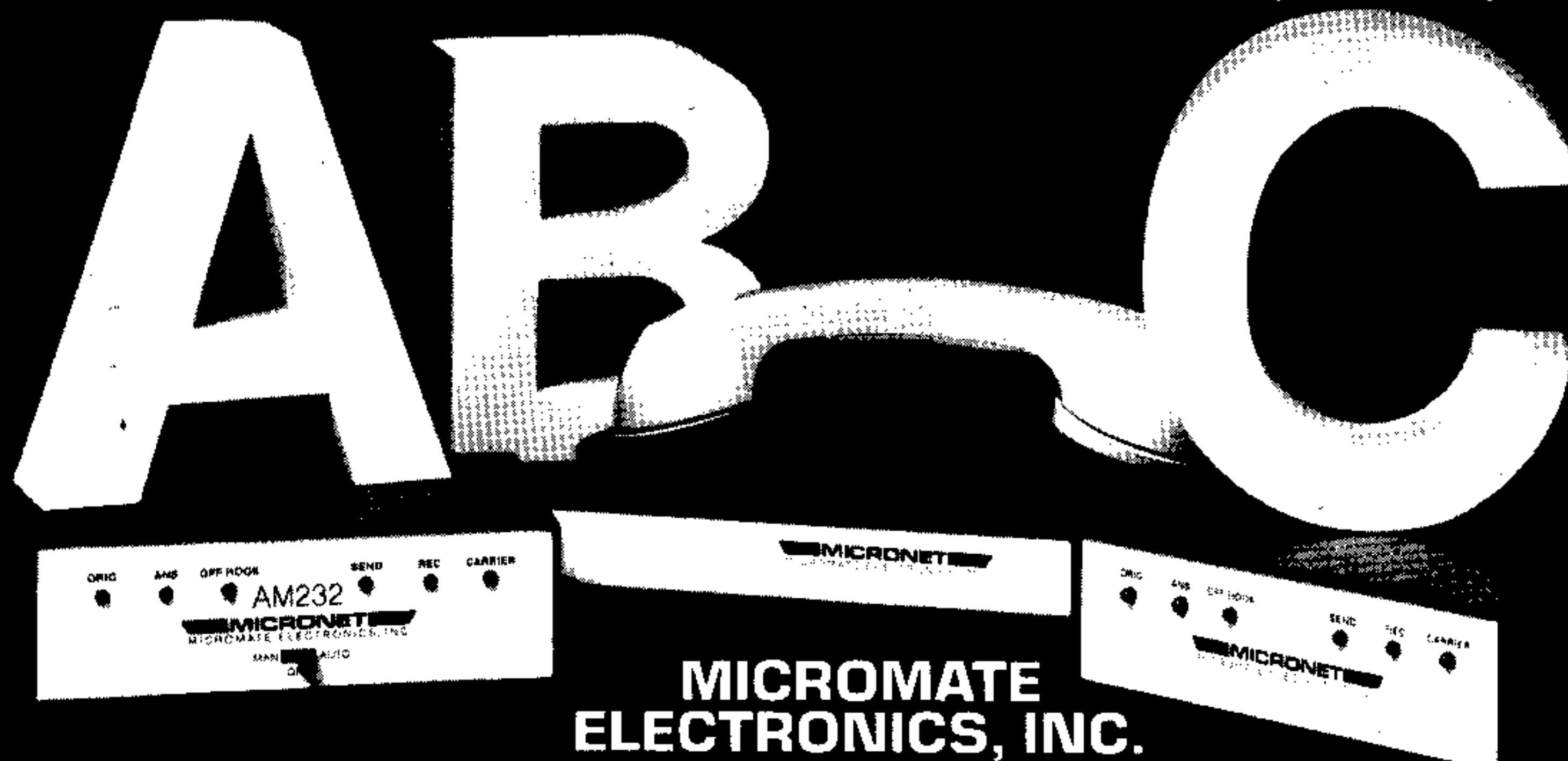
Select a modem for your computer ... easy as ABC.

CHOOSE A. The Smart One™, Model AM232, with its own built in microcontroller. Auto Dials, Auto Answers, Auto Modes (originate/answer). "Direct Connect" for any computer with an RS232 interface ... \$299.00.

CHOOSE B. The Acoustic Modem, Model AM11/A, for the Apple*. Comes complete with a powerful operating system. Supports 25 easy to use keyboard commands and operates interactively with your programs. No interface card required! ... \$179.00.

CHOOSE C. The "Direct Connect" Modem, Model AM11, for the Apple*. Comes complete with a powerful operating system. Supports Auto Dial, Auto Answer, 27 easy to use keyboard commands and operates interactively with your programs. No interface card required! ... \$289.00.

Be a part of the information revolution. It's fun, exciting, informative, educational. Call or write your order today. COD's accepted.



2094 Front Street • East Meadow, New York 11554 • Telephone 516.794.1072

All "Direct Connect" modems are FCC certified. *Registered trademark of Apple Computer, Inc.

Applesoft to TI BASIC ... from p. 24

NNN is the statement number of the needed element) will do the trick. Keep this weakness of APPLESOFT in mind whenever you see RESTORE followed by a number of READs in a APPLE program.

SCREEN COMMANDS

The APPLE has three modes of processing; TEXT mode and two different graphics modes. While in TEXT mode, the programmer has a number of commands which provide a wide range of control over the screen. As previously stated, the APPLE screen in this mode acts similar to the TI —except it starts at the top then works its way down to the bottom before scrolling. It also allows the programmer to do things like set the width of the print screen ("text window") and the length (number of lines) of the text window. Some of the most commonly encountered commands are:

CALL -936	clear the screen inside the text window
CALL -912	scrolls the text window up 1 line
CALL -868	clears the current line from the cursor to the right
HOME	same as TI's CALL CLEAR
POKE 32,L	sets left margin of window to L
POKE 33,W	sets width of window
POKE 34,T	sets top of window
POKE 35,B	sets bottom of screen
FLASH	start 'flashing' output from white letters on black to black letters on white and back again
INVERSE	reverses output to black letters on white
NORMAL	resets FLASH and INVERSE
POS(N)	get current horizontal column of the cursor (i.e., N will have column number 0-39)

To simulate FLASH or INVERSE, use TI BASIC's CALL

COLOR statement. For example, CALL COLOR (3,16,2) gives white numbers from 0 to 7 on a black background. Changing this to CALL COLOR (3,2,16) will cause the inverse of it to appear (black numbers on a white background).

RANDOM NUMBERS

APPLESOFT's ability to retain a random number for re-use can be confusing. This means that you cannot just convert the APPLE RND statement to TI. What happens is this: If the statement is RND(0), APPLESOFT re-uses its last random number. If the statement is RND(N) where N is positive, it gives a new random number. If the statement is RND(N) where N is a negative number, N acts as a 'seed' number and all other RND statements will follow a standard sequence. Note that the value N can be any positive number in order to give a new random number.

If you see a statement using RND(0), backtrack to the last statement with RND(N) and save that random number in place of RND(0). For example:

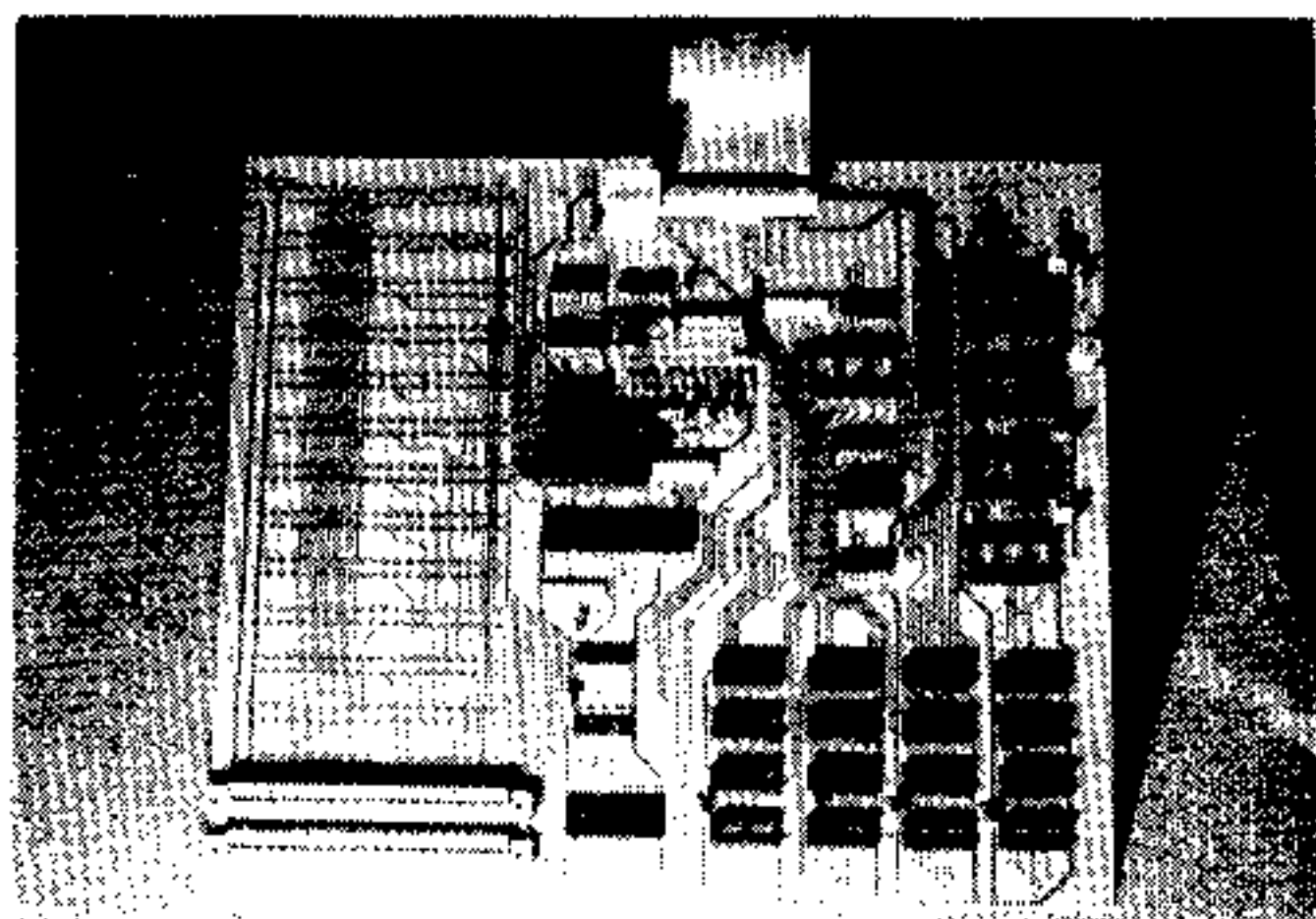
```

10 IF RND(2)>.5 THEN 500
:
:
:
60 IF RND(0)>.75 THEN 600
would be converted to be:

10 Q=RND
15 IF Q>.5 THEN 500
:
:
60 IF Q>.75 THEN 600
    
```

Continued on p. 45

FOR TI UNIVERSITY BOARD OWNERS:



MEMORY AND I/O EXPANSION MODULE (MEM)

- 8K MEMORY EXPANSION--RAM/EPROM MIX
- I/O EXPANSION TO 480 SINGLE-BIT PORTS
- BUILT-IN EPROM PROGRAMMER

EXPAND YOUR UNIVERSITY BOARD!

INCREASE THE VERSATILITY OF YOUR UNIVERSITY BOARD (TM 990/189) BY KEEPING YOUR UTILITY PROGRAMS "ON LINE". THE MEMORY AND I/O EXPANSION MODULE MAKES THE UNUSED PORTION OF THE UNIVERSITY BOARD'S MEMORY SPACE AVAILABLE WITH CONVENIENT PARTITIONING. YOU CAN HAVE UP TO 8K OF ADDITIONAL MEMORY ON LINE, WITH ANY MIX OF READ/WRITE MEMORY AND EPROM; JUMPER SELECTABLE ON 1K BOUNDARIES WITH POWER ON.

FOR I/O-INTENSIVE APPLICATIONS, THE MEM HAS A 15-SLOT I/O SECTION WITH EACH SLOT ADDRESSING 32 BITS ON THE UNIVERSITY BOARD'S UNIQUE CRU BUS. THIS GIVES SINGLE- OR MULTI-BIT ACCESS TO 480 INDIVIDUAL I/O POINTS.

THE MEM HAS A BUILT-IN PROGRAMMER FOR TMS 2708 AND 2716 EPROMS, WITH PROGRAMMING SOFTWARE CAPABLE OF BEING LOCATED AT FOUR ADDRESSES. THIS SOFTWARE ALLOWS YOU TO COPY NEW CODE FROM EXISTING ROMs OR DIRECTLY FROM THE READ/WRITE MEMORY USED TO DEVELOP THE CODE.

MEMORY AND I/O EXPANSION MODULE

(Assembled and tested with complete User's Manual)

\$199.95 plus \$5 postage and handling

(Texas residents please add 5% sales tax)

George Goode & Associates, Inc.



12606 GREENVILLE AVENUE
SUITE 100
DALLAS, TX 75243
PHONE (214) 690-0595

Applesoft to TI BASIC . . . from p. 44

MULTISTATEMENT LINES

A key point about APPLESOFT that I haven't yet mentioned is that it allows multiple statements on one line number. Each statement is separated by a colon. This allows code like:

```
10 X=X+1:Y=Y+1:Z=Z+1
```

The biggest problem with translating multistatement lines is that there may not be available line numbers to assign to the converted statement lines. For example:

```
400 A=A+1:FOR I=1 TO X:B=I*A:NEXT I
401 GOSUB 403
402 RETURN
403 REM
404 GOSUB 600
405 A=A+10
406 RETURN
```

The problem here is that there is no room to separate the multiple statements on line 400.

One way around this is to use a line number translation: Multiplying all line numbers by 10 allows you space to insert the extra lines of code. The translated code is as follows:

```
4000 A=A+1
4002 FOR I=1 TO X
4004 B=I*A
4008 NEXT I
4010 GOSUB 4030
4020 RETURN
4030 REM
4040 GOSUB 6000
4050 A=A+10
4060 RETURN
```

IF-THEN-ELSE

APPLESOFT does not require the ELSE feature of an IF statement because it allows other statements to follow the THEN part of the IF statement. The following shows how this works:

```
10 IF X=A THEN X=X+1:Y=Y+1
20 A=X+Y
```

If X is equal to A, all statements following the THEN are executed. If X isn't equal to A, the program simply advances to statement 20. The TI BASIC equivalent is:

```
10 IF X=A THEN 15 ELSE 20
15 X=X+1
16 Y=Y+1
20 A=X+Y
```

One result of multiple statements per line is that TI programs require much more coding and a concurrent increase in memory needed for code. Keep this in mind if you are tempted to enter a program requiring 16K RAM in APPLESOFT; it probably won't fit in your TI machine. [Of course, if you have the new TI Extended BASIC, all this is moot, since this Command Module allows multiple statement lines. See May/June, p. 28 —Ed.]

LOGICAL EXPRESSIONS

Both interpreters allow logical expressions to be used as if they were numeric values. APPLESOFT treats true expressions as if they are equal to 1, while false expressions are equal to 0. For TI BASIC true expressions are -1, false are 0. Whenever converting code from APPLESOFT, just insert a "--" in front of the logical expression:

```
10 X=(0$="A")*5
```

becomes

```
10 X=-(0$="A")*5
```

Continued on p. 52



Memory & I/O Expansion plus an EPROM Programmer for the TM 990/189 University Board

By John Caulfield

2211 West 119th Terrace
Leawood, Kansas 66209

If you've been bitten by the 16 bit bug, then you've probably taken a serious look at Texas Instruments' TMS 9900 family. About two years ago, TI introduced the TM990/189 University Board. It was originally designed as a tool to bootstrap oneself into the 16-bit world. The University Board itself packs a number of exciting features—a TMS 9980A 16-bit CPU, integral 45-key alphanumeric keyboard and display, resident monitor and assembler, audio cassette interface, 16-bit programmable I/O port, 2K RAM and 6K EPROM/ROM plus over 500 pages of tutorial text.

A new member has now been added to the University Board family. George Goode and Associates, Inc. offers an extremely versatile memory expansion, I/O interface, and EPROM programmer to work in concert with the TM990/189.

The memory portion of the module expands the system memory to the full address capability of the TMS9980A CPU—i.e., 16K bytes. The University Board itself has a capacity for 8K bytes (6K ROM/EPROM and 2K static RAM); the expansion module provides the other 8K capability. And it does so with utmost flexibility. Read on, all you memory addicts!

Sockets are provided for 8K bytes of EPROM using type 2716 EPROM or 4K bytes using type 2708 EPROM. There are also sockets for 8K bytes of static RAM. The unit allows access to any possible 8K combination of EPROM or RAM configurable on 1K boundaries (2K boundaries if you use type 2716 EPROMs). Now that's how I spell memory relief!

All that is necessary to alter the RAM/EPROM configuration is to change the wire jumpers on the memory configuration socket. It is not necessary that EPROM and RAM occupy contiguous areas of memory; you need only observe the boundary restrictions. Oh yes, this is a BYOMC (bring your own memory chips) product, but that should not present any problems to the enterprising user. You'll need to acquire either

four type 2708 or 2716 EPROMs and a maximum of sixteen type 2114 RAMs.

The expansion module provides a motherboard for an I/O part called the Communications Register Unit (CRU). The CRU is unique to the TI 990 computer family (of which the University Board is a member). It is used only for data transfers to or from external devices and is the serial port through which the microprocessor communicates with the outside world.

The CRU provides a serial transfer of one or more bits in or out of the CPU via two dedicated pins on the 9980A—CRUIN and CRUOUT. A clock, CRUCLK, is used as a time strobe to coordinate data transfers. Use of the CRU does not subtract from any available memory locations, and it is separate from the data bus.

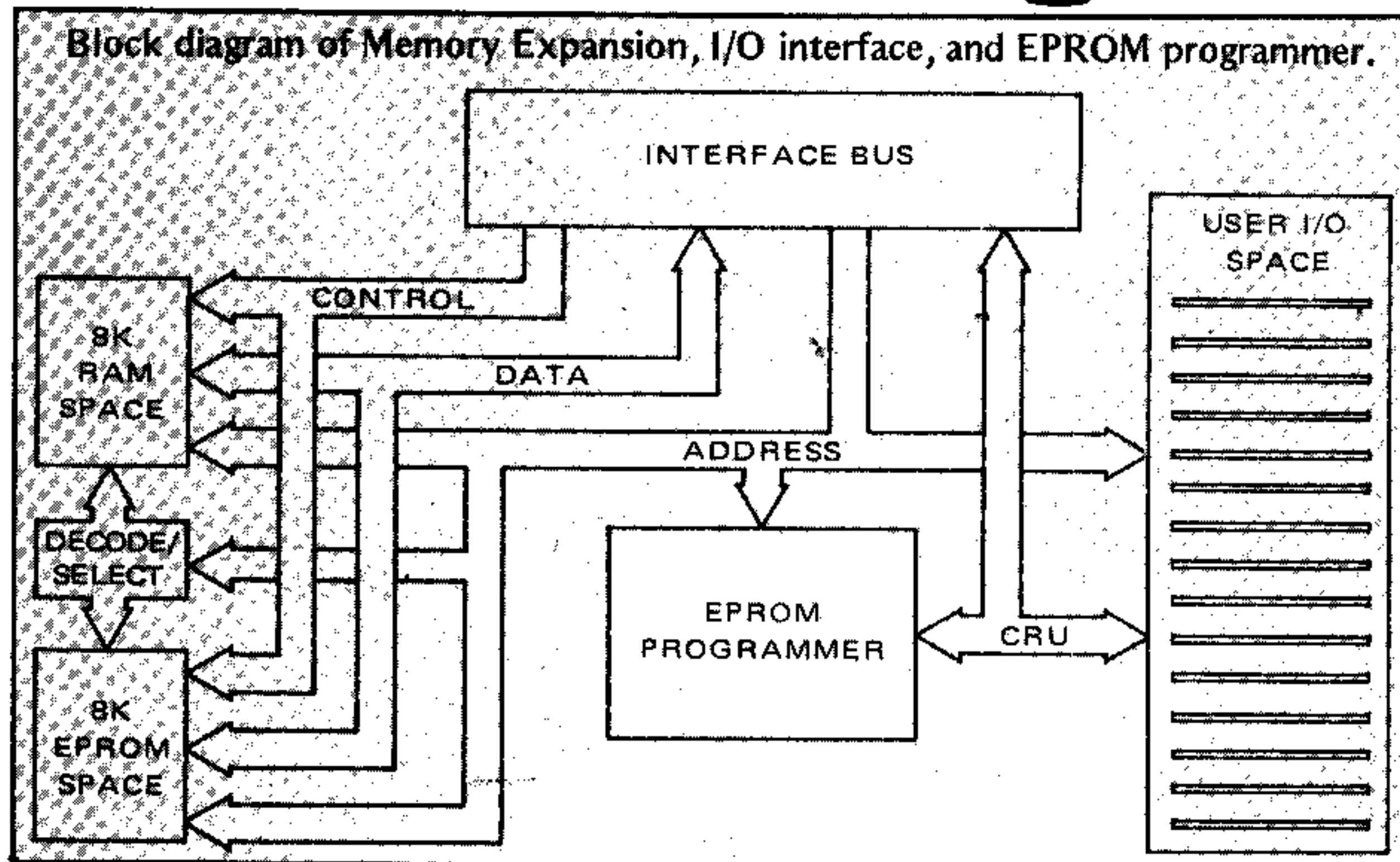
The major advantage of the CRU is "bit diddling." A single bit (or multiple bits up to 16) may be changed in the CRU output scheme. A single bit is all that is necessary to monitor or change the status of a motor, relay, switch, etc. The University Board allows easy access to 512 CRU bits. On the University Board itself, interfacing to 16 bits of CRU is provided. The motherboard on the expansion module provides capability to interface to the remaining CRU bits. The motherboard portion of the

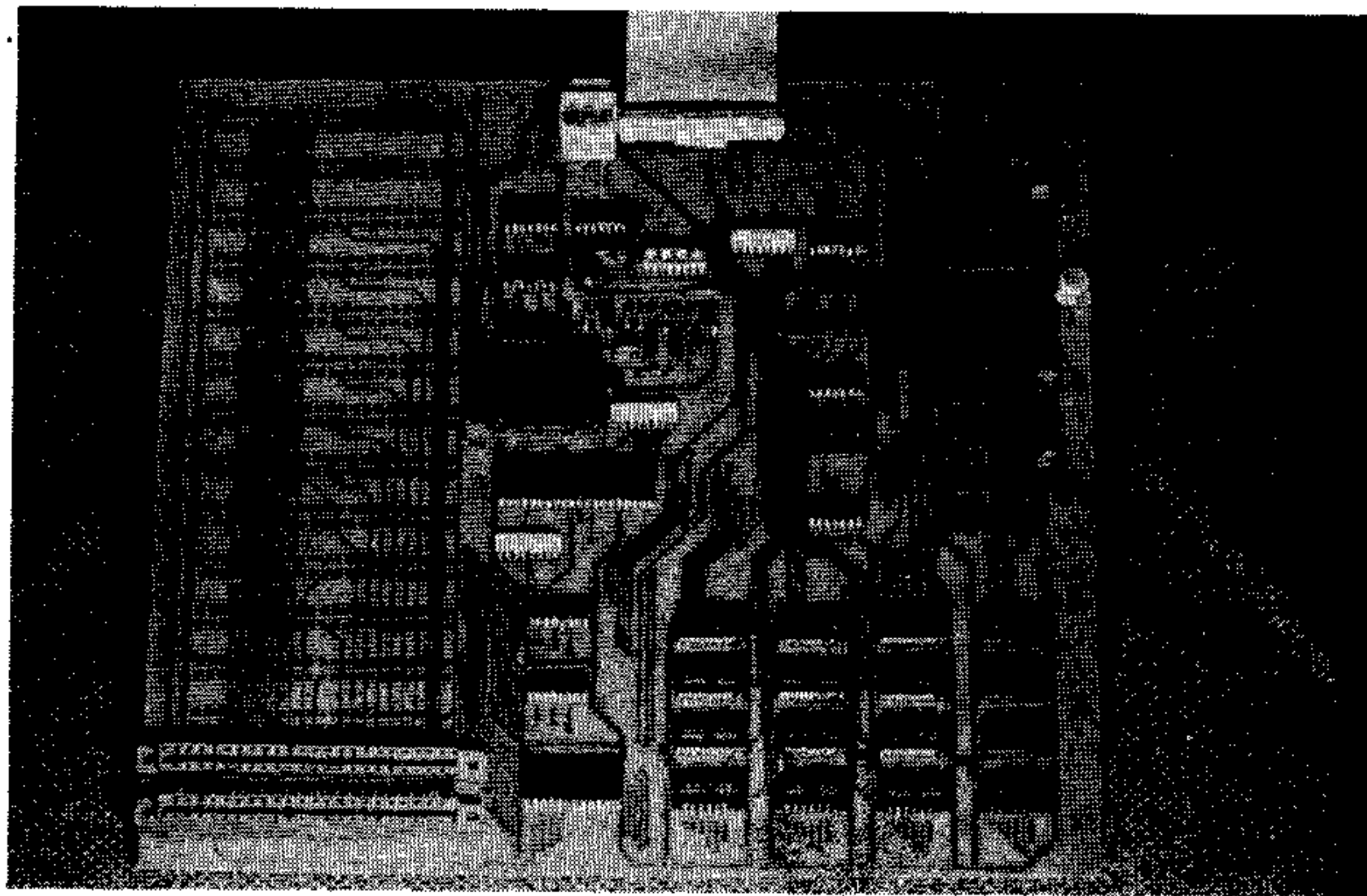
expansion module has positions for 15 individual interface boards. Each of these I/O boards can be implemented as 32 I/O bits.

Discrete hardware devices can be used on the individual boards for decoding the address bits. Complete implementations of this scheme would allow the user to directly control and/or monitor *nearly 500 separate events!* The user, however, must provide the interface boards to mount in the 22-position edgeboard connectors (contacts are on .156 inch centers). To help get started, the expansion module comes with two edgeboard connectors (plus interface and power supply cables). Users can attach additional connectors and I/O boards as required.

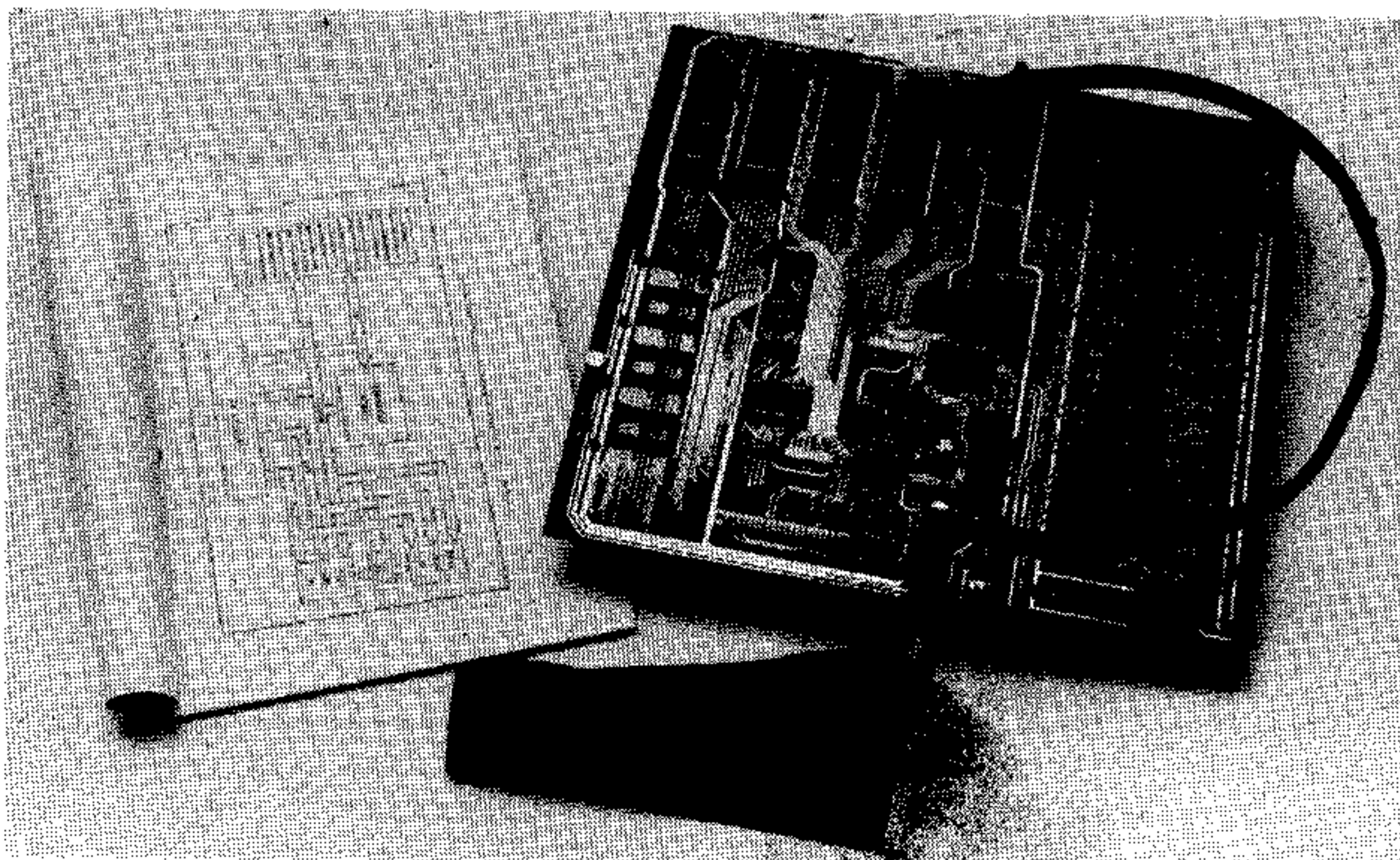
An EPROM programmer is also included on the board for type 2708 and 2716 EPROMs. The software driver has programming circuitry to automatically adjust the procedure for either type of EPROM.

The capabilities of this expansion module—with its flexible memory, I/O and EPROM programmer—are indeed impressive. I've found it to beautifully compliment the TM990/189 University Board by providing a complete laboratory environment to totally dissect the 16-bit Texas Instruments TMS9900 architecture.

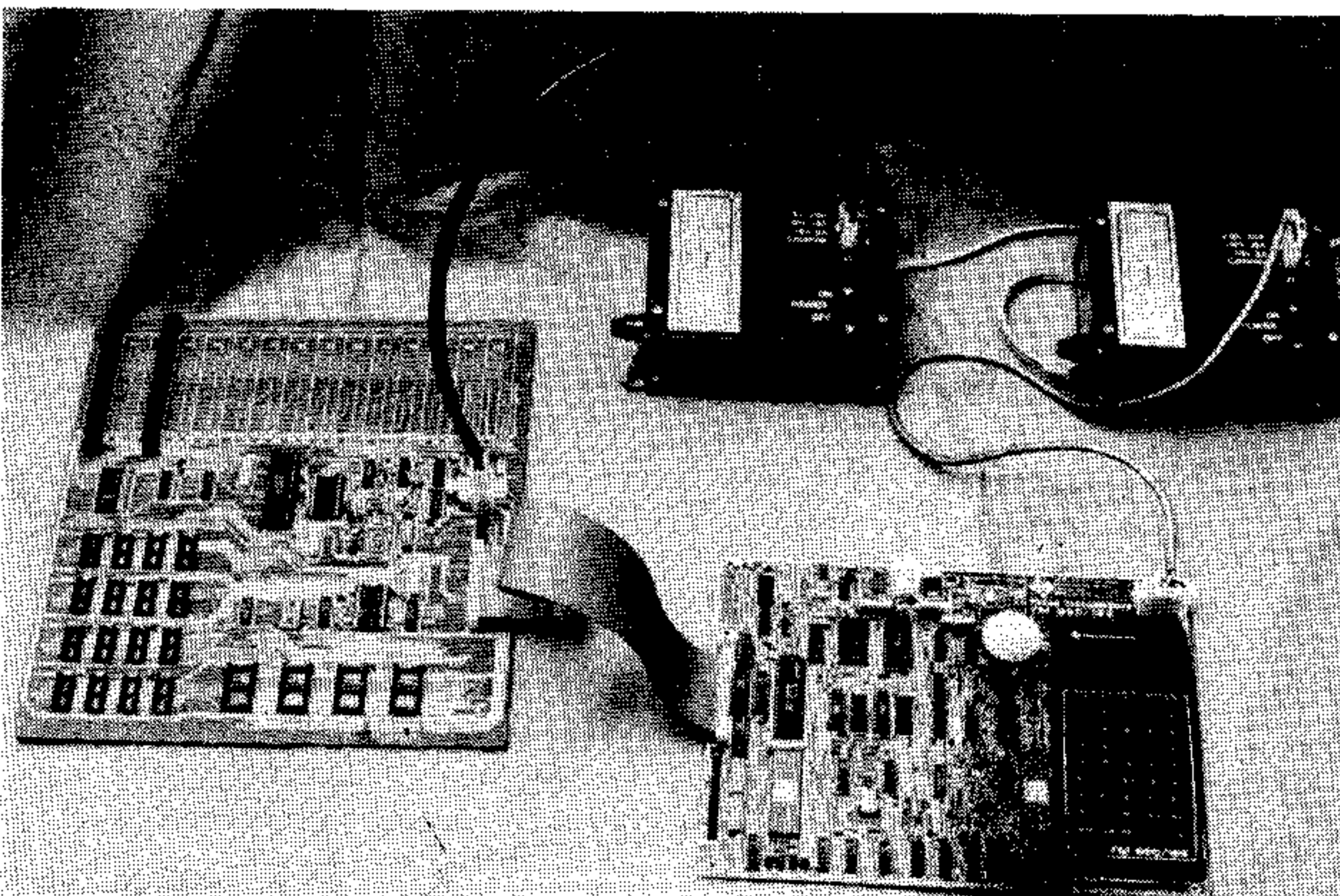




Memory expansion, I/O interface and EPROM programmer module.



Memory expansion, I/O interface, and EPROM programmer plus interconnecting cables and instruction book.



Expansion board, TM990/189 University Board, interconnecting cables and power supplies.

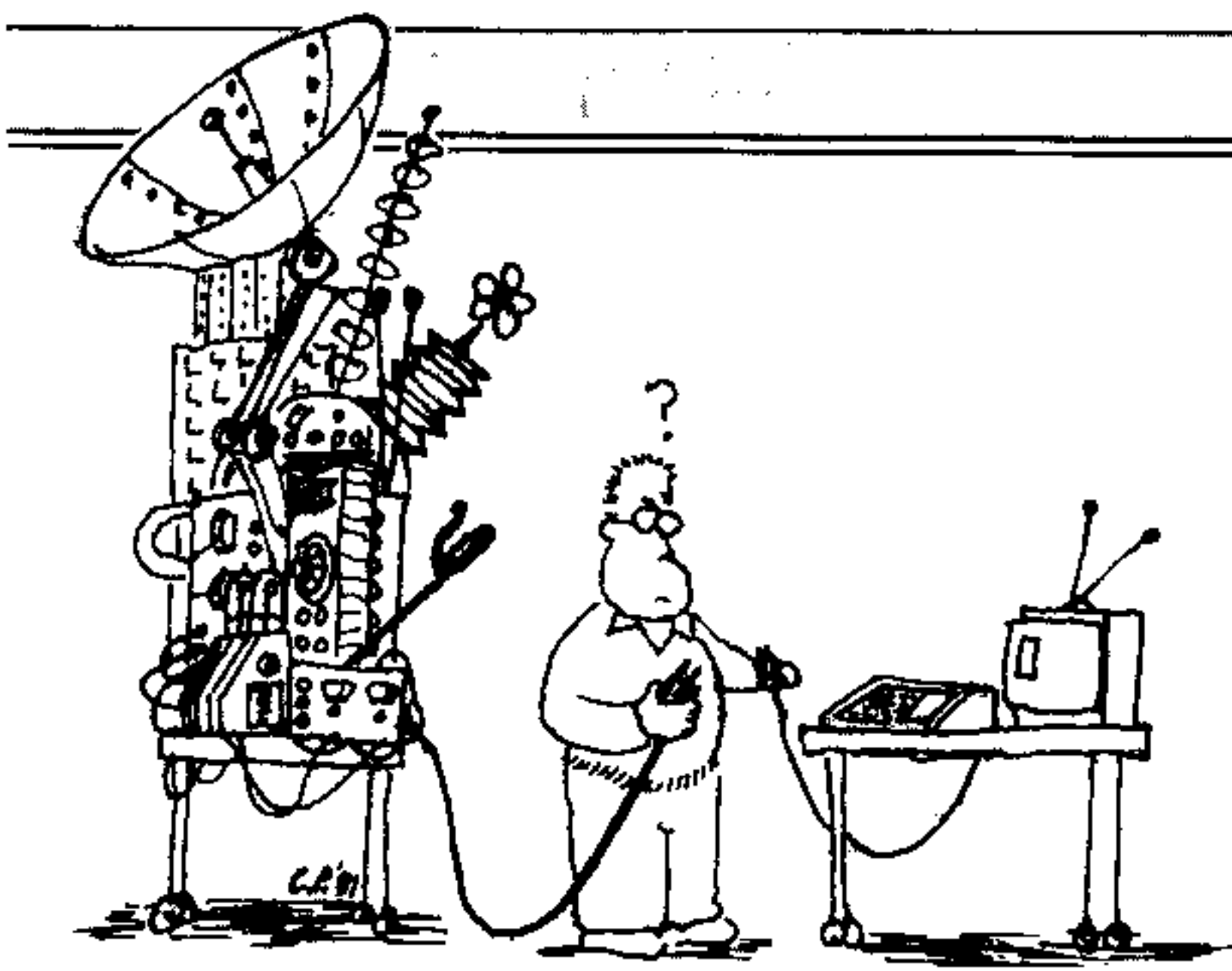
Space War . . . from p. 39

```

3550 CALL HCHAR(I,F,32)
3560 IF A<>I THEN 3590
3570 IF B<>F THEN 3590
3580 GOTO 5510
3590 IF C<>I THEN 3620
3600 IF D<>F THEN 3620
3610 GOTO 5510
3620 NEXT I
3630 GOTO 2200
3640 IF Z=97 THEN 3660
3650 IF Z<>105 THEN 3790
3660 FOR I=E+1 TO 24
3670 CALL GCHAR(I,F,P)
3680 GOSUB 6050
3690 CALL HCHAR(I,F,112)
3700 CALL HCHAR(I,F,32)
3710 IF A<>I THEN 3740
3720 IF B<>F THEN 3740
3730 GOTO 5510
3740 IF C<>I THEN 3770
3750 IF D<>F THEN 3770
3760 GOTO 5510
3770 NEXT I
3780 GOTO 2200
3790 IF Z=98 THEN 3810
3800 IF Z<>106 THEN 3940
3810 FOR I=F+1 TO 32
3820 CALL GCHAR(E,I,P)
3830 GOSUB 6050
3840 CALL HCHAR(E,I,112)
3850 CALL HCHAR(E,I,32)
3860 IF A<>E THEN 3890
3870 IF B<>I THEN 3890
3880 GOTO 5510
3890 IF C<>E THEN 3920
3900 IF D<>I THEN 3920
3910 GOTO 5510
3920 NEXT I
3930 GOTO 2200
3940 IF Z=99 THEN 3960
3950 IF Z<>107 THEN 4090
3960 FOR I=F-1 TO 1 STEP -1
3970 CALL GCHAR(E,I,P)
3980 GOSUB 6050
3990 CALL HCHAR(E,I,112)
4000 CALL HCHAR(E,I,32)
4010 IF A<>E THEN 4040
4020 IF B<>I THEN 4040
4030 GOTO 5510
4040 IF C<>E THEN 4070
4050 IF D<>I THEN 4070
4060 GOTO 5510
4070 NEXT I
4080 GOTO 2200
4090 IF Z=100 THEN 4110
4100 IF Z<>108 THEN 4260
4110 FOR I=E-1 TO 1 STEP -1
4120 F=F+1
4130 IF F=33 THEN 4250
4140 CALL GCHAR(I,F,P)
4150 GOSUB 6050
4160 CALL HCHAR(I,F,112)
4170 CALL HCHAR(I,F,32)
4180 IF A<>I THEN 4210
4190 IF B<>F THEN 4210
4200 GOTO 5510
4210 IF C<>I THEN 4240
4220 IF D<>F THEN 4240
4230 GOTO 5510
4240 NEXT I
4250 GOTO 2200
4260 IF Z=101 THEN 4280
4270 IF Z<>109 THEN 4430
4280 FOR I=E-1 TO 1 STEP -1
4290 F=F-1
4300 IF F=0 THEN 4420
4310 CALL GCHAR(I,F,P)
4320 GOSUB 6050
4330 CALL HCHAR(I,F,112)
4340 CALL HCHAR(I,F,32)
4350 IF A<>I THEN 4380
4360 IF B<>F THEN 4380
4370 GOTO 5510
4380 IF C<>I THEN 4410
4390 IF D<>F THEN 4410
4400 GOTO 5510
4410 NEXT I
4420 GOTO 2200

```

Continued on p. 51



Frugal Floppies: Adding Your Own Disk Drives

By Richard M. Bies

217 Park Entrance Drive
Pittsburgh, PA 15228

For those of you who long for the speed and power of a disk drive but shrink from the price tag, there is possible relief—a surplus drive at a fraction of list price. But before getting into the technical details of how to hook up one to your TI-99/4, let's first take a look at how a disk system operates. In effect, a disk system constitutes a *second* computer—one controlled by the TI-99/4. It contains RAM, ROM, processing and control units, plus the drive itself—a high precision electro-mechanical device that spins, reads data from, and writes data to a magnetic disk.

The disk itself is a circular piece of magnetic medium, much like the recording tape in a cassette. It has a large, central hole into which the hub of the drive

fits and by which the disk is spun (at about 300 rpm). There is also a small hole a short distance further out, by which the disk is indexed. A disk is enclosed in a square envelope (the "jacket"), with cut-outs for the central hub, the index mark, the read-write head of the drive, and a write-protect notch in the edge. These disks are referred to as "floppies" or "floppy disks" because of their flexibility—in contrast to *hard* or *rigid* disks used on larger computer systems. Floppies and their corresponding drives are presently available in two sizes: 8 inches and 5¼ inches. The drives that are compatible with the TI-99/4 disk controller are of the smaller variety.

A floppy disk is said to have *tracks* and *sectors*. Actually, the magnetic material is homogeneous—the tracks and sectors are created by the drive. Thus, when a disk is "initialized" (made ready for use) in the TI-99/4 system, these

tracks and sectors are marked magnetically on the disk (by a software-controlled stepping motor that moves the read-write head in or out), together with the name of the disk and other information. The *tracks* are concentric paths beginning with the first one near the outer periphery of the magnetic disk, and the last one outside the index hole toward the center of the disk. The TI Disk Controller is programmed to deal with either 40 or 35 such concentric tracks. The mechanical and electronic characteristics of the drive determine and specify the number of tracks available. (Older mini-floppy drives have 35, 40, or 44 tracks. Newer design "double-density" drives tend to have 77 tracks.)

Each track is further divided into a number of pie-shaped radial sections called *sectors*; the TI system is comprised of 9 sectors per track. These sectors are physically located on the disk with reference to the index hole: A small light shines on one side of the disk's surface. When the index hole passes between this light and a photocell located on the other side of the disk's surface, the light strikes the photocell, and the resulting synchronization signal indexes the sectors for each revolution of the disk. Each packet of information thus has a track and sector address. And for each file name or program name, these addresses are automatically recorded in the first track of the disk. (Because of this index information and the location markers in each sector, the actual information which can be stored is approximately 88K bytes of the 109K bytes available on the disk.) When the disk is accessed, the head is sent to the proper location to read the file requested, or to an empty sector to record the new data. Alternately, if the file requested is not listed as being present on the disk, or if the disk is full or "write-protected" (not allowing any more data to be placed on the disk), the controller software returns an appropriate error message for display on the screen.

So in effect, it is the disk controller that stands between the disk drive and the computer to perform the parallel-

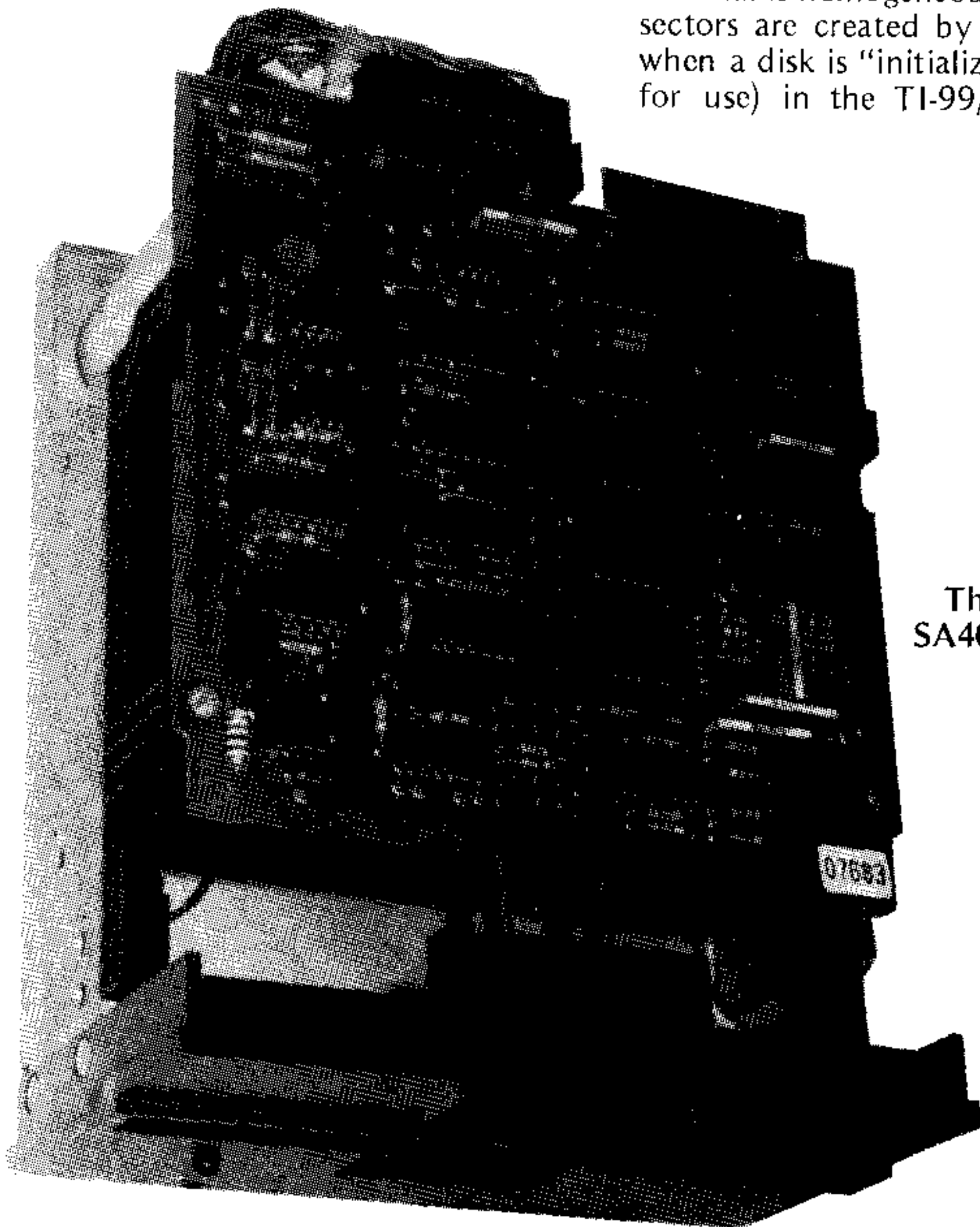


Photo 1.
This is the Shugart SA400 surplus drive, basically as we received it.

serial data transformations, time the inputs and outputs, interpret and relay the signals, and in the case of the TI-99/4, take control of part of the computer to accomplish the interface. (The most noticeable manifestation of this is the *loss* of approximately 2K bytes of RAM—memory taken over by the controller for input and output. This is the reason *longer* programs can be loaded and RUN from *tape* when disk controller is turned off.)

The disk system gains its superiority over the cassette system because this combination of sophisticated electronics and precision electro-mechanics allows data to be found quickly at any part of the disk. This is possible because the head can be moved rapidly from the outer track to the inner track and intercept the rotating sectors (going by the head 5 times per second). Furthermore, the effective data transmission rate—100 times faster than cassette—is augmented by standard disk system features that include cataloging, searching, and protecting the data on the disk. All in all, a remarkable system . . . nearly unbelievable in its reliability.

Hooking Up the Components

Now that we've discussed how the disk drive system operates, let's see how we can hook up surplus or budget drives to our TI Disk Controller. From an article in the March-April 1981 issue of *Science and Electronics* (now named *Computers and Programming*) I learned of the abundance of single-density, mini-floppy (5¼ inch) drives on the surplus market (selling for \$100 to \$160 each). These drives are currently being replaced by higher capacity ones (double-sided, double-density), larger ones (8 inch), or hard disks (of the multi-megabyte Winchester type).

The drive I purchased was a Shugart SA-400 single-sided, single-density 35-track unit that was formerly installed in Wang equipment. (The Shugart SA-400 is the industry standard single-sided, single-density mini-floppy drive. Other brands, Siemens, MPI, Teac, etc., are made to the same dimensions and electronic specifications.) Since this unit was removed from other equipment, it came without case, power supply, data cable, or terminating resistor. (See Photo 1.) This is typical of the cheapest drives you'll be most likely to find. Occasionally new units are available, and occasionally a case for the drive itself is included. For that matter, units designed for the TRS-80 computer—which are completely compatible by the way—may be available for a reasonable price, with power supply and case included. If the unit has neither case nor power supply, these components are available as an assembled package. The cost should be around \$50, although I was unable to find one in stock at that price at the time I needed it, and settled for the Vista

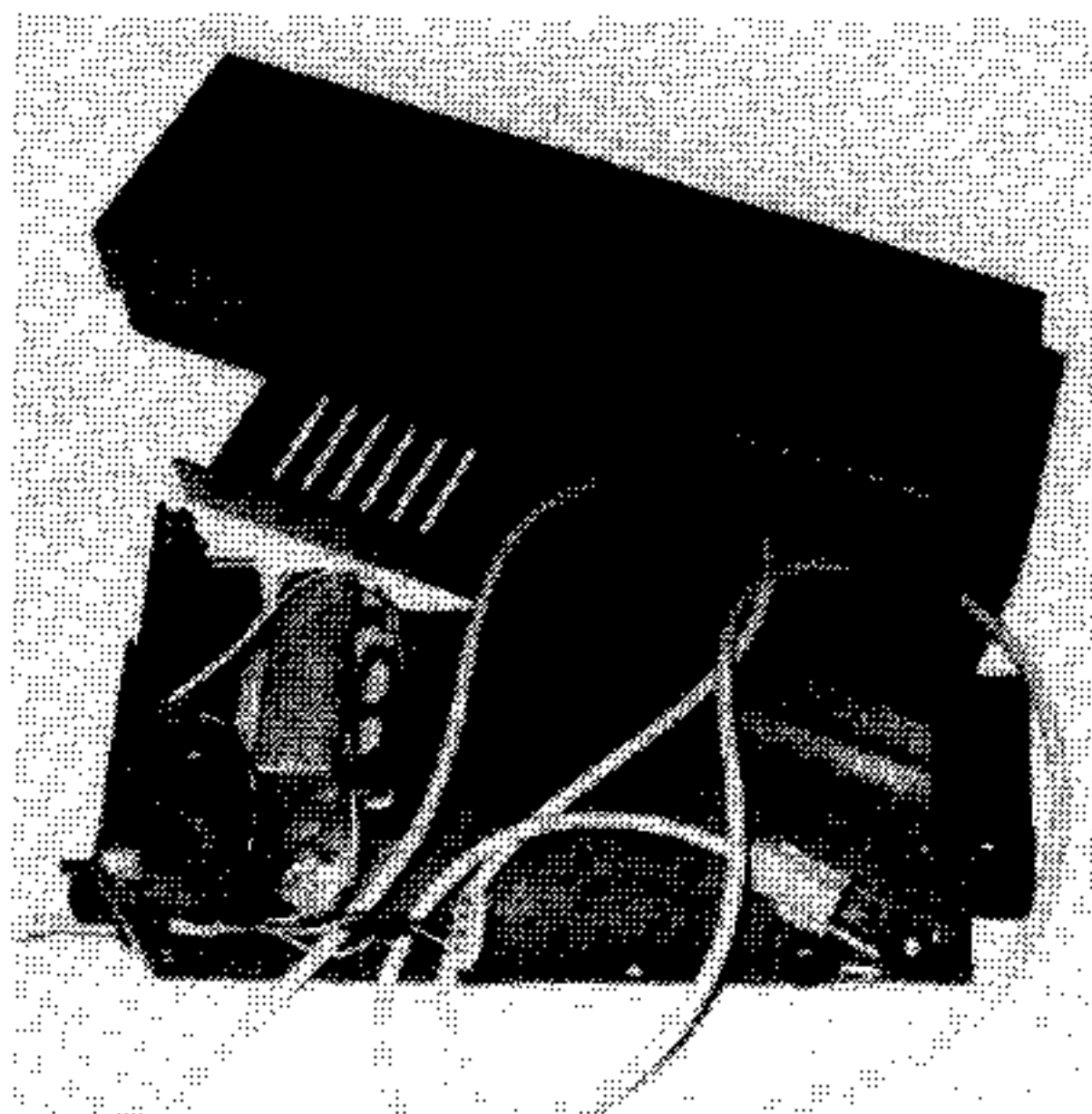


Photo 2. This is the Vista VIS 9801 case with power supply.

VIS 9801 at \$75. (See Photo 2.) If the drive comes with only a case, separate power supplies are available, starting around \$30.

This cabinet package, however, usually does not include the data cable. Again, the parts are standard, but not necessarily easy to find. Since the data cable connector on the TI Disk Controller and the connector on the circuit board of the drive are both male 34-pin edge-card type connectors, a 34-conductor edge-card to edge-card cable is required. The length can be 6 inches to a few feet; if less than 18 inches or so, the drive must be placed atop the controller, probably the wrong place for extended use. I had a 3-foot one made up at cost of \$20. (See Photo 3.) A patient shopper should be able to get one for \$10 to \$15. (Try Radio Shack part #RS 260-1406 for \$19.95 as a possibility, or purchase the connectors for \$4 to \$5 and cable for about 60 cents per foot, and fabricate one yourself.)

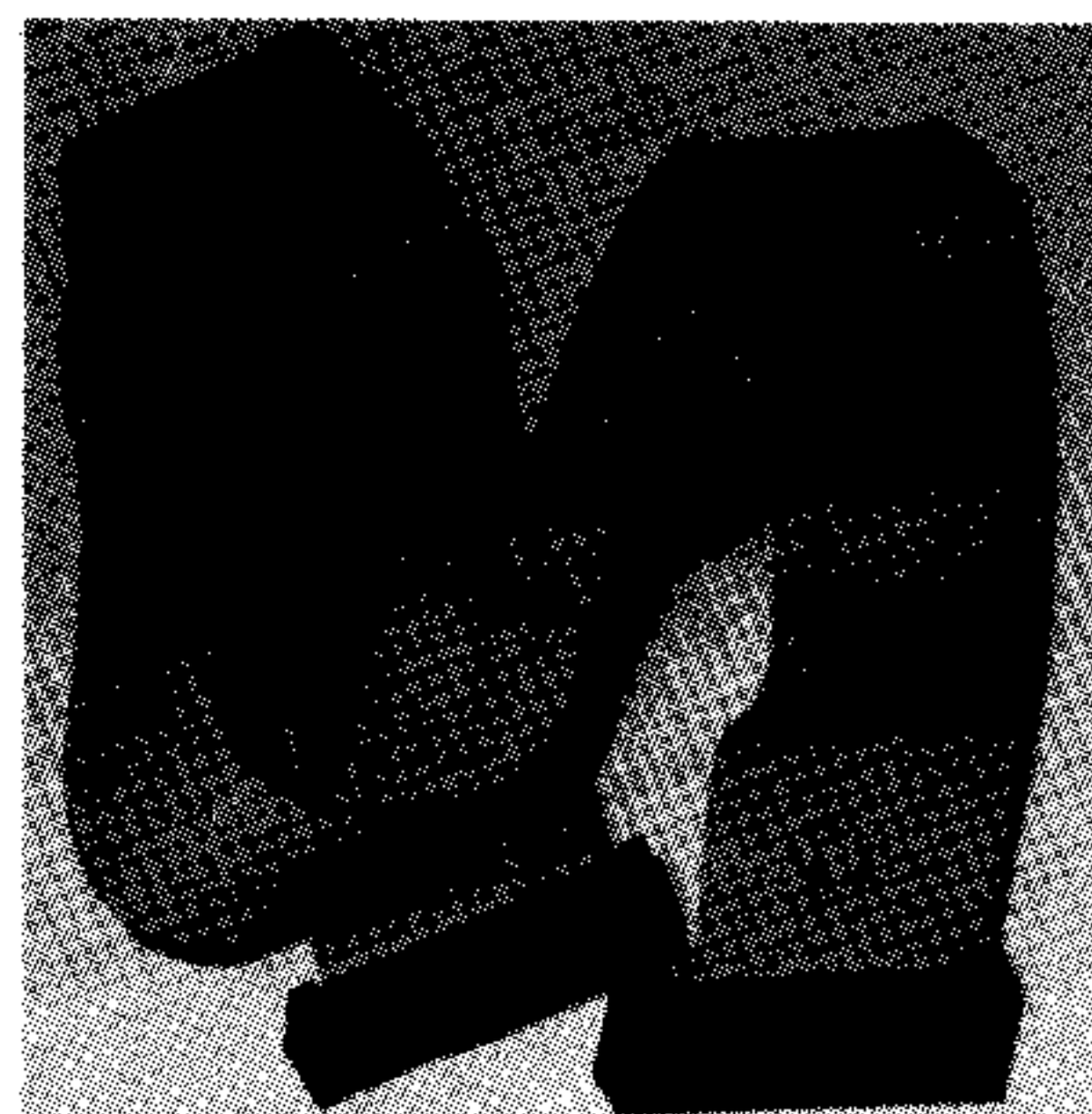
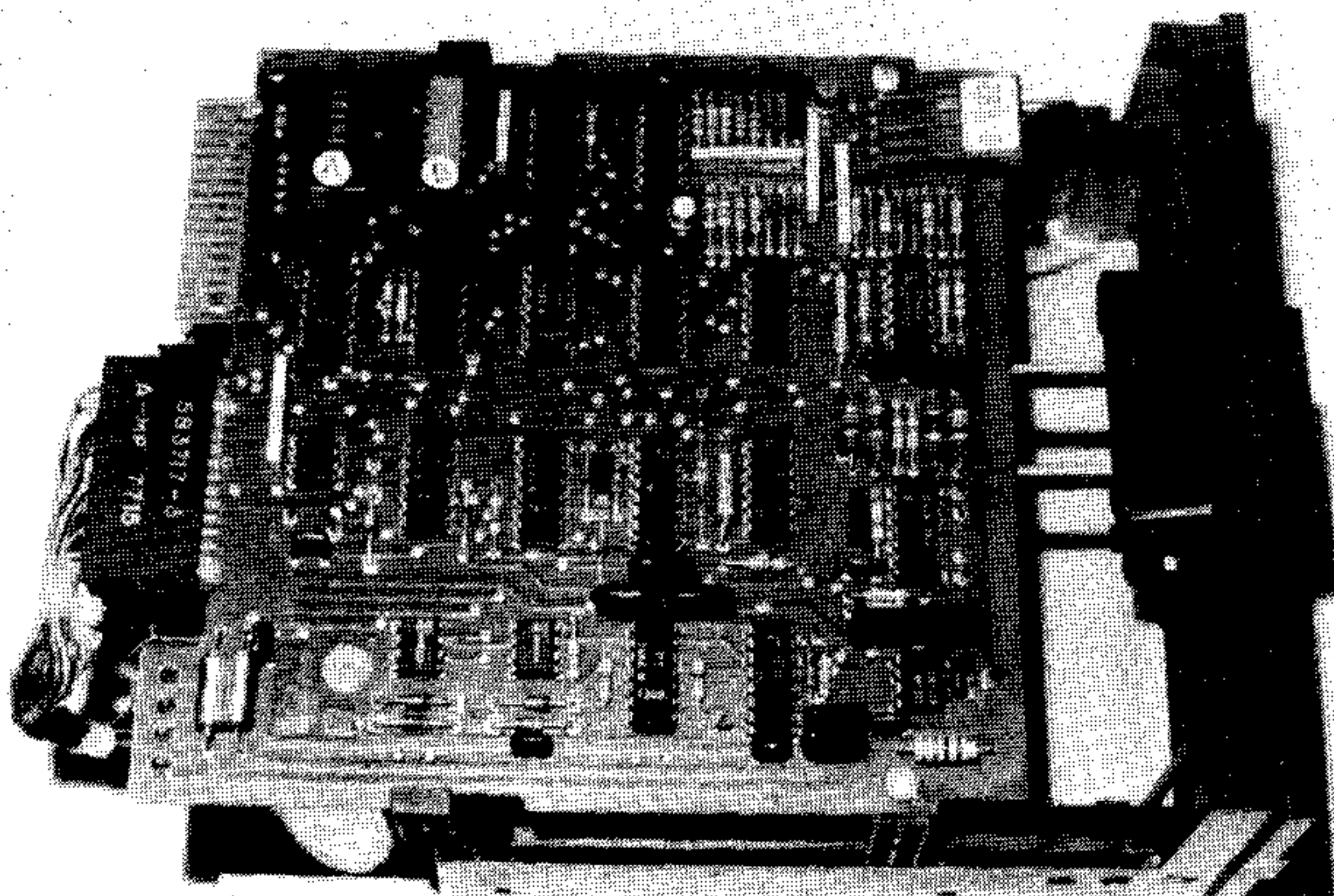


Photo 3. This is the three-foot-long 34-conductor edge-card to edge-card data cable.

Finally, the circuit board on the drive must be correctly configured. This involves the shunt and terminating resistor. (See Photo 4.) The shunt is simply a set of connections across an IC socket (located next to the board's edge connector) which provides a path for the drive number signals coming from the controller (tagged "A" on Photo 4). Any drive will probably have one in place, although it may not be intact or correctly positioned for the TI-99/4. The shunt is usually a 6-conductor, 12-pin device in a 14-pin socket. The two end positions are labeled "HS" and "HM." The drive I received had the shunt positioned to the "HM" end, with "HS" empty. The *Science and Electronics* article suggested that most personal computers required the "HS" connection. Accordingly, I removed and reinserted the shunt to the "HS" end.

The DS1 (Drive-Select), DS2 and DS3 connections (marked on the circuit board adjacent to the socket) are

Photo 4. This is the circuit board on the drive, with the shunt (A) and the terminating network (B) at upper left.



for turning on the correct drive in multiple drive applications. As I was planning one drive only, I left all the shunt connections intact. (See Photo 5.) Of course, in a multiple drive application, it would be necessary to experiment with these connections, bending two pins out at a time so they do not enter the socket, to determine which DS line the controller is using for each drive. The controllers know the drives as DSK1., DSK2., and DSK3., but do not necessarily use DS1 for DSK1. and DS2 for DSK2., etc. The *Science and Electronics* article advised that in TRS-80 applications all DS lines can be left wired, as the selection is taken care of by discontinuities in their data cables. I understand that the TI-99/4 uses a similar approach. Presumably, if you can get TI data cables, you can leave all shunts connected even in multiple drive connections. The bottom line is that between shunt connections and data cables, you and your controller must come to a common understanding of which drive is which.

The *Science and Electronics* article also suggests that the "MX" connection is not used in multiple drive applications. Speculating on the Radio Shack practice, I would suspect that one "MX" connection should still be intact, and the entire matter may be taken care of in the data cables. At any rate, I defer to someone else to make the definitive statement as to multiple-drive shunt configuration, as I have not tried it.

The other plug-in is the terminating network. This is the next IC socket in from the card edge after the shunt (tagged "B" in Photo 4). Again, if you get one in your drive—fine; if not, each system must have *one* such network. In multiple drive applications, the last drive on the cable should have it. (Radio Shack does this wrong, with the network

installed on the first drive.) The network is simply a number of resistors—each equal in resistance to the characteristic impedance of the input lines—encapsulated in an IC type package. (In the Shugart 400, the characteristic impedance is 150 ohms. Other manufacturers' drives could differ, but probably are the same.) I was able to order a terminating network for \$1.00 from the same source as the data cable. It is also possible to sim-

ply bend and clip the leads on 1/8 watt resistors of the appropriate resistance and insert them in the socket. The documentation for the drive will tell you the characteristic impedance and show the pin numbers to be bridged. The IC socket is 14 pins, but on the Shugart only five pairs have to be bridged.

Testing It Out

With the parts all collected and assembled, I borrowed a TI Disk Controller and drive. First, we operated the standard system, and initialized a fresh disk. Then, unplugging the standard drive, we plugged in the surplus drive and found that it would read the prepared disk correctly. This test seemed to establish compatibility and correct operation of the drive.

However, when we attempted to *write* to the disk, the first and only hitch appeared. Every disk, including any fresh new disk, was always regarded as "write-protected" and wouldn't accept any data. With the TI Disk Controller, this signal (which prevents the write-enable signal from energizing the recording head) can arise either *mechanically*, as in putting tape over the write-protect notch in the disk jacket cover, or *electronically*, by virtue of encoding the disk itself by the routine in the Disk Manager Command Module.)

We checked the drive and found that the micro-switch sensing the open notch seemed to operate correctly, but that the test point which should have dropped

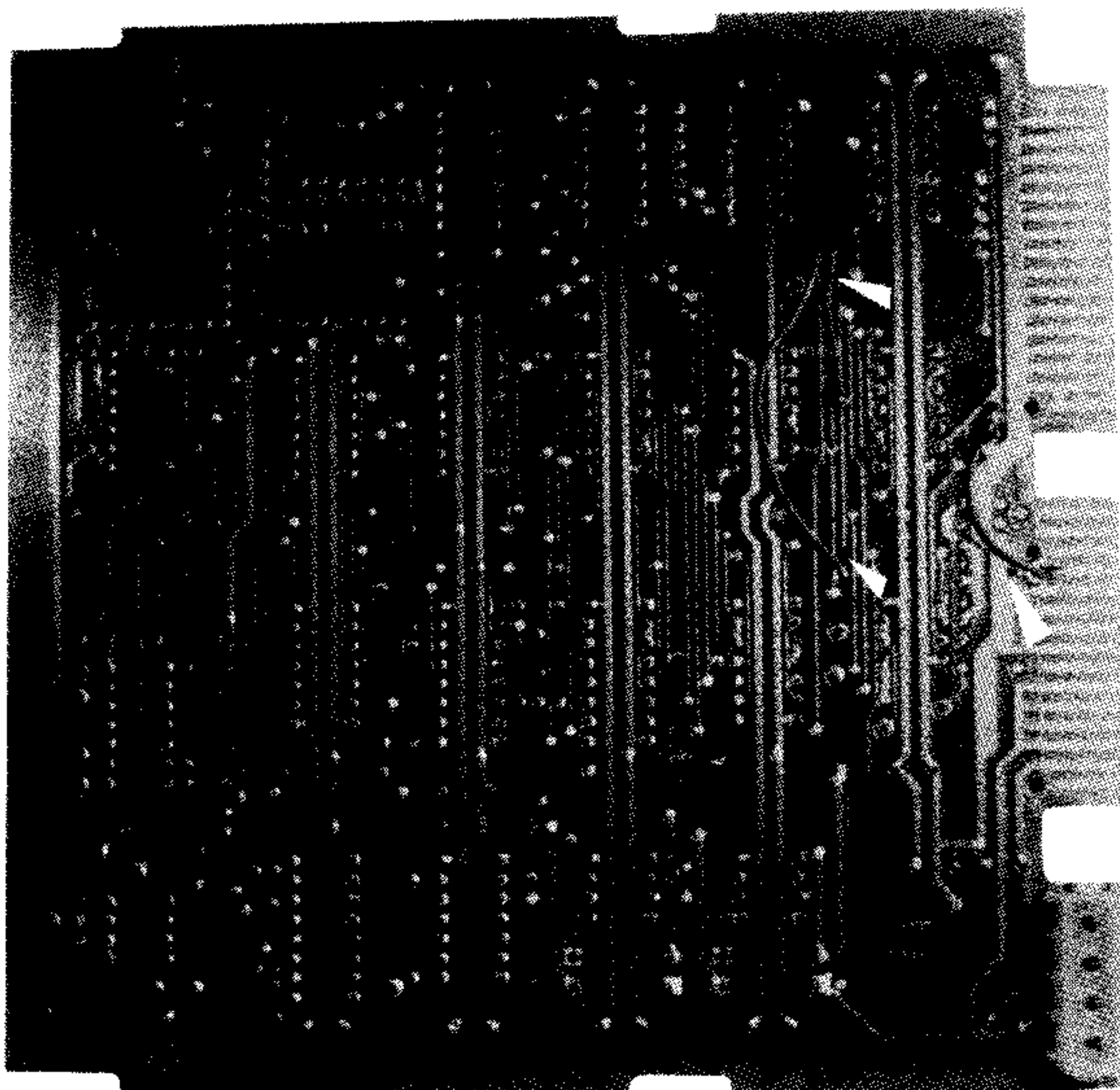
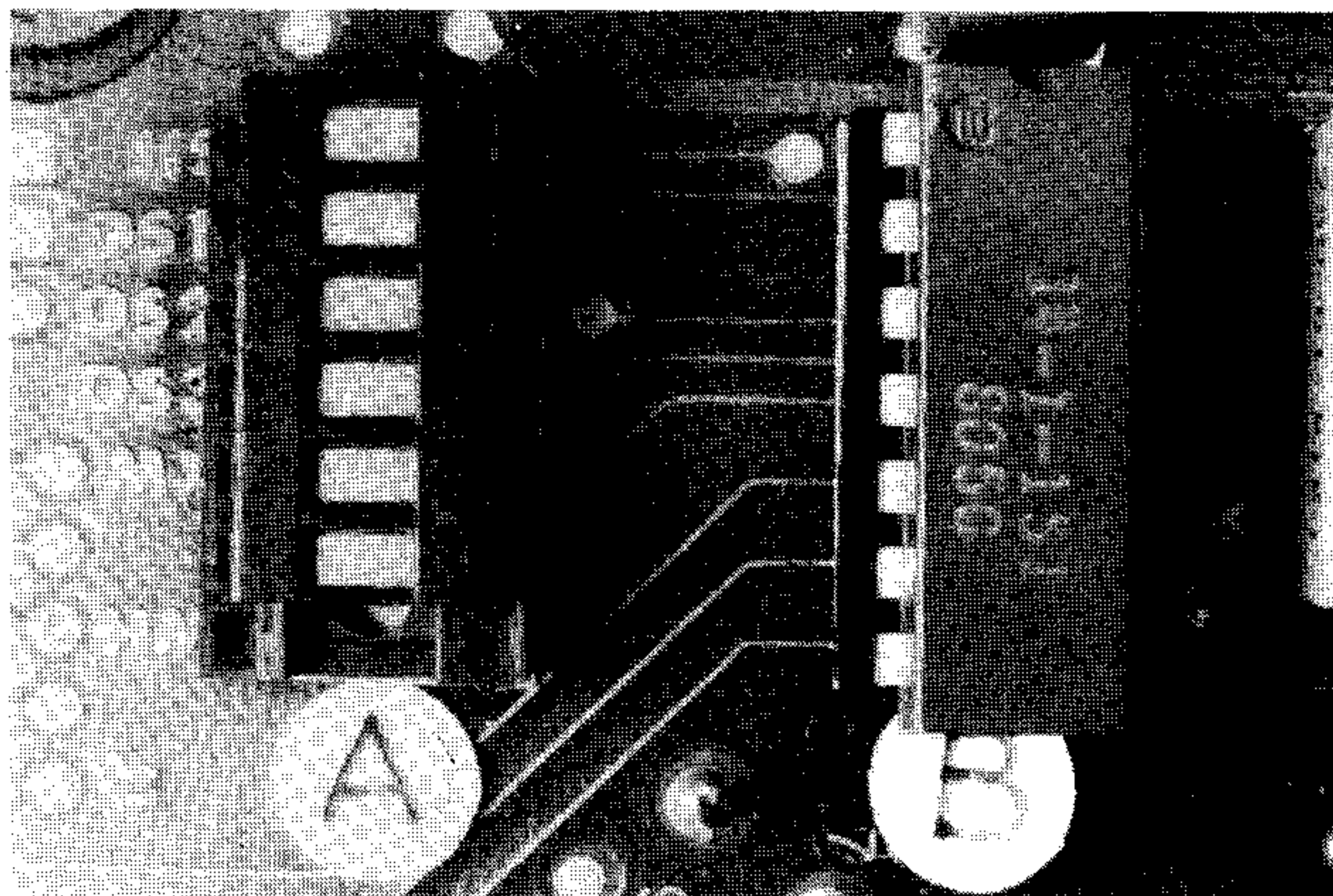


Photo 6. On the back of the circuit board, three white pointers on the right side show the points where the circuit tracing was cut.

Photo 5. This close-up shows the shunt (A) and the terminating network (B). The DS and other markings are to the left of the shunt socket.



to ground (0 volts) when the notch was open actually stayed high. We then removed the printed circuit board from the drive, and discovered that a resistor and a couple of wires had been added to the underside of the board. Study of the circuit board also revealed a couple of cuts in the circuit tracing (white pointers on Photo 6), and when compared to the schematics (you should be sure to get a copy of the documentation, including schematics), disclosed that the location of the cut leads and added circuits was indeed the write-protect circuit. Hence, it was highly probable that in its former use, the drive unit was configured for a read-only application. Yet it took a bit of courage (and philosophical reflection on the probable cost of a new board and the vagaries of Murphy's Law) to apply the soldering iron to the board, remove the addition, and restore the original circuit changes (as in Photo 6). [Note that electronic technicians frequently make minor circuit changes to perfect some application, to compensate for some design fault, or to adjust for a replacement part.]

But Murphy was nowhere to be found . . . and the drive actually worked! It seems clear to me, now that I've been through it, that adding a "foreign" drive (a single-density TRS-80 Model 1 or other standard drive) to the TI-99/4 Disk Controller shouldn't be feared. The more adventuresome among you can save even more by careful shopping and collecting the needed parts. Keep in mind, however, that most *surplus* drives and some others as well (such as the Apple's) will not have a power supply. Also, some of the component pieces can be a bit hard to locate, and can, in fact, cost more than would seem reasonable.

I have seen bare drives, without even a circuit board, offered for \$80 (as in

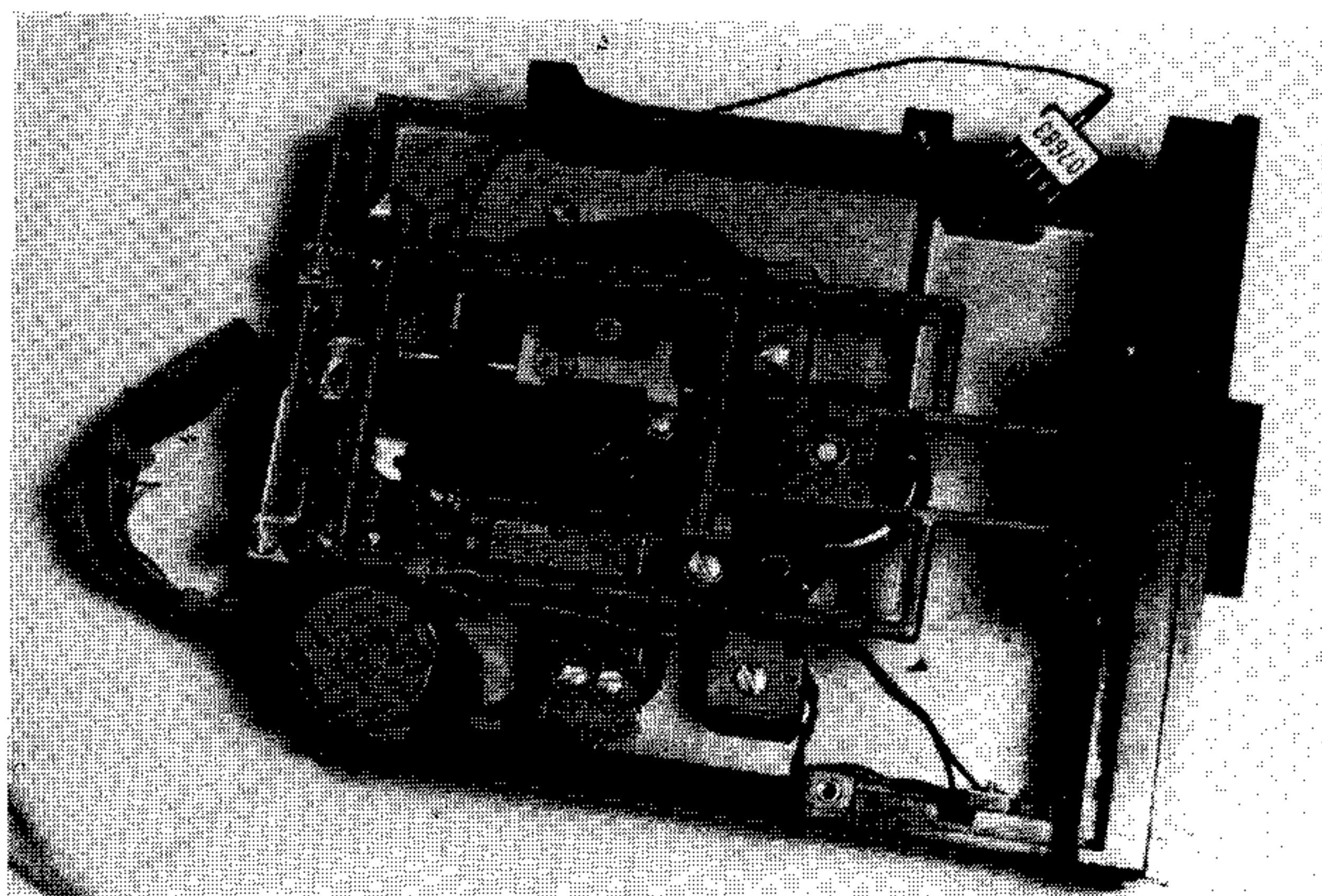
Photo 7). It may be worthwhile to explore the cost of a new circuit board for one of these. (I expect the circuit boards are not interchangeable among brands; thus it would be necessary to determine the brand of the drive, and shop for a circuit board for it.) This would insure that any problem should be limited to the mechanics of the drive, and these seem reasonably reliable. The board is attached to the drive electrically with a couple of plugs, and mechanically with a couple of screws.

It remains for each reader to decide how adventuresome he wishes to be. I will admit to having a BSEE ('58) in my background, and even having studied the basics of digital computing back then. As I followed other callings, however, this training has lain dormant in the intervening years, during which a few things have happened in the field (such as disk memory). Presently, my most sophisticated test equipment is a \$12 multi-meter. Clearly, problems with pulse timing would be over my head. Yet with much of the sophisticated electronics around today, actual failures are very often simple matters of *continuity*—of the type the added circuit and cut traces (described in this article) presented. I would urge anyone who can follow a schematic diagram, who would like to save some money, and who enjoys the satisfaction of knowing what's inside those "black boxes" to give it a try.

A Note About Drives

99'er Magazine's technical editors have found that in addition to the Shugart drive, both Siemens and MPI drives also work. The Teac drive, however, doesn't appear to be compatible because of its slower track-to-track access time.

Photo 7. This shows the bare drive without the circuit board.



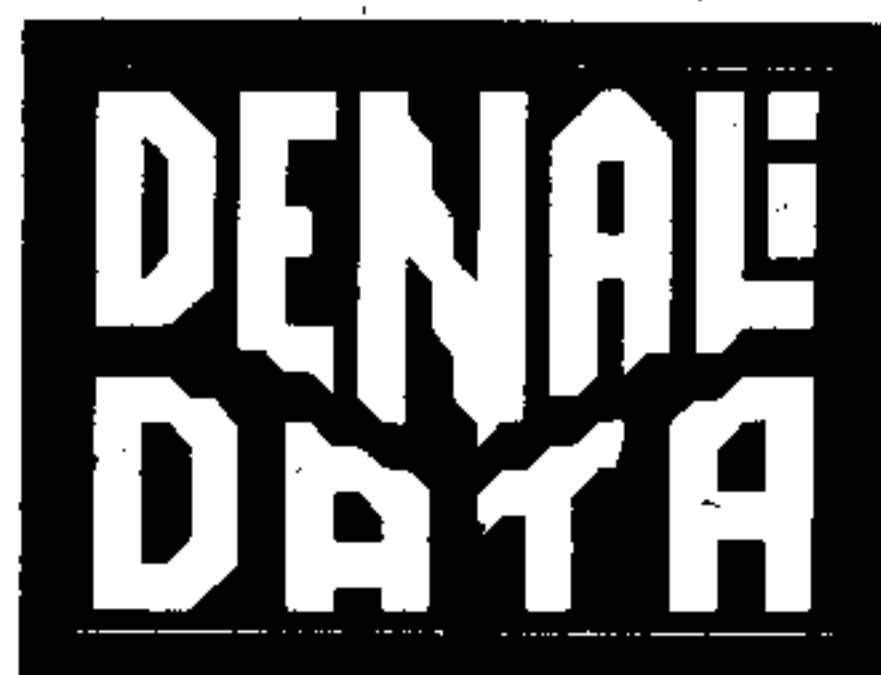
Space War . . . from p. 47

```

4430 IF Z=102 THEN 4450
4440 IF Z<>110 THEN 4600
4450 FOR I=E+1 TO 24
4460 F=F+1
4470 IF F=33 THEN 4590
4480 CALL GCHAR(I,F,P)
4490 GOSUB 6050
4500 CALL HCHAR(I,F,112)
4510 CALL HCHAR(I,F,32)
4520 IF A<>I THEN 4550
4530 IF B<>F THEN 4550
4540 GOTO 5510
4550 IF C<>I THEN 4580
4560 IF D<>F THEN 4580
4570 GOTO 5510
4580 NEXT I
4590 GOTO 2200
4600 FOR I=E+1 TO 24
4610 F=F-1
4620 IF F=0 THEN 4740
4630 CALL GCHAR(I,F,P)
4640 GOSUB 6050
4650 CALL HCHAR(I,F,112)
4660 CALL HCHAR(I,F,32)
4670 IF A<>I THEN 4700
4680 IF B<>F THEN 4700
4690 GOTO 5510
4700 IF C<>I THEN 4730
4710 IF D<>F THEN 4730
4720 GOTO 5510
4730 NEXT I
4740 GOTO 2200
4750 C=C+1
4760 IF C<>25 THEN 4780
4770 C=1
4780 CALL GCHAR(C,D,P)
4790 GOSUB 6140
4800 IF V=1 THEN 5510
4810 CALL HCHAR(C,D,105)
4820 GOTO 2200
4830 C=C-1
4840 IF C<>0 THEN 4860
4850 C=24
4860 CALL GCHAR(C,D,P)
4870 GOSUB 6140
4880 IF V=1 THEN 5510
4890 CALL HCHAR(C,D,104)
4900 GOTO 2200
4910 D=D+1
4920 IF D<>33 THEN 4940
4930 D=1
4940 CALL GCHAR(C,D,P)
4950 GOSUB 6140
4960 IF V=1 THEN 5510
4970 CALL HCHAR(C,D,106)
4980 GOTO 2200
4990 D=D-1
5000 IF D<>0 THEN 5020
5010 D=32
5020 CALL GCHAR(C,D,P)
5030 GOSUB 6140
5040 IF V=1 THEN 5510
5050 CALL HCHAR(C,D,107)
5060 GOTO 2200
5070 C=C-1
5080 D=D+1
5090 IF C<>0 THEN 5110
5100 C=24
5110 IF D<>33 THEN 5130
5120 D=1
5130 CALL GCHAR(C,D,P)
5140 GOSUB 6140
5150 IF V=1 THEN 5510
5160 CALL HCHAR(C,D,108)
5170 GOTO 2200
5180 C=C-1
5190 D=D-1
5200 IF C<>0 THEN 5220
5210 C=24
5220 IF D<>0 THEN 5240
5230 D=32
5240 CALL GCHAR(C,D,P)
5250 GOSUB 6140
5260 IF V=1 THEN 5510
5270 CALL HCHAR(C,D,109)
5280 GOTO 2200
5290 C=C+1
5300 D=D+1

```

Continued on p. 53



DENALI DATA DESIGN

ARTI-STIC™ JOYSTICK ADAPTER

Allows the use of faster,
smoother-acting, low cost,
Atari™ Joystick on TI 99/4 & 4A
JUST PLUG IN AND PLAY

\$20 postpaid

Adapter + 2 Atari Joysticks

\$35

Please add \$2.00 for shipping

OTHER PRODUCTS NOW OR SOON AVAILABLE FROM DENALI DATA DESIGN PROGRAMMING AIDS

- * SCREEN GRAPHICS LAYOUT SHEETS (INCLUDING SPRITE DEFINITION GRIDS)
- * MONITOR SCREEN OVERPLAY (GRID ON CLEAR FILM TO FIND WHERE THOSE CHARACTERS ARE REALLY AT.)
- * CODING SHEETS (BASIC & X-BASIC)

HARDWARE

- * ARTI-STIC™ (JOYSTICK ADAPTER) ALLOWS THE USE OF FASTER, SMOOTHER-ACTING, LOW COST, ATARI™ JOYSTICKS ON TI 99/4 & 4A JUST PLUG IN AND PLAY
- * JOYPAD™ PLUGS INTO JOYSTICK CONNECTOR ON 99/4 & 99/4A AND ALLOWS DIRECTION & NUMERIC ENTRY IN ONE HAND HELD UNIT. (REQUIRES SUPPORT SOFTWARE — SUPPLIED IN BASIC AND X-BASIC ON TAPE OR DISK.)
- * 9 PIN & 25 PIN EXTENSION CABLES (SAME PLUGS AS ON JOYSTICK, CASSETTE & RS-232 PORTS.)
- * STACKER BUS™ (ALLOWS VERTICAL STACKING OF UP TO 4 PERIPHERAL DEVICES.)
- * BACKER BUS™ (ALLOWS PLACEMENT OF PERIPHERALS BEHIND CONSOLE.)
- * SPLIT-T BOX™ (DOUBLE T-SWITCH FOR RS232. ALLOWS FOUR DEVICES TO BE PLUGGED INTO RS232 WITH HARDWARE SWITCHES TO ELIMINATE THE NEED FOR CABLE-SWITCHING.) (SOFTWARE SWITCHABLE AVAILABLE LATER THIS YEAR.)

SOFTWARE

- * CHECK BOOK RECONCILIATION (REQUIRES JOYPAD)
- * SCREEN TO EPSON MX80™ BLOCK GRAPHICS DUMP (REQUIRES JOYPAD)

Dealer
Inquiries
Invited

SEND OR CALL FOR
FURTHER INFORMATION:

Toll Free 1-800-654-8499
In OK dial 1-405-524-7764

1413 N. MCKINLEY
OKC, OK. 73106

Applesoft to TI BASIC . . . from p. 45

AND/OR

APPLESOFT allows multiple IF tests to be combined using the Boolean operators AND and OR. TI BASIC also allows this using the "*" and "+" arithmetic expressions respectively. For example:

```

10 IF (A=B) AND (C=D) THEN X=X+1
is replaced with
10 IF (A=B)*(C=D) THEN 15 ELSE .....
15 X=X+1

```

In some cases, a straight conversion of the APPLESOFT IF... THEN will result in wasteful code. It is always a good idea to understand the purpose of the tests being made, and if possible, re-code them more efficiently. For example:

```

10 IF (A=B) AND (C=D) THEN X=X+1
20 Y=Y+1
would convert to:
10 IF (A=B)*(C=D) THEN 15 ELSE 20
15 X=X+1
20 Y=Y+1

```

but it would take less code (and therefore less core!) to invert the test:

```

10 IF (A<>B)+(C<>D) THEN 20
15 X=X+1
20 Y=Y+1

```

SPECIAL FUNCTIONS

Each interpreter has special functions oriented toward the manufacturer's hardware. Some of these are similar to other functions available in a different computer. I will only list the ones most commonly seen in APPLESOFT programs.

- CLEAR** Initializes all variables. Automatically done by TI BASIC as part of 'RUN'ning.
- HIMEM** Sets highest and lowest memory available to BASIC. No equivalent in TI BASIC.
- LOMEM** Gets amount of available memory left.
- FRE(0)** Gets joystick input. Use CALL JOYST instead.
- PDL(N)** The PDL function returns with values from 0 to 255. Figuring out exactly what's happening using PDL in an APPLESOFT program is difficult because if the value N is 0 to 3, you are referencing the joysticks. (Values from 4 to 255 can do wierd things.) Luckily, the APPLE joysticks don't seem to be used much. Also, the only way to test for the 'FIRE' buttons is to PEEK(-16287) through PEEK(-16284) for paddles 0 thru 3.

POP

Cancels the last GOSUB. This is mostly used in edit subroutines where an error causes the program to go to an error routine instead of RETURNing. The only way to code an equivalent in TI BASIC is to have the edit routine coded in an error switch which is interrogated as soon as the subroutine RETURNS.

ON ERR RESUME

This tells APPLESOFT to GOTO a part of the program if it encounters certain errors while processing. In TI BASIC, any errors are either handled by the BASIC interpreter (e.g., dividing by zero), or cause the program to end (e.g., reading past the last DATA statement). The ON ERR is most often used to trap an error expected by or consciously caused by the programmer. For example, it is used in a program that is reading DATA statements containing the notes of a song. The ON ERR code is hit after the last note is READ and causes the program to execute a RESTORE, starting the song all over again. The RESUME acts like a RETURN and brings the program back to the point where the error occurred.

USR(X) Jump to a machine language subroutine.

As you can see from the foregoing, converting most code from APPLESOFT to TI BASIC is straightforward with most of the effort devoted to converting PRINT statements. Most importantly, don't get frustrated if your first attempts don't succeed the way you intended. After a while, it will all become second nature.

**SUPPORT OUR ADVERTISERS.
THEY TOO MAKE THIS MAGAZINE
POSSIBLE . . .**

Custom Peripherals from **COMPUTECH**

-We Mak'em to Order-
Separate your Peripherals
with custom, shielded, I/O
cables from **COMPUTECH**.

100 Y\$ = Your Choice
110 Length = 1 Ft.*Y\$
120 Price = 24.00+(Length*4.00)

417-869-1684
Dealer Inquiries Invited

COMPUTECH

209 E. Walnut • Springfield, MO 65806

Numeric Data Entry is a Breeze With a Computech "SPEED KEY"

Designed for the TI 99/4 or /4A.

The "SPEED KEY" docks directly into the console (RS232 NOT Required). Required support software is resident in the "SPEED KEY" and no external power supply is required. Comes complete and ready to use. Price 169.95 + Shipping. (Allow 3-4 wk. Del.)

Space War ... from p. 51

```
5310 IF C<>25 THEN 5330
5320 C=1
5330 IF D<>33 THEN 5350
5340 D=1
5350 CALL GCHAR(C,D,P)
5360 GOSUB 6140
5370 IF V=1 THEN 5510
5380 CALL HCHAR(C,D,110)
5390 GOTO 2200
5400 C=C+1
5410 D=D-1
5420 IF C<>25 THEN 5440
5430 C=1
5440 IF D<>33 THEN 5460
5450 D=32
5460 CALL GCHAR(C,D,P)
5470 GOSUB 6140
5480 IF V=1 THEN 5510
5490 CALL HCHAR(C,D,111)
5500 GOTO 2200
5510 CALL SOUND(1000,-5,0)
5520 IF G=2 THEN 5560
5530 E=C
5540 F=D
5550 GOTO 5580
5560 E=A
5570 F=B
5580 CALL HCHAR(E,F,136)
5590 CALL COLOR(14,2,2)
5600 IF F+1<33 THEN 5620
5610 A1=1
5620 IF F-1>0 THEN 5640
5630 B1=1
5640 IF E-1<1 THEN 5700
5650 IF B1=1 THEN 5670
5660 CALL HCHAR(E-1,F-1,138)
5670 IF A1=1 THEN 5690
5680 CALL HCHAR(E-1,F+1,137)
5690 CALL HCHAR(E-1,F,139)
5700 IF E+1>24 THEN 5760
5710 IF A1=1 THEN 5730
5720 CALL HCHAR(E+1,F+1,138)
5730 IF B1=1 THEN 5750
5740 CALL HCHAR(E+1,F-1,137)
5750 CALL HCHAR(E+1,F,139)
5760 IF B1=1 THEN 5780
5770 CALL HCHAR(E,F-1,140)
5780 IF A1=1 THEN 5800
5790 CALL HCHAR(E,F+1,140)
5800 FOR I=1 TO 10
5810 CALL COLOR(14,2,2)
5820 CALL COLOR(14,7,2)
5830 NEXT I
5840 IF PT<>9 THEN 5880
5850 PRINT "TIE GAME!"
5860 PT=0
5870 GOTO 5980
5880 IF PTS>0 THEN 5900
```

```
5890 PTS=5
5900 IF G=2 THEN 5950
5910 PRINT "BLUE WINS!"
5920 BL=BL+PTS
5930 PTS=0
5940 GOTO 5980
5950 PRINT "YELLOW WINS!"
5960 YL=YL+PTS
5970 PTS=0
5980 PRINT "SCORE: BLUE "&STR$(BL)&"," YELLOW "&STR$(YL)&". "
5990 YPR=50
6000 BPR=50
6010 INPUT "PLAY AGAIN?":B$
6020 IF SEG$(B$,1,1)<>"N" THEN 1800
6030 CALL CLEAR
6040 STOP
6050 FOR X=144 TO 147
6060 IF P<>X THEN 6120
6070 IF Z>103 THEN 6100
6080 BL=BL-1
6090 GOTO 2200
6100 YL=YL-1
6110 GOTO 2200
6120 NEXT X
6130 RETURN
6140 IF P=32 THEN 6200
6150 IF P<144 THEN 6160 ELSE 6180
6160 PT=9
6170 GOTO 6190
6180 PTS=3
6190 V=1
6200 RETURN
6210 FOR X=144 TO 147
6220 IF P<>X THEN 6250
6230 V=1
6240 RETURN
6250 NEXT X
6260 RETURN
6270 PRINT "YELLOW RUNS OUT OF FUEL."
6280 PTS=2
6290 GOTO 5910
6300 PRINT "BLUE RUNS OUT OF FUEL."
6310 PTS=2
6320 GOTO 5950
6330 END
```

**→ NEED
DUSTCOVERS ?**

See Inside Back Cover
(page 95)

The Cube
An incredible 3-D
simulation of
the puzzle

Cassette for the
TI-99/4

\$14.95

Linear Aesthetic Systems
POB 23
West Cornwall
CT 06796

Sunshine Software

•• CRIBBAGE ••

Play against the TI-99/4. Computer plays the able opponent.

•• QUBIQ ••

A unique mind-boggling puzzle. Full Color & Graphics.

•• SPIRAL-GRAPHICS ••

Creates thousands of amazingly beautiful patterns in High Resolution. You specify the pattern, the computer does the rest.

•• OBSTACLE COURSE ••

Can you make it through this game of logic? Deduction tells you where the obstacles are.

\$8.95 each; 4 for \$24.95

Cassette only; TI BASIC

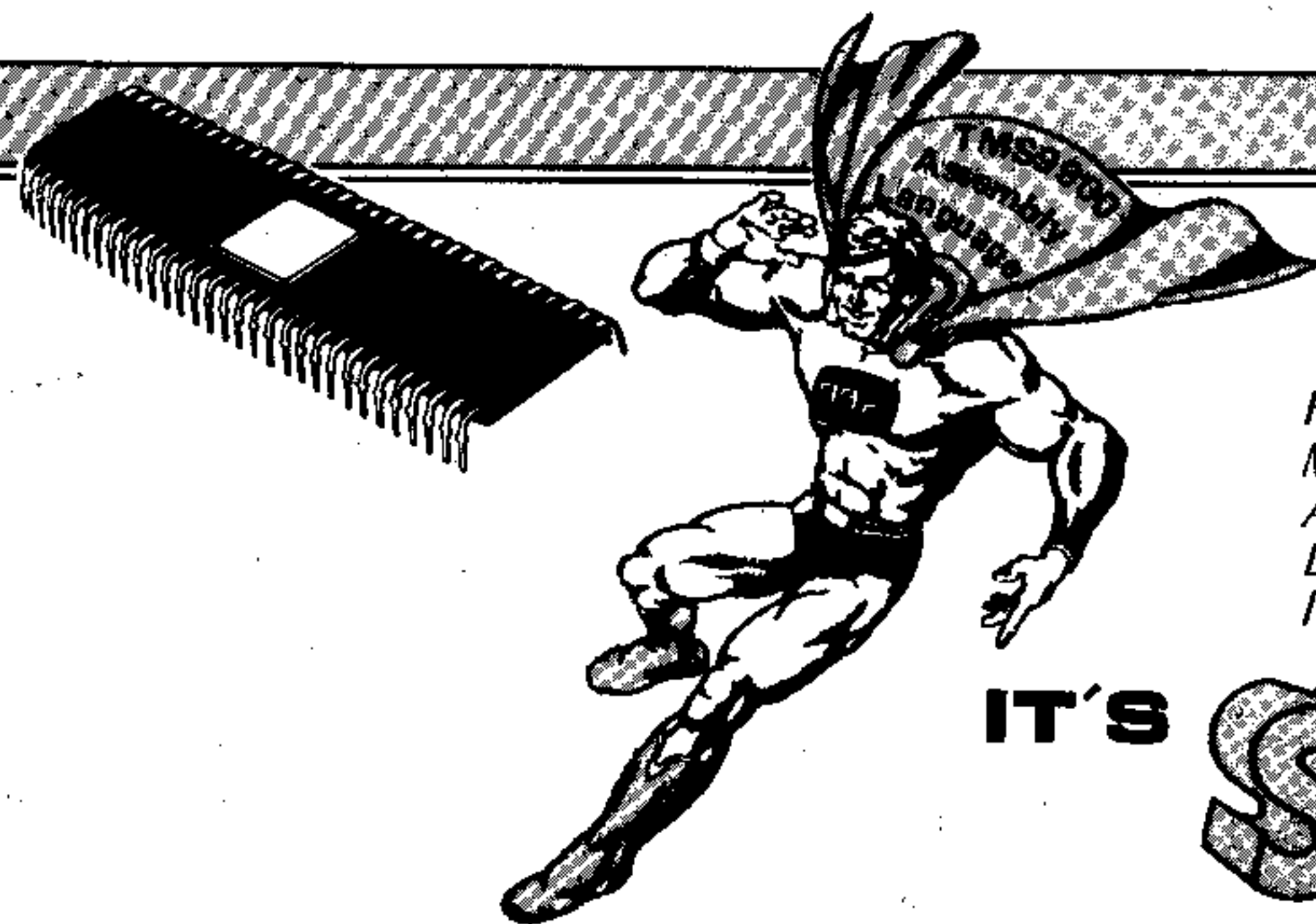
Complete documentation included

Sunshine Software

520 Milton Lane

Hoffman Estates, IL 60194

Tel (312)-885-2862



Faster than a speeding alien laser,
More powerful than an 8-bit data train,
Able to leap tall sorts in a single bound...
Look — Up in the high RAM —
It's an editor... It's an assembler...

IT'S

SUPER

LANGUAGE!

In this issue we interrupt our tutorial series on TMS9900 Machine and Assembly Language Programming so that we may introduce an exciting new product which will soon be available on the TI-99/4(A). The series will resume with Part 5 in the November/December issue.

By Patricia Swift

Assembly Language Editor (*The Human One*)

Before getting into the details of TI's new Editor/Assembler package, we should first consider what an assembler is and what it can do for us. Most readers are already familiar with the TI BASIC language, and many have already experienced the disk-oriented features of Extended BASIC. These BASIC's are "interpreted" languages. This means that when a program is being run, each BASIC statement is converted (interpreted) to machine language—the binary ones and zeros that the computer understands—and then executed. Since a *single* BASIC statement usually generates *several* machine instructions, programs can execute relatively slowly. This is especially true in programs containing loops, because each statement in a loop is interpreted each time it is encountered.

In contrast, assembly language programs are "assembled" before they are run. This means that the assembly language statements are converted to machine language by the assembler; it is the machine-language (or "object") program which is run. That is why programs written in assembly language run so fast; there is no waiting for each instruction to be interpreted at runtime. Programming in assembly language, however, does involve an extra step which is not apparent in BASIC programming—namely the assembler stage. BASIC programs are simply input and RUN, but assembly language programs must be input, then *assembled*, and RUN.

Another major difference between BASIC and assembly language is the difficulty of writing programs. A BASIC program is relatively easy to code because the instructions are English-like and the programmer does not have to worry about where variables reside in memory, or have to understand the structure of the machine. Assembly language programs, on the other hand, are harder and more time-consuming to code because the instructions are machine-oriented (see TMS9900 Machine and Assembly Language Programming, part 2, in the July/August issue) and the programmer must understand the structure of the machine. Debugging assembly language programs is harder, too. But these difficulties are not necessarily disadvantages, since an understanding of the machine allows a programmer to create more efficient programs. Programming in assembly language is an education in itself, and is one of the best ways to learn how a computer works.

A programmer must consider these tradeoffs in choosing the best language for each application. In general, BASIC is faster to code and debug, but assembly language programs execute faster. Happily, TI has made it possible to choose *both* by enabling Extended BASIC programs to CALL assembly language subroutines. This means that a programmer can write mainly in Extended BASIC and use assembly language for portions of the program where faster execution is required (such as loops or especially sorts). This is a good way to ease into assembly language programming, and after some practice you may find yourself writing entire applications in assembly language.

What follows is a preliminary look at TI's new Editor/Assembler package. It is, however, only an *overview* of the product. Future articles will go into more depth on specific features of the software.



A PRELIMINARY LOOK AT THE NEW TI EDITOR/ASSEMBLER

Software Media and Required Hardware

The Editor/Assembler software resides in a Command Module and on a disk. To run it, you'll need at least one disk drive and the 32K expansion RAM. Both the editor and assembler are selectable from menus, and most of the screens include easy-to-understand prompting messages.

The Editor

The editor is used to input assembly language source programs initially, to update programs previously saved on disk, and to print programs. The editor's features compare favorably to those of larger systems.

CUMBERLAND TECHNOLOGY

10 Wagner Drive
Carlisle, PA 17013

99/4(A) Programs

- ENGINEERING
- MATH
- PROGRAMMING AIDS
- GAMES
- Many programs written in Assembly Language

Please send name and address
for a current list

★★ TI-99/4 SOFTWARE ★★

TEXT EDITOR \$99

TEXT/99 - Allows entry, editing, disk or cassette storage and printing of your written text. Upper & lower case. Word oriented for fullest editing freedom: change, insert, delete or move. Chains files for long documents. (Manual \$10.)

PERSPECTIVE PLOTTING \$199

DRAW/99 - For architects, engineers, artists. True perspective or axonometric line drawings of any object you enter. Drives Mauro Eng. MP-250 plotter (11-in paper, about \$800). (Manual \$20.)

NAVAL ARCHITECTURE

99/4 programs for hull design, hydrostatics, performance and most marine design calculations. Inquire.

LETCHER OFFSHORE DESIGN

P.O. Box 104, Southwest Harbor, Maine 04879
Tel. 207-244-7347

Computer Assistance

WE SPECIALIZE
IN ASSEMBLER
AND SCIENTIFIC
PROGRAMMING FOR
THE TI 9900 FAMILY



Computer Assistance
82277 Weiss Road
Creswell, OR 97426
(503) 895-2012

There are two editing modes: Edit Mode and Command Mode. Edit Mode is always used to input a program for the first time, although both modes can be used to change existing programs after loading them from disk or typing them in Edit Mode.

Edit Mode is entered directly from the menu. The screen is a 40 x 24 window on the source program. Function keys allow you to move this window to the right or left in 20-character increments, or up and down 24 lines at a time. (Since most of my assembler programs have fewer than 40 characters per line, I tend to view the leftmost 40 characters and make heavy use of the up and down scrolling.) The four cursor keys are enabled in Edit Mode, making it especially easy to correct typographical errors. Whole lines can be inserted into the text by moving the cursor to the adjacent line and pressing the Insert function key; a new blank line is inserted, and the user simply types in the new line. Similarly, a whole line can be deleted by moving the cursor there and pressing the Delete function key; the line is removed and the line numbers of the following lines are automatically decremented. There are also keys for inserting or deleting characters. A Tab key is also provided for tabbing to columns 8 and 16. Edit Mode makes it very easy to enter new programs, because the user can both type the source program in a natural manner, and correct errors and omissions as they occur. Edit Mode is exited via the Back function key, which puts the editor into Command Mode.

Command Mode reminds me of the UCSD Pascal editor. The first line of the screen shows the Command Mode options: Escape, Find, Replace, Move, Insert, Copy, Delete, Show, and Adjust. Line 2 is reserved for parameters to be input by the user, so in this mode the text window is 40 x 22. Most options require further information to be given on line 2, and very clear prompts are given so the user knows what to enter.

Each option is selected by typing the first character of the option name. For example, to find an occurrence of a string in the source program, the user enters F. The system responds with the prompt <count>< (start col, end col)>/string/. To find the second occurrence of the string ABCD between columns 1 and 50, the user would type 2 (1, 50) / ABCD /. The system would then display the section of text containing the second such occurrence of ABCD (if any) with the cursor over the A. The brackets < > in the prompting message indicate optional parameters. To find the next occurrence of the string ABCD in the whole source program, the user need only type / ABCD /. The Replace option is like the Find, except that each specified occurrence of the string is replaced by a second string given by the user. The Replace includes an

optional verify operator which allows the user to say yes or no to each replacement. The Move option allows the user to move sections of text, indicated by an interval of line numbers, to a different place in the source program. Copy is similar, except that the section of text ends up in *both* the original position and the new position. Delete allows easy removal of several contiguous lines from the text. Insert takes a file from disk and places it anywhere you want in the program being edited. Show is a way of moving the window so that a certain line number is at the top of the screen. Adjust is an easy way to make the line numbers disappear so that the window shows the source program only. Escape gets you out of Command Mode and back to the editor's menu, where you can choose to save the source program to disk, print it, purge it, or edit the same or another program.

The editor performs all line numbering automatically as lines are entered, and maintains these numbers in sequence as lines are added or deleted. The user can refer to them for operating on sections of the program; they also appear on the assembler output listing, which is handy for debugging.

TI has incorporated most of the features found in editors for larger systems into the 99/4 editor. In fact, the abilities to edit at the character, line, and group-of-lines levels are *not* always *all* available in larger editors. The only feature missing from the editor is a variable right margin—a feature which is really not too significant for assembler source programs. [But that would be nice for word processing applications, since this editor already performs 95% of what most people would need for correspondence and document preparation—Ed.]

The Assembler

The assembler is a program which converts assembly language source programs into object form—the machine-language program that executes on the TI-99/4(A). The object program is written to disk. Optionally, a user can print out or write an assembler listing to disk.

The 99/4 assembler is a lot like the 9900 assembler, TXMIRA, which runs on larger TI systems. Most of the 69 instructions are implemented. A programmer who is familiar with TXMIRA will be able to write assembler programs for the 99/4 without too much difficulty, since the same addressing modes are used and most of the instructions operate in the same way.

One big difference, as might be expected, is in the way a programmer handles input and output to the monitor. The 99/4 assembler includes some Basic Support Utilities—a set of built-in subroutines ("macros") for screen I/O. These utilities make it unnecessary to use the CRU (Communica-

Continued on p. 76

How The New Tax Laws Affect Computer Owners

By Vernon K. Jacobs

Should you buy your first (or next) computer this year or next? Should you lease or buy? Are there any other areas of the new tax law that might affect those who own or are thinking of owning a computer? There are at least 109 specific provisions in the "Economic Recovery Tax Act of 1981," and it will be months (perhaps years) before the impact of all the provisions is evaluated. It's almost certain that we will have another tax bill early next year to correct the inevitable technical errors and flaws in this hastily drafted and complex set of tax law changes. Nonetheless, here is a brief summary of some of the provisions of the new tax law that should be of specific concern to computer owners and lessees.

Full Write Off For Small Computers

One of the provisions of the new tax law will permit businesses to deduct the first \$5,000 of business equipment acquired in 1982 and 1983, the first \$7,500 of purchases in 1984 and 1985 and the first \$10,000 of purchases after 1985. This means that many small desktop computers could be fully expensed in the year acquired. No investment credit would be allowed on such purchases but the immediate write off would usually be better. If the cost of the computer exceeds the deductible amount. The excess would be eligible

Vernon K. Jacobs, CPA, is Editor of Tax Angles newsletter and publisher of Financial Systems Report. P. O. Box 8137-P, Prairie Village, KS. 66208.

for the new depreciation method. This full write off provision is not available for investors. It's only available if the equipment is to be used in a trade or business.

New Depreciation Rules

If you purchased a computer in 1981, the 100% write off won't be available, but the new method of depreciation (called the "Asset Cost Recovery System") does apply to 1981 equipment purchased. Under the new method, computers will be depreciated over a five year period using specified rates for each of the five years. (If computers can be classed as research and development equipment they can be depreciated over a three year period.) For five year class equipment purchased in 1981 through 1984, the first year's depreciation will be 15% of the cost. The second years depreciation will be 22% of the cost and the rate will be 21% in each of the next three years. The entire cost will be deducted over the five year period.

By contrast, the prior law permitted a computer owner to write off up to 40% of the cost in the first year if the equipment was placed in service before July first. An additional 24% of the cost would be written off in the second year, 14.4% in the third year and 10.8% in the fourth and fifth years. This assumes a five year life, which has been typical for computer owners. Consequently, owners of larger and more expensive computers won't fare as well under the new

Continued on p. 82

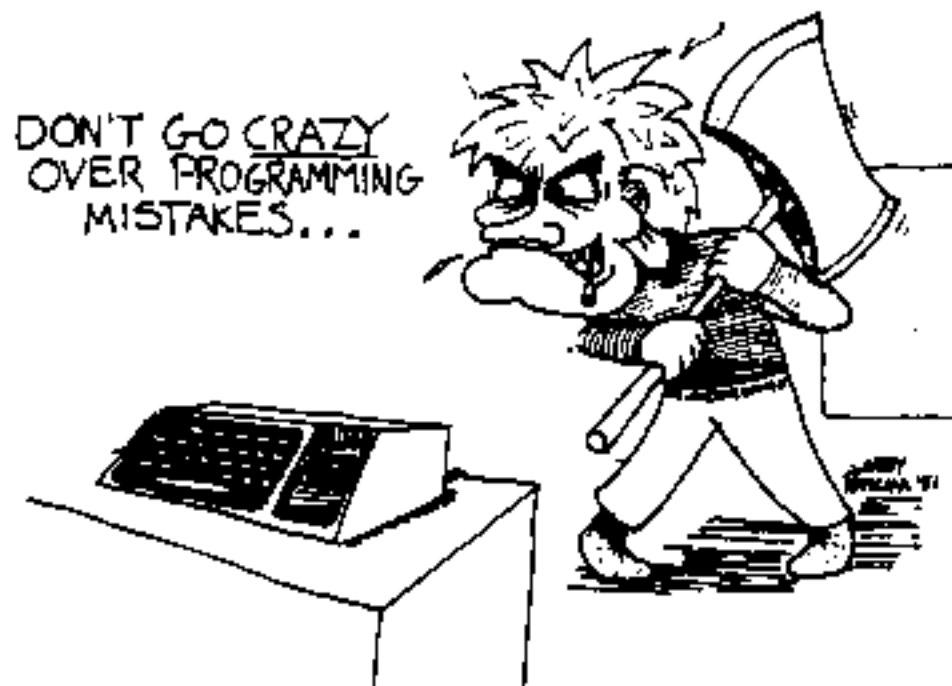
Murphy's Law . . . from p. 32

Rule #3 — Walk, Don't RUN !

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

temptation to correct the error right away. Doing this will cause the following:

- If you change the program line, TI BASIC resets all the variable data (that you entered during your test) to zero or spaces. Subsequently, you will have to retype all that data back into your program.
- You may loose your trend of thought when you stop your test in the middle. It's possible to miss testing a major item because you spent your time fixing a minor problem.
- You may forget to keep track of corrections if you do them in pieces.



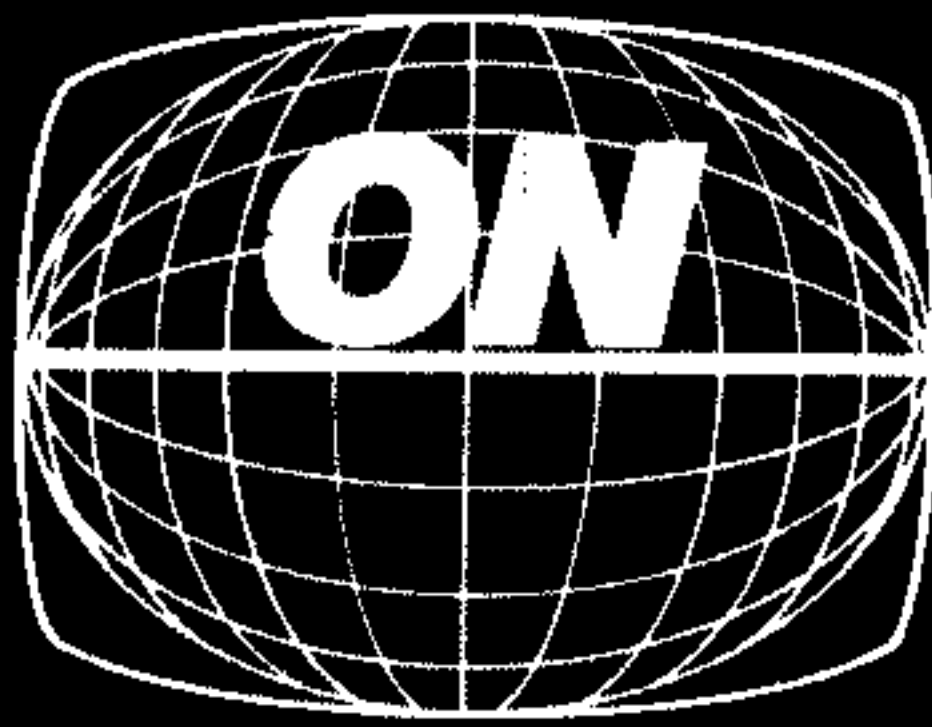
After you have done as much of your test as possible, stop the program and

begin fixing all the "bugs." Don't forget to mark down the statements that you change. This will come in handy if you must restore a previous version because of a problem in making corrections. There are various ways to correct (debug) a program, but since it would require an entire article in itself, we'll hold off on that until a forthcoming issue.

Rule #4 — Test, Test, TEST !

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

Support Our Advertisers. They Too Make This Magazine Possible. . .



LoCAITiON™

THE INTERNATIONAL JOURNAL OF COMPUTER ASSISTED INSTRUCTION



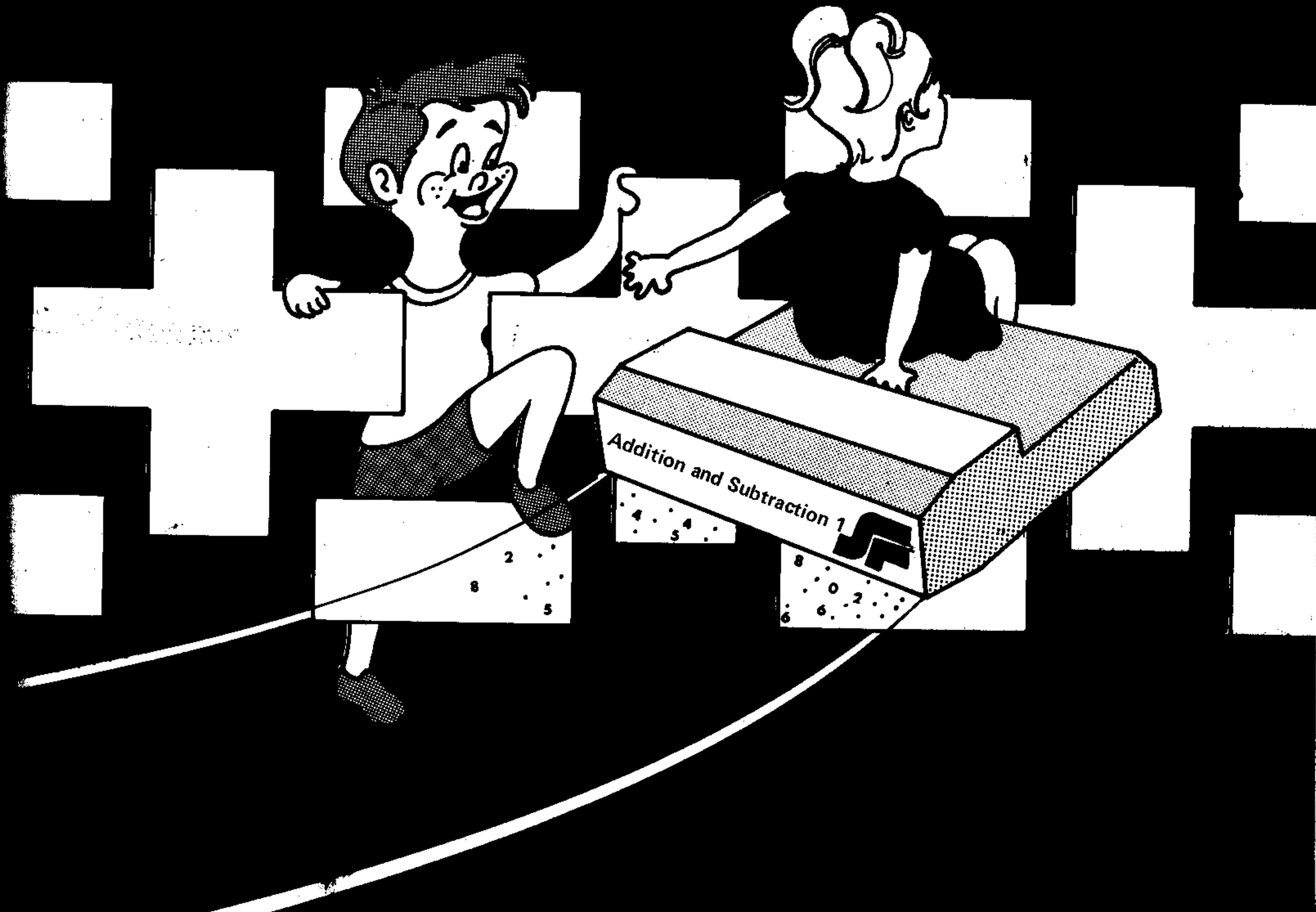
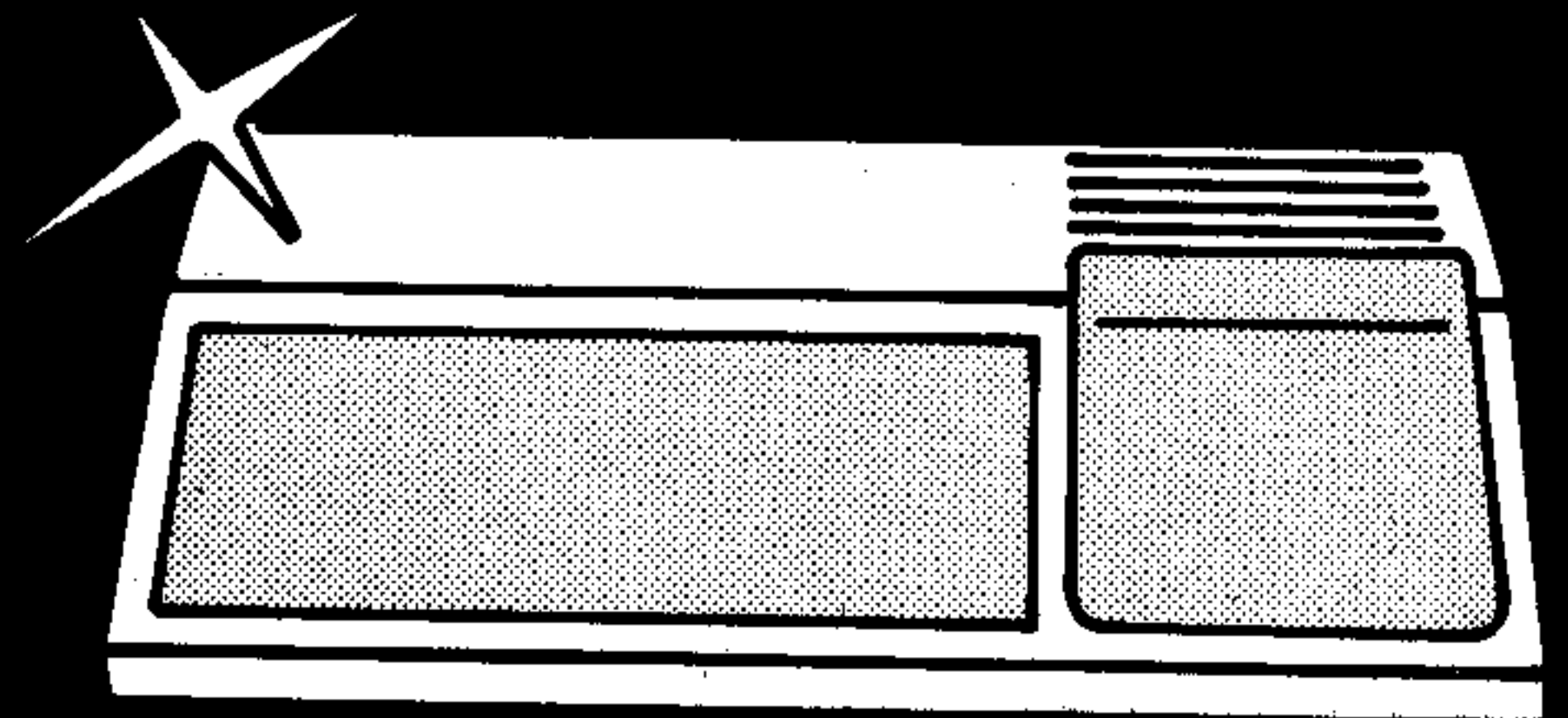
State-of-the-Art CAI

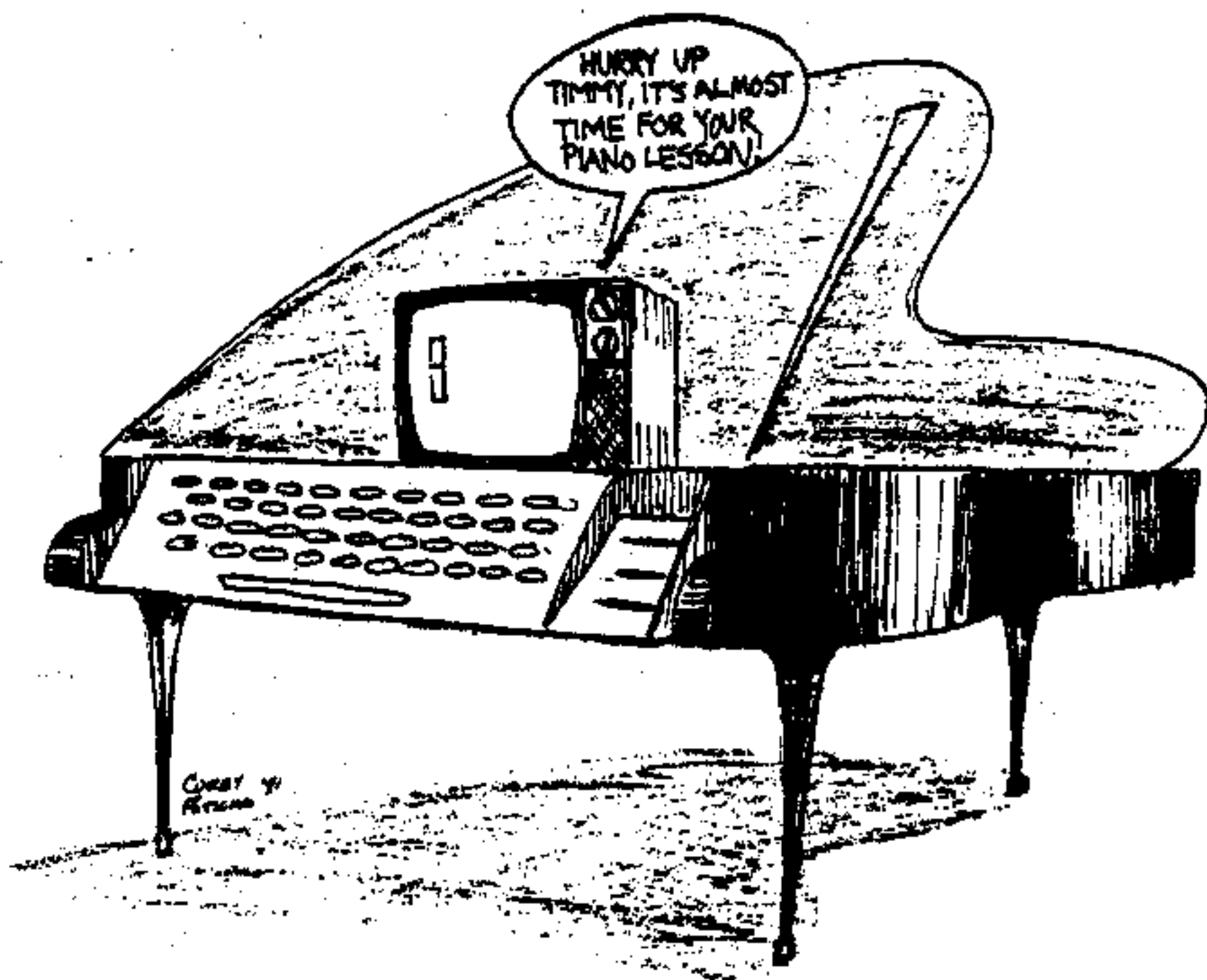
Poetry with LOGO

How to Beat Video Chess

Let's Learn Notes

Typing for Accuracy





LET'S LEARN NOTES

By Regena

"Let's Learn Notes" was designed to be used by beginning music students. A piano or organ teacher can use the program during the lesson to give the student a different approach to learning musical notes, or a student can run the program before or after his/her regular lesson time with the music teacher. Students can also use the program at home for additional practice in learning the musical notes. Even preschool children can begin learning notes with this program.

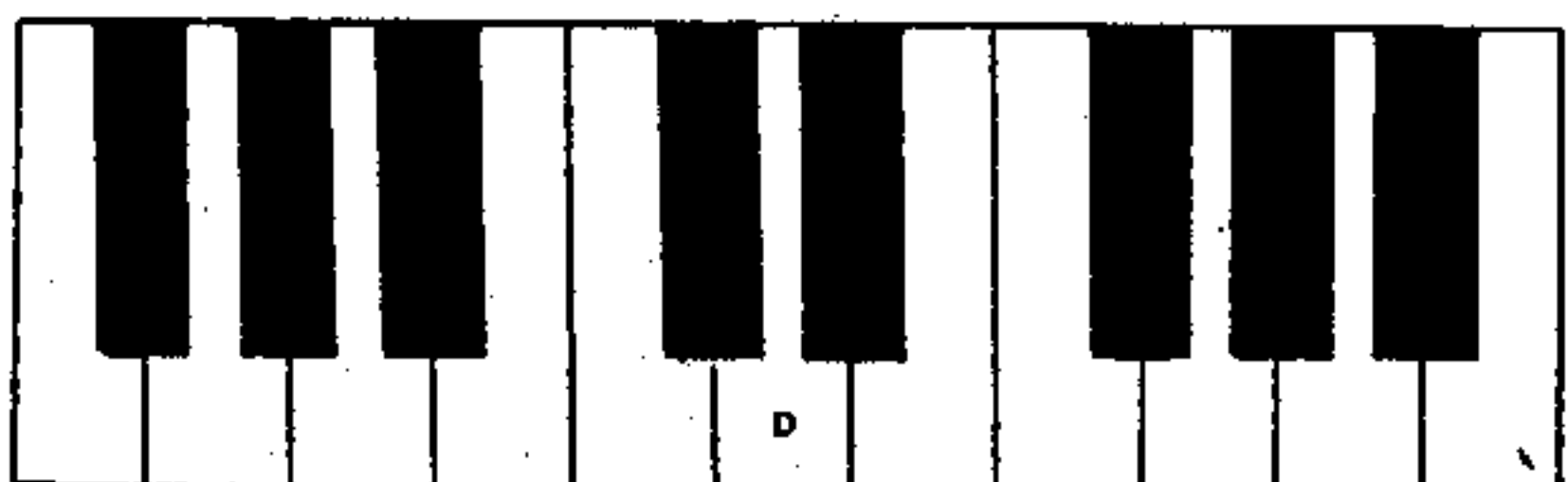
The program is written in TI BASIC and uses color graphics to draw piano keyboards, musical staves, and notes. In addition, the program generates musical tones.

This program provides three options: Keyboard Learning, Treble Clef Learning and Bass Clef Learning. Each option asks for ten responses. An incorrect response is recognized by a slight non-musical noise; the correct response must be entered before the program will continue.

"Keyboard Learning" randomly selects and displays one of two piano keyboards (starting at the left with either two black keys or three black keys). The program randomly selects one of the 11 displayed piano keys and flashes a question mark on the key. The student responds by pressing the letter on the computer keyboard that corresponds to the letter name of the piano key shown. If the response is correct, the corresponding musical tone is played and the letter name is printed on the piano key. The program randomly chooses Keyboard 1 or Keyboard 2 for each question. If the keyboard chosen is the same as for the previous question, the keyboard is not redrawn.

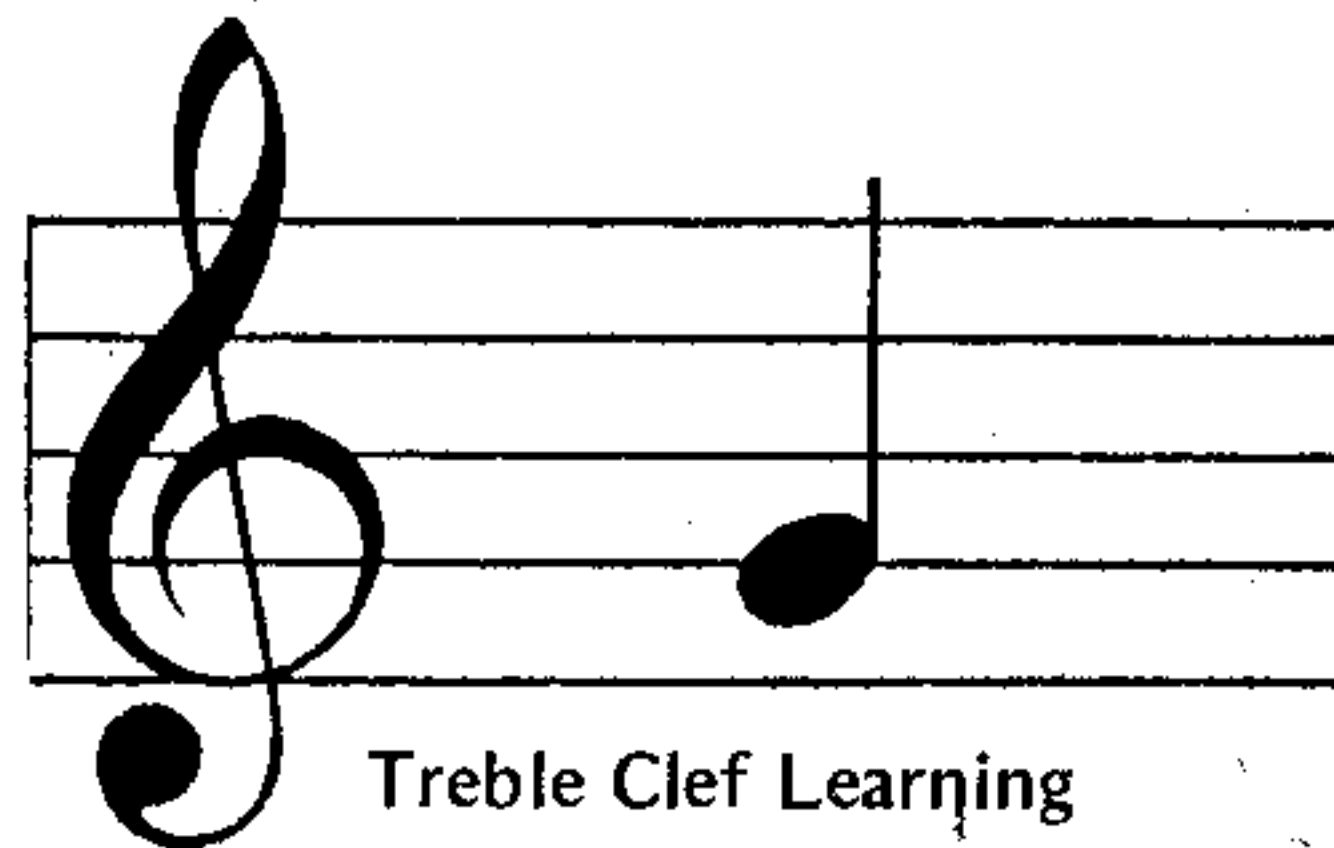


Keyboard 1



Keyboard 2

"Treble Clef Learning" displays a treble staff and clef. A note is selected randomly from Middle C to high F (top line of the staff) and displayed as a red quarter note. The student presses the letter on the computer keyboard that corresponds to the letter name of the note. If the response is correct, the corresponding musical tone is played and the letter name is printed on the note.



Treble Clef Learning

"Bass Clef Learning" displays a bass staff and clef. A note is selected randomly from low G (bottom line of the staff) to Middle C and displayed as a red quarter note. The student presses the letter on the computer keyboard that corresponds to the letter name of the note. If the response is correct, a five-note scale is played and the letter name is printed on the note.



Bass Clef Learning

This program is very easy to use and "student-friendly" —even for the youngest piano learners. A student can select the three learning options at the beginning of the program or after each option has finished, simply by pressing 1, 2, or 3 on the computer keyboard. If a number greater than 3 is pressed, the program ends.

By using this program, repetitious drill is made much more fun for the piano student and much less boring for the teacher. TI's color graphics and sound in this program greatly enhance the student's motivation to learn the letter names of piano keys and notes.

EXPLANATION OF THE PROGRAM
Let's Learn Notes

Line Nos.	Description
150	T=1500 for the CALL SOUND(T, - - - , -) statements.
170-450	Defines colors and characters for the title screen.
460-1220	Displays the characters for musical notes and a treble clef for the title screen. Musical tones of the C major scale and arpeggio are played while the title screen is being displayed.
1230	Asks which option the student wants and branches to that option.
1240-1340	Option 1, Keyboard. Defines colors and characters for drawing the keyboard.
1360	COUNT set to zero and incremented for each question. There are 10 questions in each option.
1370	Keyboard number is randomly chosen, 1 or 2.
1380-1440	Prints "NAME THE KEY"
1450-1550	Draws the white keys.
1570-1730	Draws the black keys for one of the two piano keyboards.
1740-1750	Chooses one of the 11 keys randomly.
1760-1810	Blinks a red question mark on the key.
1820-1830	Reads the student's response.
1840-2680	Tests the response. If it is incorrect there is a nonmusical sound and another response is required. If it is correct the corresponding musical tone is played and the letter name of the key is displayed on the key.
2690-2710	Delays then erases the letter name.
2720-2730	Increments COUNT and checks if there have been 10 questions.
2740-2750	Chooses keyboard pattern randomly. If it is the same as the previous question only a new key is chosen; if it is different a new keyboard is drawn before the key is chosen.
2760	Treble Clef and Bass Clef option.
2770-2800	Resets colors for this screen.
2810-3270	Defines special characters for staff, treble clef, and note.
3280	Draws staff.
3290-3340	Prints "NAME THE NOTE"
3360-3450	Treble Clef option. Draws treble clef.
3460	Sets COUNT for number of problems.
3470-3530	Chooses note and draws it.
3540-3550	Reads the student's response.
3560-4010	Tests the response. If it is incorrect there is a nonmusical sound and another response is required. If it is correct the corresponding musical tone is played and the letter name is displayed on the note.
4020-4030	After a delay erases the note and chooses a new note. If there have been 10 notes the options are listed again.
4040-4160	Bass Clef option. Defines special characters for the bass clef.
4170-4230	Prints bass clef.
4240	Sets COUNT=0 for the number of problems.
4250-4310	Chooses one of 11 notes randomly and draws it.
4320-4330	Reads the student's response.
4340-4900	Tests the response. If it is incorrect there is a nonmusical sound and another response is required. If it is correct a five-note scale is played starting at frequency J and the letter name is displayed on the note.
4910-4920	After a delay erases the note and chooses a new note. If there have been 10 notes the options are listed again.
4930-4970	Subroutine for playing the 5-note scale.
4980-5060	Subroutine for drawing the staff.
5070-5210	Subroutine for drawing the note.
5130-5210	Draws the stem of the note up or down from the note depending on where the note is.
5220-5390	Subroutine for procedure after each note.
5230-5240	Delays.
5250-5260	Increments and tests COUNT.
5260-5390	If COUNT<10, erase the note and returns.
5400-5430	If COUNT=10, prints menu screen of options.
5540-5490	Branches to Option 1, 2, or 3.
5500	If 4 is pressed, the program ends.

```

100 REM *****
110 REM LET'S LEARN NOTES
120 REM *****
130 REM BY REGENA
140 REM 99'ER VERSION 9.81.1
150 T=1500
160 CALL CLEAR
170 CALL COLOR(11,13,1)
180 CALL COLOR(12,13,1)
190 CALL COLOR(13,14,1)
200 CALL COLOR(14,5,1)
210 CALL COLOR(15,5,1)
220 CALL CHAR(112,"0000020705050505")
230 CALL CHAR(113,"0505050303020202")
240 CALL CHAR(114,"050D091030206040")
250 CALL CHAR(115,"00000080B0B09CE3")
260 CALL CHAR(116,"40C1C2C2868686C2")
270 CALL CHAR(117,"C040404020202020")
280 CALL CHAR(118,"8080404040404040")
290 CALL CHAR(119,"C1404020100C07")
300 CALL CHAR(120,"20201011121CF010")
310 CALL CHAR(121,"B0B0B0")
320 CALL CHAR(122,"109090E0")
330 CALL SOUND(T,262,2)
340 CALL CHAR(136,"0000030F1F1F3F3F")
350 CALL CHAR(137,"0202E2FAFAFEFEFE")
360 CALL CHAR(138,"1F1F0F03")
370 CALL CHAR(139,"FCFCFBEO")
380 CALL CHAR(140,"0202020202020202")
390 CALL CHAR(141,"0000000303020202")
400 CALL CHAR(142,"0000000000804020")
410 CALL CHAR(143,"2020101010080808")
420 CALL CHAR(144,"00011EE")
430 CALL CHAR(145,"OFF0")
440 CALL CHAR(146,"0000000000031CE0")
450 CALL CHAR(147,"000000003EC20202")
460 CALL SOUND(T,294,2)
470 DATA 76,69,84,59,83,32,76,69,65,82,78,32
480 RESTORE 470
490 FOR Y=12 TO 23
500 READ L
510 CALL HCHAR(6,Y,L)
520 NEXT Y
530 CALL SOUND(T,330,2)
540 DATA 78,79,84,69,83,32
550 RESTORE 540
560 FOR Y=13 TO 23 STEP 2
570 READ L
580 CALL HCHAR(10,Y,L)
590 NEXT Y
600 CALL SOUND(T,349,2)
610 DATA 8,4,138,8,5,139,7,4,136,7,5,137,6,5,140,
5,5,140,4,5,141,4,6,144,4,7,145
620 RESTORE 610
630 FOR I=1 TO 9
640 READ X,Y,GR
650 CALL HCHAR(X,Y,GR)
660 NEXT I
670 CALL SOUND(T,392,2)
680 DATA 3,8,146,3,9,147,4,9,140,5,9,140,6,9,137,
6,8,136,7,8,138,7,9,139
690 RESTORE 680
700 FOR I=1 TO 8
710 READ X,Y,GR
720 CALL HCHAR(X,Y,GR)
730 NEXT I
740 CALL SOUND(T,440,2)
750 CALL VCHAR(16,12,49)
760 DATA 75,69,89,66,79,65,82,68
770 RESTORE 760
780 FOR Y=14 TO 21
790 READ L
800 CALL HCHAR(16,Y,L)
810 NEXT Y
820 CALL SOUND(T,494,2)
830 DATA 4,26,112,5,26,113,6,26,114,6,27,115,7,
26,116,7,27,117
840 RESTORE 830
850 FOR I=1 TO 6

```



Let's Learn Notes ...

```

860 READ X,Y,GR
870 CALL HCHAR(X,Y,GR)
880 NEXT I
890 CALL SOUND(T,523,2)
900 DATA 7,28,118,8,26,119,8,27,120,8,28,121,
9,27,122
910 RESTORE 900
920 CALL SOUND(T,262,2)
930 FOR I=1 TO 5
940 READ X,Y,GR
950 CALL HCHAR(X,Y,GR)
960 NEXT I
970 CALL SOUND(T,330,2)
980 CALL HCHAR(18,12,80)
990 DATA 84,82,69,66,76,69,32,67,76,69,70
1000 RESTORE 990
1010 FOR Y=14 TO 24
1020 READ L
1030 CALL HCHAR(18,Y,L)
1040 NEXT Y
1050 CALL SOUND(T,392,2)
1060 CALL SOUND(T,523,2)
1070 CALL HCHAR(20,12,51)
1080 DATA 66,65,63,63,32,67,76,69,70
1090 RESTORE 1080
1100 FOR Y=14 TO 22
1110 READ L
1120 CALL HCHAR(20,Y,L)
1130 NEXT Y
1140 CALL SOUND(T,392,2)
1150 DATA 18,27,138,18,28,139,17,27,136,17,
28,137,16,28,140,15,28,140,14,28,141,
14,29,142,15,29,143
1160 RESTORE 1150
1170 FOR I=1 TO 9
1180 READ X,Y,GR
1190 CALL HCHAR(X,Y,GR)
1200 NEXT I
1210 CALL SOUND(T,330,2)
1220 CALL SOUND(T,262,2)
1230 GOTO 5430
1240 CALL CLEAR
1250 CALL COLOR(4,2,7)
1260 CALL COLOR(9,2,1)
1270 CALL COLOR(10,2,16)
1280 CALL CHAR(96,"FF")
1290 CALL CHAR(104,"FF")
1300 CALL CHAR(105,"7F7F7F7F7F7F7F7F")
1310 CALL CHAR(106,"FEFEFEFEFEFEFEFE")
1320 CALL CHAR(107,"0101010101010101")
1330 CALL CHAR(108,"8080808080808080")
1340 CALL CHAR(109,"0")
1350 RANDOMIZE
1360 COUNT=0
1370 KB=INT(2*RND)+1
1380 CALL CLEAR
1390 DATA 78,65,77,69,32,84,72,69,32,75,69,89
1400 RESTORE 1390
1410 FOR Y=11 TO 22
1420 READ L
1430 CALL HCHAR(4,Y,L)
1440 NEXT Y
1450 PAT=KB
1460 REM DRAW KEYBOARD
1470 CALL HCHAR(8,1,104,32)
1480 CALL HCHAR(18,1,96,32)
1490 CALL VCHAR(8,1,109,10)
1500 CALL VCHAR(8,32,109,10)
1510 FOR Y=3 TO 30 STEP 3
1520 CALL VCHAR(8,Y-1,109,10)
1530 CALL VCHAR(8,Y,107,10)
1540 CALL VCHAR(8,Y+1,108,10)
1550 NEXT Y
1560 IF KB=2 THEN 1660
1570 REM KEYBOARD 1
1580 DATA 3,6,12,15,18,24,27,27
1590 RESTORE 1580
1600 FOR I=1 TO 8
1610 READ Y
1620 CALL VCHAR(8,Y,105,7)
1630 CALL VCHAR(8,Y+1,106,7)
1640 NEXT I
1650 GOTO 1740
1660 REM KEYBOARD 2
1670 DATA 3,6,9,15,18,24,27,30,30
1680 RESTORE 1670
1690 FOR I=1 TO 9
1700 READ Y
1710 CALL VCHAR(8,Y,105,7)

```

```

1720 CALL VCHAR(8,Y+1,106,7)
1730 NEXT I
1740 REM PICK KEY
1750 NN=INT(11*RND)+1
1760 J=3*NN-1
1770 CALL HCHAR(16,J,63)
1780 FOR I=1 TO 10
1790 CALL COLOR(4,2,7)
1800 CALL COLOR(4,16,7)
1810 NEXT I
1820 CALL KEY(0,NOTE,STATUS)
1830 IF STATUS<>1 THEN 1820
1840 IF KB=2 THEN 1860
1850 ON NN GOTO 2050,2120,2190,2260,2330,2400,2470,
2540,2580,2620,2660
1860 ON NN GOTO 1890,1930,1970,2010,2050,2120,2190,
2260,2330,2400,2470
1870 CALL SOUND(500,-4,2)
1880 GOTO 1820
1890 IF NOTE<>70 THEN 1870
1900 CALL SOUND(T,175,2)
1910 CALL HCHAR(16,2,70)
1920 GOTO 2690
1930 IF NOTE<>71 THEN 1870
1940 CALL SOUND(T,196,2)
1950 CALL HCHAR(16,5,71)
1960 GOTO 2690
1970 IF NOTE<>65 THEN 1870
1980 CALL SOUND(T,220,2)
1990 CALL HCHAR(16,8,65)
2000 GOTO 2690
2010 IF NOTE<>66 THEN 1870
2020 CALL SOUND(T,247,2)
2030 CALL HCHAR(16,11,66)
2040 GOTO 2690
2050 IF NOTE<>67 THEN 1870
2060 CALL SOUND(T,262,2)
2070 IF KB=2 THEN 2100
2080 CALL HCHAR(16,2,67)
2090 GOTO 2690
2100 CALL HCHAR(16,14,67)
2110 GOTO 2690
2120 IF NOTE<>68 THEN 1870
2130 CALL SOUND(T,294,2)
2140 IF KB=2 THEN 2170
2150 CALL HCHAR(16,5,68)
2160 GOTO 2690
2170 CALL HCHAR(16,17,68)
2180 GOTO 2690
2190 IF NOTE<>69 THEN 1870
2200 CALL SOUND(T,330,2)
2210 IF KB=2 THEN 2240
2220 CALL HCHAR(16,8,69)
2230 GOTO 2690
2240 CALL HCHAR(16,20,69)
2250 GOTO 2690
2260 IF NOTE<>70 THEN 1870
2270 CALL SOUND(T,349,2)
2280 IF KB=2 THEN 2310
2290 CALL HCHAR(16,11,70)
2300 GOTO 2690
2310 CALL HCHAR(16,23,70)
2320 GOTO 2690
2330 IF NOTE<>71 THEN 1870
2340 CALL SOUND(T,392,2)
2350 IF KB=2 THEN 2380
2360 CALL HCHAR(16,14,71)
2370 GOTO 2690
2380 CALL HCHAR(16,26,71)
2390 GOTO 2690
2400 IF NOTE<>65 THEN 1870
2410 CALL SOUND(T,440,2)
2420 IF KB=2 THEN 2450
2430 CALL HCHAR(16,17,65)
2440 GOTO 2690
2450 CALL HCHAR(16,29,65)
2460 GOTO 2690
2470 IF NOTE<>66 THEN 1870
2480 CALL SOUND(T,494,2)
2490 IF KB=2 THEN 2520
2500 CALL HCHAR(16,20,66)
2510 GOTO 2690
2520 CALL HCHAR(16,32,66)
2530 GOTO 2690
2540 IF NOTE<>67 THEN 1870
2550 CALL SOUND(T,523,2)
2560 CALL HCHAR(16,23,67)
2570 GOTO 2690
2580 IF NOTE<>68 THEN 1870
2590 CALL SOUND(T,587,2)

```

Let's Learn Notes...

```

2600 CALL HCHAR(16,26,68)
2610 GOTO 2690
2620 IF NOTE<>69 THEN 1870
2630 CALL SOUND(T,659,2)
2640 CALL HCHAR(16,29,69)
2650 GOTO 2690
2660 IF NOTE<>70 THEN 1870
2670 CALL SOUND(T,698,2)
2680 CALL HCHAR(16,32,70)
2690 FOR DELAY=1 TO 1000
2700 NEXT DELAY
2710 CALL HCHAR(16,J,109)
2720 COUNT=COUNT+1
2730 IF COUNT=10 THEN 5400
2740 KB=INT(2*RND)+1
2750 IF KB=PAT THEN 1740 ELSE 1380
2760 REM TREBLE CLEF NOTES
2770 CALL CLEAR
2780 FOR I=9 TO 15
2790 CALL COLOR(I,2,1)
2800 NEXT I
2810 CALL CHAR(96,"FF")
2820 CALL CHAR(104,"0000071F3F78F0E0")
2830 CALL CHAR(105,"000000C0F0F0783C")
2840 CALL CHAR(106,"C0B0B0B0B0B0B0B0")
2850 CALL CHAR(107,"1C0C0C0C0C040404")
2860 CALL CHAR(108,"80B0B0B0B0B0B040")
2870 CALL CHAR(109,"0404040408080818")
2880 CALL CHAR(110,"FF40404040202020")
2890 CALL CHAR(111,"FF3040604040C080")
2900 CALL CHAR(112,"21213312161C1C18")
2910 CALL CHAR(113,"FF0000103070F1E")
2920 CALL CHAR(114,"FF78EBC8B4840404")
2930 CALL CHAR(115,"000000010103070F")
2940 CALL CHAR(116,"3C78F0E0C0B080")
2950 CALL CHAR(117,"0404020202020202")
2960 CALL CHAR(118,"FF0E1C1838307070")
2970 CALL CHAR(119,"FF01070F1F3D3971")
2980 CALL CHAR(120,"FFFFFFC0")
2990 CALL CHAR(121,"FFFEFF0F0301")
3000 CALL CHAR(122,"FF0000B0C0C0E060")
3010 CALL CHAR(123,"606060E0E0606060")
3020 CALL CHAR(125,"60E0C0C0C0C0C0E0")
3030 CALL CHAR(126,"80B0B0B0B0B0B080")
3040 CALL CHAR(127,"3030101010101010")
3050 CALL CHAR(128,"FF707038383C1C1E")
3060 CALL CHAR(129,"FF60603010")
3070 CALL CHAR(130,"FF40404040404040")
3080 CALL CHAR(131,"FF1020202040C080")
3090 CALL CHAR(132,"0F070301")
3100 CALL CHAR(133,"00B0C0F0F83C0F03")
3110 CALL CHAR(134,"00000000000000C0")
3120 CALL CHAR(135,"202020202020212E")
3130 CALL CHAR(136,"01030608106080")
3140 CALL CHAR(137,"FF20202020202020")
3150 CALL CHAR(138,"1010101010101010")
3160 CALL CHAR(139,"3C7EF0F0F07C391F")
3170 CALL CHAR(140,"10101030204080")
3180 CALL CHAR(141,"FF80B0B0B0B080B0")
3190 CALL CHAR(152,"030F1F3F7FFFFFFF")
3200 CALL CHAR(153,"FCFFFFFFF0F0F0F0")
3210 CALL CHAR(154,"FFFFFFF7F3F1F0F03")
3220 CALL CHAR(155,"FFFFFFF0F0F0F0FC")
3230 CALL CHAR(156,"F0F0F0E0C080")
3240 CALL CHAR(157,"101090D0F0F0F0F0")
3250 CALL CHAR(158,"000080C0E0F0F0F0")
3260 CALL CHAR(159,"0101010101010101")
3270 CALL COLOR(16,7,1)
3280 GOSUB 4980
3290 DATA 78,65,77,69,32,84,72,69,32,78,79,84,69
3300 RESTORE 3290
3310 FOR Y=10 TO 22
3320 READ L
3330 CALL HCHAR(6,Y,L)
3340 NEXT Y
3350 IF GAME=3 THEN 4040
3360 REM TREBLE CLEF
3370 DATA 9,5,104,9,6,105,10,5,106,10,6,107,11,5,
108,11,6,109,12,5,110,12,6,111,13,5,112
3380 RESTORE 3370
3390 DATA 14,4,113,14,5,114,15,3,115,15,4,116,15,
5,117,16,3,118,16,5,119,16,6,120,16,7,121,
16,8,122
3400 DATA 17,3,123,17,5,125,17,6,126,17,8,127,18,
3,128,18,5,129,18,6,130,18,8,131,19,3,132
3410 DATA 19,4,133,19,5,134,19,6,135,19,7,136,20,
6,137,21,6,138,22,5,139,22,6,140,1,1,32
3420 FOR I=1 TO 37
3430 READ X,Y,GR

```

```

3440 CALL HCHAR(X,Y,GR)
3450 NEXT I
3460 COUNT=0
3470 RANDOMIZE
3480 NN=INT(11*RND)+1
3490 M=22-NN
3500 GOSUB 5070
3510 IF NN<>1 THEN 3540
3520 CALL HCHAR(M+1,14,96)
3530 CALL HCHAR(M+1,18,96)
3540 CALL KEY(0,NOTE,STATUS)
3550 IF STATUS<>1 THEN 3540
3560 ON NN GOTO 3590,3630,3670,3710,3750,3790,
3830,3870,3910,3950,3990
3570 CALL SOUND(600,-8,2)
3580 GOTO 3540
3590 IF NOTE<>67 THEN 3570
3600 CALL SOUND(T,262,2)
3610 CALL HCHAR(M+1,16,67)
3620 GOTO 4020
3630 IF NOTE<>68 THEN 3570
3640 CALL SOUND(T,294,2)
3650 CALL HCHAR(M+1,16,68)
3660 GOTO 4020
3670 IF NOTE<>69 THEN 3570
3680 CALL SOUND(T,330,2)
3690 CALL HCHAR(M+1,16,69)
3700 GOTO 4020
3710 IF NOTE<>70 THEN 3570
3720 CALL SOUND(T,349,2)
3730 CALL HCHAR(M+1,16,70)
3740 GOTO 4020
3750 IF NOTE<>71 THEN 3570
3760 CALL SOUND(T,392,2)
3770 CALL HCHAR(M+1,16,71)
3780 GOTO 4020
3790 IF NOTE<>65 THEN 3570
3800 CALL SOUND(T,440,2)
3810 CALL HCHAR(M+1,16,65)
3820 GOTO 4020
3830 IF NOTE<>66 THEN 3570
3840 CALL SOUND(T,494,2)
3850 CALL HCHAR(M+1,16,66)
3860 GOTO 4020
3870 IF NOTE<>67 THEN 3570
3880 CALL SOUND(T,523,2)
3890 CALL HCHAR(M+1,16,67)
3900 GOTO 4020
3910 IF NOTE<>68 THEN 3570
3920 CALL SOUND(T,587,2)
3930 CALL HCHAR(M+1,16,68)
3940 GOTO 4020
3950 IF NOTE<>69 THEN 3570
3960 CALL SOUND(T,659,2)
3970 CALL HCHAR(M+1,16,69)
3980 GOTO 4020
3990 IF NOTE<>70 THEN 3570
4000 CALL SOUND(T,698,2)
4010 CALL HCHAR(M+1,16,70)
4020 GOSUB 5220
4030 GOTO 3480
4040 CALL CHAR(97,"FF000000003078FC")
4050 CALL CHAR(98,"FC7830")
4060 CALL CHAR(99,"FFFF7E3C")
4070 CALL CHAR(100,"18102060607C7EFF")
4080 CALL CHAR(101,"FF00000103060408")
4090 CALL CHAR(102,"FF0F70C0")
4100 CALL CHAR(103,"FFC03B1C0E070303")
4110 CALL CHAR(124,"C0C060303B383838")
4120 CALL CHAR(142,"FF1C1C1C1C1C1C")
4130 CALL CHAR(143,"1C1C1C1C1C1C1C")
4140 CALL CHAR(144,"FF3838387070C0C0")
4150 CALL CHAR(145,"030306040C1870C0")
4160 CALL CHAR(146,"FF03060C186080")
4170 DATA 14,3,99,13,5,100,12,3,101,12,4,102,12,
6,103,13,7,124,14,7,142
4180 RESTORE 4170
4190 DATA 15,7,143,16,7,144,17,6,145,18,5,146,12,
9,97,13,9,98,14,9,97,15,9,98,1,1,32
4200 FOR I=1 TO 16
4210 READ X,Y,GR
4220 CALL HCHAR(X,Y,GR)
4230 NEXT I
4240 COUNT=0
4250 RANDOMIZE
4260 NN=INT(11*RND)+1
4270 M=20-NN
4280 GOSUB 5070
4290 IF NN<>11 THEN 4320
4300 CALL HCHAR(M+1,14,96)

```

Fantasy Computing is dedicated to the idea that computer gaming should be challenging, fun and lasting in content.

AVAILABLE NOW!

"RINGWRAITH'S LAIR"

A fantasy gaming program in which the player tries to acquire treasure, win battles and survive to free a captured princess. Playing time varies with skill but cannot be mastered easily and then maximizing score will take much longer. \$24.95

"RINGWRAITH'S LAIR, SCENARIO II"

This disk is used in conjunction with "RINGWRAITH'S LAIR" disk in order to create a new lair — totally different from the basic scenario. \$12.00

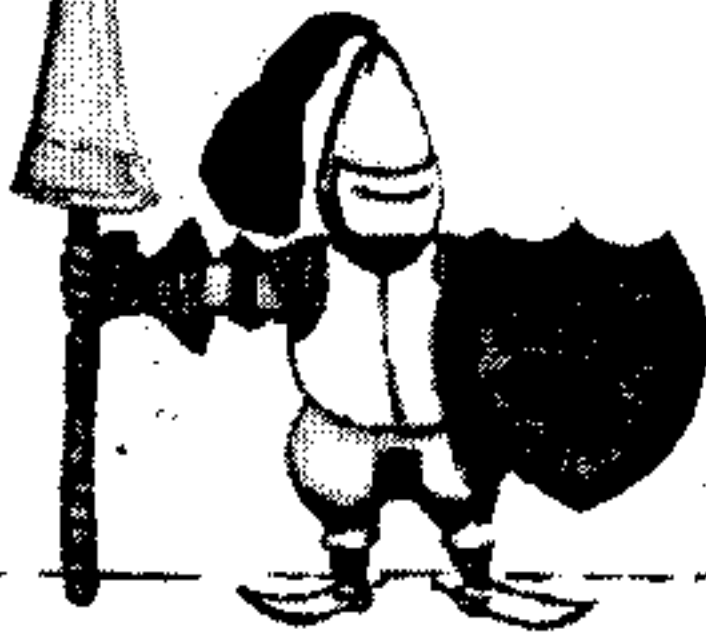
"RINGWRAITH'S LAIR, SCENARIO III"

Still more adventure in a new scenario! \$12.00

"LAIR DESIGNER"

Allows YOU to design and run your own "LAIR" within "RINGWRAITH'S LAIR" format. You design your own monsters, devices and location descriptions and store it all on disk. \$29.95

All of the above require TI-99/4 with disk, expansion memory, extended basic.



ORDER DIRECT
OR
WRITE FOR BROCHURE

Fantasy Computing

1586 South Citrus
Escondido, CA 92027

Advertise In

99'erTM
magazine

**Call or Write for
Rate Card Today!**

99'er Magazine / Ad. Dept.
2715 Terrace View Drive
Eugene, Oregon 97405
(503)-485-8796

ATTENTION TI DEALERS

Did You Know That You Can Be
Reimbursed For Advertising Your
Products In This Magazine?

Call or Write About Space Ads
or Dealer Directory Today!

Let's Learn Notes . . .

```

4310 CALL HCHAR(M+1,18,96)
4320 CALL KEY(O,NOTE,STATUS)
4330 IF STATUS<>1 THEN 4320
4340 ON NN-GOTO 4370,4420,4470,
4520,4570,4620,4670,4720,
4770,4820,4870
4350 CALL SOUND(600,-8,2)
4360 GOTO 4320
4370 IF NOTE<>71 THEN 4350
4380 J=110
4390 GOSUB 4930
4400 CALL HCHAR(20,16,71)
4410 GOTO 4910
4420 IF NOTE<>65 THEN 4350
4430 J=116
4440 GOSUB 4930
4450 CALL HCHAR(19,16,65)
4460 GOTO 4910
4470 IF NOTE<>66 THEN 4350
4480 J=123
4490 GOSUB 4930
4500 CALL HCHAR(18,16,66)
4510 GOTO 4910
4520 IF NOTE<>67 THEN 4350
4530 J=131
4540 GOSUB 4930
4550 CALL HCHAR(17,16,67)
4560 GOTO 4910
4570 IF NOTE<>68 THEN 4350
4580 J=147
4590 GOSUB 4930
4600 CALL HCHAR(16,16,68)
4610 GOTO 4910
4620 IF NOTE<>69 THEN 4350
4630 J=165
4640 GOSUB 4930
4650 CALL HCHAR(15,16,69)
4660 GOTO 4910
4670 IF NOTE<>70 THEN 4350
4680 J=175
4690 GOSUB 4930
4700 CALL HCHAR(14,16,70)

```

```

4710 GOTO 4910
4720 IF NOTE<>71 THEN 4350
4730 J=196
4740 GOSUB 4930
4750 CALL HCHAR(13,16,71)
4760 GOTO 4910
4770 IF NOTE<>65 THEN 4350
4780 J=220
4790 GOSUB 4930
4800 CALL HCHAR(12,16,65)
4810 GOTO 4910
4820 IF NOTE<>66 THEN 4350
4830 J=247
4840 GOSUB 4930
4850 CALL HCHAR(11,16,66)
4860 GOTO 4910
4870 IF NOTE<>67 THEN 4350
4880 J=262
4890 GOSUB 4930
4900 CALL HCHAR(10,16,67)
4910 GOSUB 5220
4920 GOTO 4260
4930 FOR I=1 TO 5
4940 CALL SOUND(200,J,2)
4950 J=J+15
4960 NEXT I
4970 RETURN
4980 REM STAFF
4990 FOR X=12 TO 20 STEP 2
5000 CALL HCHAR(X,1,96,31)
5010 NEXT X
5020 FOR X=12 TO 18 STEP 2
5030 CALL HCHAR(X,1,141)
5040 CALL HCHAR(X+1,1,126)
5050 NEXT X
5060 RETURN
5070 REM NOTE
5080 CALL HCHAR(M,15,152)
5090 CALL HCHAR(M,16,153)
5100 CALL HCHAR(M+1,15,154)
5110 CALL HCHAR(M+1,16,155)
5120 CALL HCHAR(M+1,17,156)

```

```

5130 IF GAME=3 THEN 5210
5140 IF NN<8 THEN 5180
5150 CALL HCHAR(M,17,158)
5160 CALL VCHAR(M+1,14,159,6)
5170 RETURN
5180 CALL HCHAR(M,17,157)
5190 CALL VCHAR(M-5,17,138,5)
5200 RETURN
5210 IF NN<5 THEN 5180 ELSE 5150
5220 REM CHANGE NOTES
5230 FOR DELAY=1 TO 1000
5240 NEXT DELAY
5250 COUNT=COUNT+1
5260 IF COUNT=10 THEN 5400
5270 CALL HCHAR(M,14,32,5)
5280 CALL HCHAR(M+1,14,32,5)
5290 IF GAME=3 THEN 5350
5300 IF NN<8 THEN 5330
5310 CALL VCHAR(M+1,14,32,6)
5320 GOTO 5360
5330 CALL VCHAR(M-5,17,32,5)
5340 GOTO 5360
5350 IF NN<5 THEN 5330 ELSE 5310
5360 FOR K=12 TO 20 STEP 2
5370 CALL HCHAR(K,14,96,5)
5380 NEXT K
5390 RETURN
5400 PRINT "PRESS 1 FOR KEYBOARD QUIZ"
5410 PRINT TAB(7);"2 FOR TREBLE NOTES"
5420 PRINT TAB(7);"3 FOR BASS NOTES"
5430 PRINT TAB(7);"4 TO END PROGRAM"
5440 CALL KEY(O,K,S)
5450 IF K=52 THEN 5500
5460 IF K>51 THEN 5440
5470 IF K<49 THEN 5440
5480 GAME=K-48
5490 IF GAME=1 THEN 1240 ELSE 2770
5500 END

```

HUGE ELEK-TEK DISCOUNTS ON TI-99/4A Home Computer System



TEXAS INSTRUMENTS
INCORPORATED

Model	Name	Mfr. Sugg. Ret.	Elek-Tek Price	Model	Name	Mfr. Sugg. Ret.	Elek-Tek Price
CONSOLE							
PHC	004 TI-99/4A Home Computer	649.95	370.00	PHD	5019 Teach Yourself Extended Basic*	24.95	21.20
PERIPHERALS							
PHP	1500 Solid State Speech™ Synthesizer	149.95	125.00	PHD	5020 Music Maker Demonstration	14.95	12.70
PHP	1600 Telephone Coupler (Modem)	224.95	185.00	PHD	5023 Basketball Statistics*	24.95	21.20
PHP	1700 RS-232 Accessories Interface	224.95	185.00	PHD	5025 Bridge Bidding I*	29.95	25.45
PHP	1800 Disk Drive Controller (One Disk Manager module packed with each Disk Controller)	299.95	238.00	Cassette			
PHP	1850 Disk Memory Drive	499.95	395.00	PHT	6007 Teach Yourself Basic	29.95	25.45
PHP	1900 Solid State Printer	399.95	318.00	PHT	6009 Music Skills Trainer	24.95	21.20
PHP	2200 Memory Expansion (32K RAM)*	399.95	318.00	PHT	6011 Computer Music Box	14.95	12.70
PHA	2100 R.F. Modulator (TV Adapter)	49.95	40.00	PHT	6018 Market Simulation	14.95	12.70
PHA	4100 10" Color Monitor*	374.95	300.00	PHT	6019 Teach Yourself Extended Basic*	19.95	16.95
OPTIONAL ACCESSORIES							
PHP	1100 Wired Remote Controllers (Pair)	34.95	28.00	PHT	6025 Bridge Bidding I*	14.95	21.20
PHA	2000 Dual Cassette Cable	14.95	11.75	Entertainment			
PHA	2010 Monitor Cable	19.95	16.00	Command Modules			
PHA	2020 Audio Adapter (Headphone Jack)	19.95	16.00	PHM	3009 Football	29.95	25.45
PHA	1950 Thermal Paper (2 Pack)	9.95	8.00	PHM	3018 Video Games I	29.95	25.45
PHA	2605 Blank Overlays (4 Pack)	7.95	6.50	PHM	3023 Hunt the Wumpus	24.94	21.20
APPLICATION PROGRAMS							
Home Management/Personal Finance							
Command Modules							
PHM	3006 Home Financial Decisions	29.95	25.45	PHM	3024 Indoor Soccer	29.95	25.45
PHM	3007 Household Budget Management	39.95	33.95	PHM	3025 Mind Challengers	24.95	21.20
PHM	3012 Securities Analysis	54.95	46.70	PHM	3030 A-Maze-Ing	24.95	21.20
PHM	3013 Personal Record Keeping	49.95	42.45	PHM	3031 The Attack††	39.95	33.95
PHM	3016 Tax/Investment Record Keeping	69.95	59.45	PHM	3032 Blasto*†††	24.95	21.20
PHM	3022 Personal Real Estate	69.95	59.45	PHM	3033 Blackjack and Poker	24.95	21.20
Diskette							
PHD	5001 Mailing List	69.95	59.45	PHM	3034 Hustle*†††	24.95	21.20
PHD	5003 Personal Financial Aids	19.95	16.95	PHM	3036 Zero Zap†††	19.95	16.95
PHD	5021 Checkbook Manager*	19.95	16.95	PHM	3037 Hangman	19.95	16.95
PHD	5022 Business Aids Library-Finance Management*	39.95	33.95	PHM	3038 Connect Four†††	19.95	16.95
PHD	5024 Inventory Management*	69.95	59.45	PHM	3039 Yahtzee†††	24.95	21.20
Cassette							
PHT	6003 Personal Financial Aids	14.95	12.70	Diskette			
Education/Personal Enrichment							
Command Modules							
PHM	3002 Early Learning Fun	29.95	25.45	PHD	5002 TI-Trek (w/speech)	14.95	12.70
PHM	3003 Beginning Grammar	29.95	25.45	PHD	5010 Mystery Melody	14.95	12.70
PHM	3004 Number Magic	19.95	16.95	PHD	5015 Oldies But Goodies-Games I	19.95	16.95
PHM	3005 Video Graphs	19.95	16.95	PHD	5017 Oldies But Goodies-Games II	24.95	21.20
PHM	3008 Video Chess	69.96	59.45	Cassette			
PHM	3010 Physical Fitness	29.95	25.45	PHT	6010 Mystery Melody	9.95	8.50
PHM	3015 Early Reading††	54.95	46.70	PHT	6015 Oldies But Goodies-Games I	14.95	12.70
PHM	3020 Music Maker	39.95	33.95	PHT	6017 Oldies But Goodies-Games II	19.95	16.95
PHM	3021 Weight Control and Nutrition*	59.95	50.95	OTHER APPLICATION PROGRAMS			
PHM	3027 Addition and Subtraction I*††	39.95	33.95	Command Modules			
PHM	3028 Addition and Subtraction II*††	39.95	33.95	PHM	3000 Diagnostic	29.95	25.45
PHM	3029 Multiplication I*††	39.95	33.95	PHM	3001 Demonstration	69.96	59.45
Diskette							
PHD	5007 Teach Yourself Basic	34.95	29.70	PHM	3011 Speech Editor	44.95	38.20
PHD	5009 Music Skills Trainer	29.95	25.45	PHM	3014 Statistics	44.95	38.20
PHD	5011 Computer Music Box	19.95	16.95	PHM	3017 Terminal Emulator I	44.95	38.20
PHD	5018 Market Simulation	19.95	16.95	PHM	3026 Extended Basic*	99.95	84.95
Diskette							
PHD	5004 Programming Aids I	14.95	12.70	PHM	3035 Terminal Emulator II*	49.95	42.45
PHD	5005 Programming Aids II	24.95	21.20	Diskette			
PHD	5006 Math Routine Library	29.95	25.45	PHD	5004 Programming Aids I	14.95	12.70
PHD	5008 Electrical Engineering Library*	29.95	25.45	PHD	5005 Programming Aids II	24.95	21.20
PHD	5012 Programming Aids III*	19.95	16.95	PHD	5006 Math Routine Library	29.95	25.45
PHD	5013 Graphing Package	19.95	16.95	PHD	5008 Electrical Engineering Library*	29.95	25.45
PHD	5016 Structural Engineering Library*	29.95	25.45	PHD	5012 Programming Aids III*	19.95	16.95
Cassette							
PHT	6004 Programming Aids I	9.95	8.50	PHD	5013 Graphing Package	19.95	16.95
PHT	6006 Math Routine Library	24.95	21.20	PHD	5016 Structural Engineering Library*	29.95	25.45
PHT	6008 Electrical Engineering Library*	24.95	21.20	Cassette			
PHT	6013 Graphing Package	14.95	12.70	PHT	6004 Programming Aids I	9.95	8.50
PHT	6016 Structural Engineering Library*	24.95	21.20	PHT	6006 Math Routine Library	24.95	21.20

†† Developed by Scott Foresman

††† The Attack, Blasto, Hustle, Zero Zap, Connect Four and Yahtzee are trademarks of Milton Bradley.



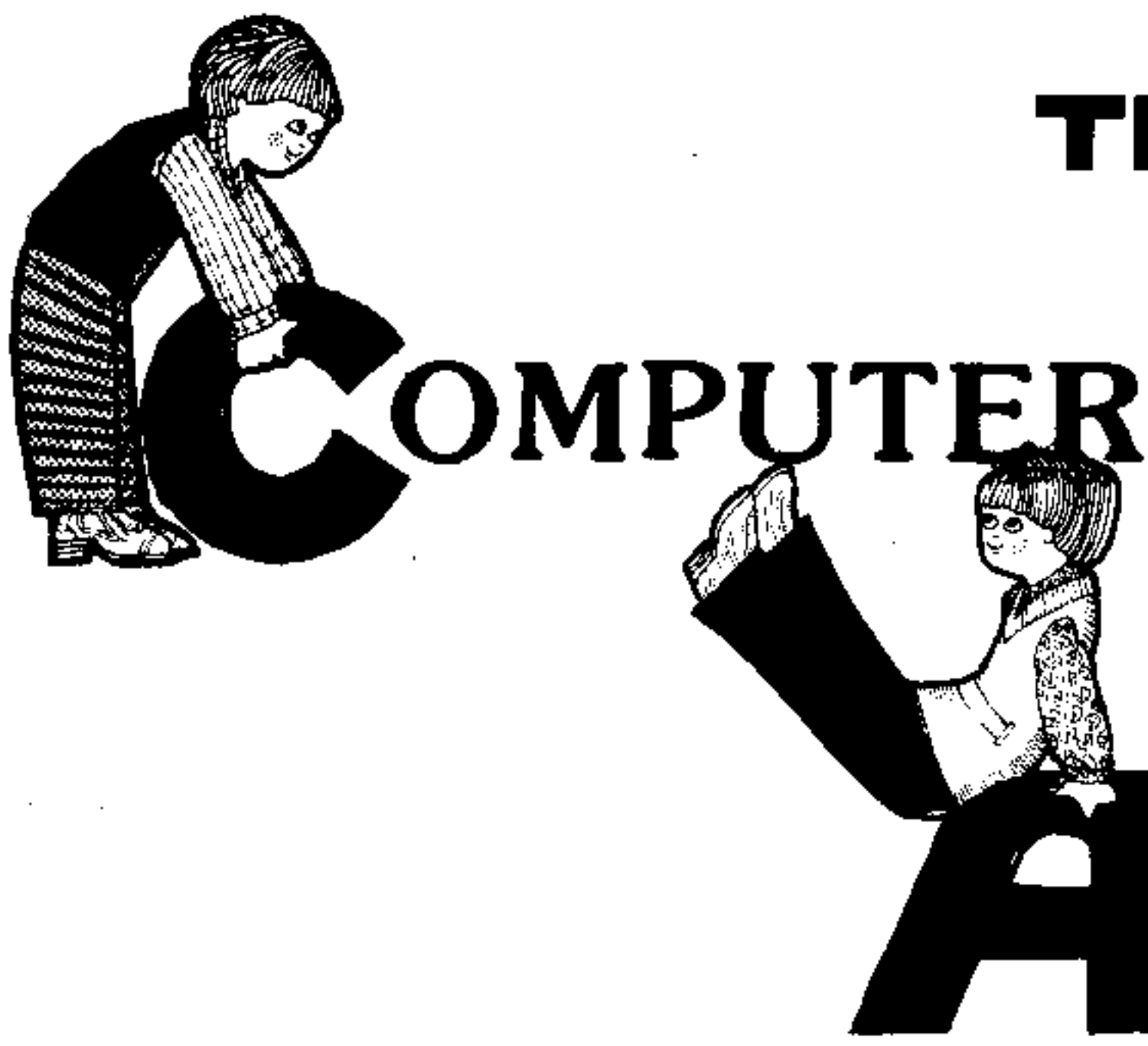
ELEK-TEK, inc.

5344 West Devon Avenue, Chicago IL 60646
(800) 621-1269 (312) 631-7800

CALL TOLL FREE 800-621-1269 (EXCEPT Illinois, Alaska, Hawaii)

Accessories discounted too. **Corporate Accounts Invited.** Mastercharge or Visa by mail or phone. Mail Cash, Ck., Mon. Drd., Pers. Ck. [2 wks to clear]. Add \$4.00 1st item; \$1.00 ea. add'l shpg. & handl. Shipments to IL address add 7% tax. Prices subject to change. **WRITE for free catalog. ALL ELEK-TEK MERCHANDISE IS BRAND NEW, FIRST QUALITY AND COMPLETE.**

THE STATE OF THE ART:



O.K. Class.
It's time for a little quiz:

INSTRUCTION

By Gary M. Kaplan

WHAT ARE THE CHARACTERISTICS OF GOOD EDUCATIONAL SOFTWARE?

- A. IT SHOULD BE EASY TO USE.
- B. IT SHOULD BE SUITABLE FOR THE LEVEL OF ITS INTENDED USERS.
- C. IT SHOULD CONTRIBUTE SIGNIFICANTLY TO THE LEARNING PROCESS.
- D. IT SHOULD HOLD THE INTEREST OF ITS USERS.
- E. IT SHOULD TAKE MAXIMUM ADVANTAGE OF THE COMPUTER'S APPROPRIATE CAPABILITIES WITHOUT DISTRACTING THE USERS.
- F. IT SHOULD PROVIDE ENOUGH PERCEIVED VALUE TO JUSTIFY THE USE OF THE HARDWARE NEEDED TO RUN IT.
- G. ALL OF THE ABOVE.

FRAME 1.

WHO IS SCOTT FORESMAN?

- A. THE LONE RANGER'S REAL NAME.
- B. THE HOLLYWOOD FILM PRODUCER WHO RECENTLY RE-MADE THE GREAT SCREEN CLASSIC, "BEDTIME FOR BONZO."
- C. THE POLAR EXPLORER WHO INVENTED THE ESKIMO BAR.
- D. ONE OF THE LARGEST AND OLDEST PUBLISHERS OF TEXT BOOKS USED IN PRIMARY AND SECONDARY SCHOOLS.

FRAME 2.

WHAT DO FRAMES NUMBER 1 AND 2 HAVE IN COMMON?

- A. THE MULTIPLE CHOICE QUESTION STRUCTURE FOR BOTH FRAMES IS CHARACTERISTIC OF POOR EDUCATIONAL USE OF THE COMPUTER.
- B. THE RELATIONSHIP BETWEEN SCOTT FORESMAN AND THE SUBJECT MATTER OF FRAME 1 IS WHAT THIS ARTICLE IS ALL ABOUT.
- C. BOTH OF THE ABOVE.

FRAME 3.

Although microcomputers have been used in schools and in the home for the last few years, teaching simple mathematics by computer-assisted instruction (CAI) hasn't really progressed much beyond the first primitive drill and practice software—programs that simply substituted a TV screen and typewriter-like keyboard for the more traditional pencil and paper. CAI software development just hasn't kept up with the improvements in hardware that have occurred during this same period. In fact, it's very rare these days to see a piece of tutorial software that utilizes *most* of a particular computer's capabilities. And yet that's exactly what a software developer *must* do to give his program the capability to hold a child's attention, be easy and fun to use repeatedly, and help the child to actually learn. All the "bells and whistles" that can be effectively used should indeed be used.

But producing a software series that meets with these and the remaining standards set forth in FRAME 1 requires close cooperation between programmers and educators. It also takes a lot of time. You don't attempt an ambitious project like this without strong belief in the virtues of a particular computer and an unwavering faith in the appeal of that computer system to your intended audience. You should also be prepared to stick it out for the "long haul." The payoff just won't occur overnight . . .

This brings us up to FRAME 2 in our little quiz and allows me to introduce Scott, Foresman and Company (SFC)—one of the players in this software development game. With a wealth of educational materials and experience to draw from, the Electronic Publishing Division of SFC has teamed up with Texas Instruments to produce Command Module software for the TI-99/4(A) computer.

And what a formidable alliance this is turning out to be: Take one 16-bit home computer—the only microcomputer that can employ color graphics, animation, sound effects, music and speech in the *same* program—and add the extensive knowledge resources of one of the biggest names in school text books; then cap it off with the aid and support of the renowned Texas Instruments Learning Center [its "Learning Factors Engineering" discipline is the subject of a forthcoming article], allow enough time to progress along the unavoidable "learning curve," and you have the basis for the best reading and math software in the industry.

It's unfortunate, but far too many educational software producers act as if they are exempt from the learning and improvement process, expressed graphically as the aforementioned learning curve; they release package after package in an attempt to quickly get out an entire series of programs ahead of the competition. With this approach there's no time to evaluate the effectiveness of the early releases in an attempt to incorporate design improvements into later ones. Sure, much of this software is often "maintained" (i.e., bugs that pop out of the woodwork are removed), but design changes are practically non-existent. The main attribute of cassette and disk-based software—the fact that it's "soft" and can be modified on the *same* magnetic media—has been terribly under-utilized.

The producer of software in Command Module format doesn't have this luxury. These plastic cases are filled with GROMs (Graphic Read-Only Memory) and ROMs—types of semiconductor devices that contain programs that are "burned in" and cannot be altered. It's perhaps more descriptive to think of these as "firmware"—occupying a place somewhere between hardware and software. Although this medium allows programming in languages other than BASIC (most typically GPL—an optimized Graphics Programming Language specifically for the TI-99/4), makes it possible to get at more powerful machine features, and effectively adds to the console's self-contained memory, it does require more thorough and time-consuming testing and debugging to prevent costly errors: Once in ROM or GROM, the only way to change the software is to throw all the chips away and "burn in" new ones—a very expensive proposition.

Besides Texas Instruments, other manufactures of color computers (Atari and Tandy) have weighed the advantages of software in cartridge form and have built machines capable of using "ROM Packs." But regardless who the software developer is, or the machine the software is written for, the fact that educational software is produced in this instant-loading, plug-in format *doesn't* necessarily make it *good* software. It may indeed be "bug free," have impressive color graphic displays, and contain music and sound effects, but if it doesn't meet *all* the criteria set forth previously (in FRAME 1), it just won't be very effective software—software that can justify the use of a microcomputer rather than a cheaper and more accessible pencil and workbook.

Although other personal computers can accept software in cartridge form similar to the push-in/pull-out method implemented on the TI-99/4(A) by Texas Instruments, that's where the similarity ends: TI's Command Modules have a far greater memory capacity (30K vs. 12K), and their interface with the console is better "human engineered"—allowing users to get in and out of the firmware with a single keystroke, without restricting the use of a peripheral port. [Don't forget that TI was the pioneer in the field of Solid State Software (TM), starting way back with their programmable calculators. We'll cover the technical differences between cartridge-loading computers in a forthcoming issue—Ed].

As a result of its abundant built-in capabilities and advanced design, the TI computer *offers* more to, as well as *demand*s more from programmers and educators if they hope to *fully* exploit the hardware, and transform the TI-99/4(A) into a "learning machine" without equal.

Those of us with school age children are indeed fortunate that so prestigious an educational publisher as Scott, Foresman and Company has joined forces with Texas Instruments for just this purpose. Their first product released, **Early Reading**, was revolutionary. It was in the first batch of Command Modules distributed by Texas Instruments, and so has been around for some time now. Those of you who have had the opportunity to use it with young children can, I'm sure, attest to the fact that it *really worked*—it taught your child to read and was extremely enjoyable to use. Here, for the first time, was a piece of educational software that employed synthetic speech, color animation, music, and sound effects in the *same* program—a learning activity that a child could

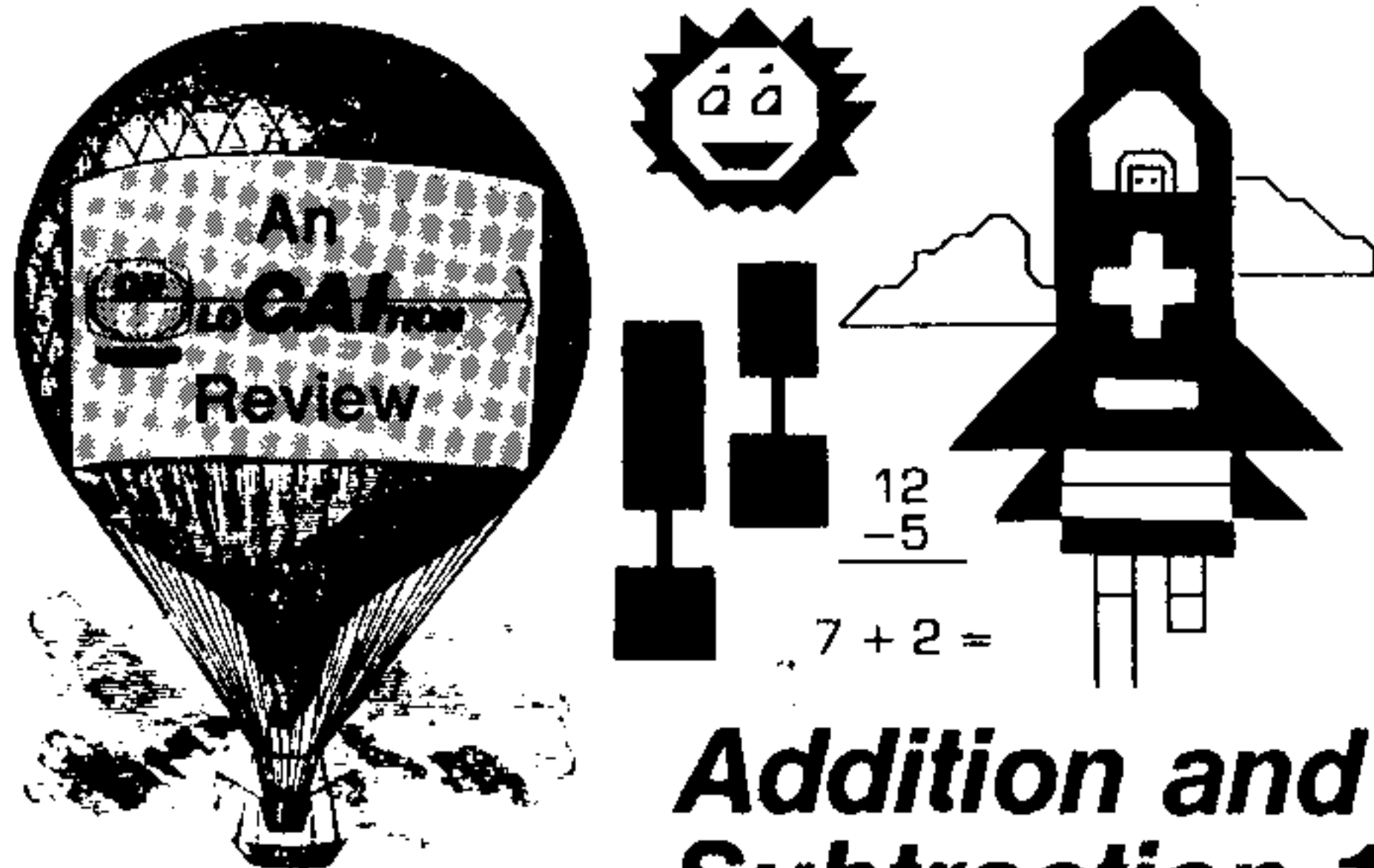
participate in *without* constant adult supervision. This software is indeed "worth its weight in GROMS . . ."

Although the **Early Reading Command Module** was clearly far superior to any CAI software I'd seen on any microcomputer at that time, it did contain some minor rough spots—in particular, a little too much use of *instructional* speech, and not an effective enough user-operable control system for getting into and out of certain "pages" of the activities. At that time, I thought it would indeed be interesting to see what improvements Scott Foresman and Texas Instruments would come up with in later reading and math releases.

Addition and Subtraction 1 wasn't ready for distribution until nearly one year after I had first seen **Early Reading**. And yes, there was quite an improvement in the software design. The two troublesome areas were, in fact, gone. It was evident that SFC and TI had made some impressive progress along the learning curve during the interim.

In a recent visit to the Chicago area, I had an opportunity to observe some of the development process that goes into this software. Situated within Scott, Foresman's strikingly beautiful Glenview complex is the Electronic Publishing Division—the division that was organized for the purpose of propelling one of America's most *traditional* educational publishers into the world of the future. This division is not only responsible for *educational* software, but is equally responsible for *administrative* software—a program series (designed to run on a TI-99/4(A) disk system with a TI 825 printer and Command Module driver programs) that automates administrative work and thus helps run the schools more efficiently: The computer keeps attendance records and inventories, does scheduling, salary planning and mailings, maintains personnel and student data, and assists in accounting and payroll.

Software development is done on large 990-series Texas Instruments computers by a team of programmers proficient in GPL, Assembly, and BASIC. Development programs are shuttled back and forth across the country between test sites. When a program finally gets as refined as its going to get (usually after many round trips), a TI facility will make the "mask," burn in the final GROM chips, and then assemble them into Command Module cases. The process is slow, but unusually thorough, and the result—as evidenced by the fine **Addition and Subtraction 1** software product—"speaks" for itself (pun intended) . . . [Watch the next issue for an article explaining this software development process in more detail.]

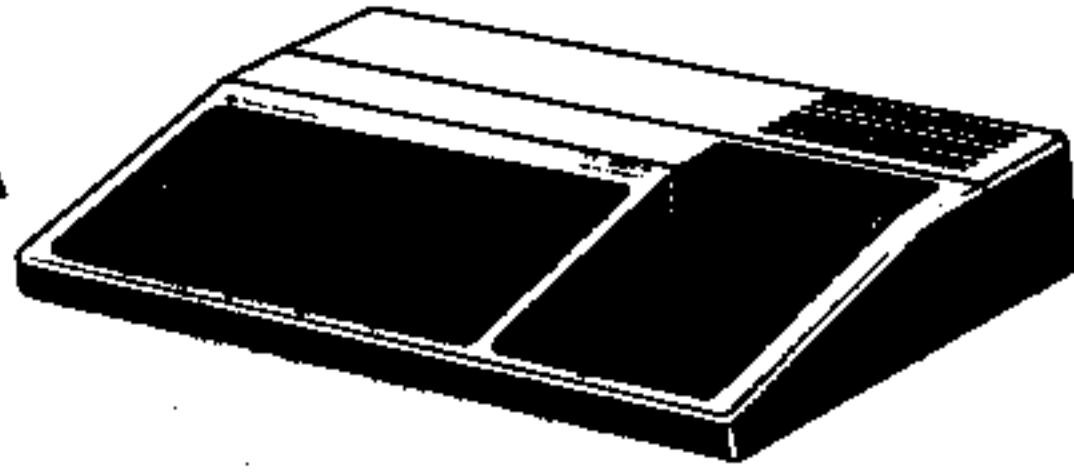


Now let's take a look at the software. The program architecture appears simple enough: The module begins by teaching the basic concept of counting from zero to nine. Then, step-by-step, the principles of addition and subtraction (using this same number group) are introduced. A child watches and listens (with the optional speech synthesizer attached to the TI-99/4(A) console) to tutorial problems at the beginning of each of the separate activities. The program then gives the child a choice of more example problems or of moving on to the practice drill. If the child answers 80 percent of the drill



SAVE MONEY ON TEXAS INSTRUMENTS PERSONAL COMPUTERS

**TI-99/4A
\$379**



DISK DRIVE CONTROLLER	\$229
DISK MEMORY DRIVE	\$369
SOLID STATE PRINTER	\$239
32K MEMORY EXPANSION	\$239
R.F. MODULATOR	\$43
TELEPHONE MODEM	\$169
RS 232 ACC. INTERFACE	\$169
CONTROLLERS (PAIR)	\$31
USER REFERENCE MANUAL	\$9

SOFTWARE

PHM 3007 Household Budget Management	35.00
PHM 3016 Tax/Investment Record Keeping	60.00
PHD 5003 Personal Financial Aids	18.00
PHD 5022 Business Aids Library	34.00
Finance Management	
PHD 5024 Inventory Management	60.00
PHM 3005 Video Graphs	18.00
PHM 3020 Music Maker	35.00
PHM 3021 Weight Control and Nutrition	53.00
PHM 3023 Hunt The Wumpus	22.00
PHM 3030 A-Maze-Ing	22.00
PHM 3011 Speech Editor	39.00
PHM 3014 Statistics	39.00
PHD 5004 Programming Aids I	13.00
PHD 5005 Programming Aids II	22.00
PHD 5008 Electrical Engineering Library	26.00
PHD 5012 Programming Aids III	18.00
PHD 5013 Graphing Package	18.00
PHD 5016 Structural Engineering Library	26.00
PHT 6004 Programming Aids I	9.00
PHT 6008 Electrical Engineering Library	23.00
PHT 6013 Graphing Package	13.00
PHT 6016 Structural Engineering Library	23.00

CALL TOLL FREE

(800) 233-8950

IN PA, CALL (717) 327-9575

COMPUTER MAIL ORDER

501 E. THIRD ST., WILLIAMSPORT, PA 17701
OVER 40 YEARS EXPERIENCE IN SOPHISTICATED ELECTRONICS

Phone orders invited (800 number is for order desk only). Or send check or money order and receive free shipping. Pennsylvania residents add 6% sales tax. Add 3% for Visa or M.C. Equipment is subject to price change and availability without notice. Please call between 11 AM & 6 PM.



problems correctly, he or she automatically advances to the next activity; if less than 60%, the program returns to the appropriate previous activity for more practice.

The nine separate activities are presented in order of increasing difficulty—each building upon the skills learned in the previous one. The first two activities start with the concept of counting and recognizing quantities. Then, in the third and fourth activities, addition is taught through the concept of “bringing together,” and the child is taken from a graphic representation of horizontal addition into the realm of vertical addition. Activities 5 and 6 teach and drill subtraction in a similar way—with the “taking away” concept and a progression from the horizontal to vertical. When the child reaches Activity 7, the module creates a box that is divided into 9 sections for testing addition and subtraction skills in both horizontal and vertical formats. The final two activities are structured in the form of a game to reinforce the relationship between numbers that were learned in the previous activities: First an addition table, and then a subtraction table are displayed, and the child is challenged to fill in the intersections of rows and columns with the correct answers.

The design of this CAI software is certainly straightforward—nothing revolutionary (apart from the speech) in teaching basic math skills. Other CAI math programs use this or a similar approach. But the polished way in which the design is implemented, however, sets this software far apart from the rest of the pack. **Addition and Subtraction 1** scores very impressively when rated against the standards set forth in Frame 1 at the beginning of this article.

Ease of Use & Suitability

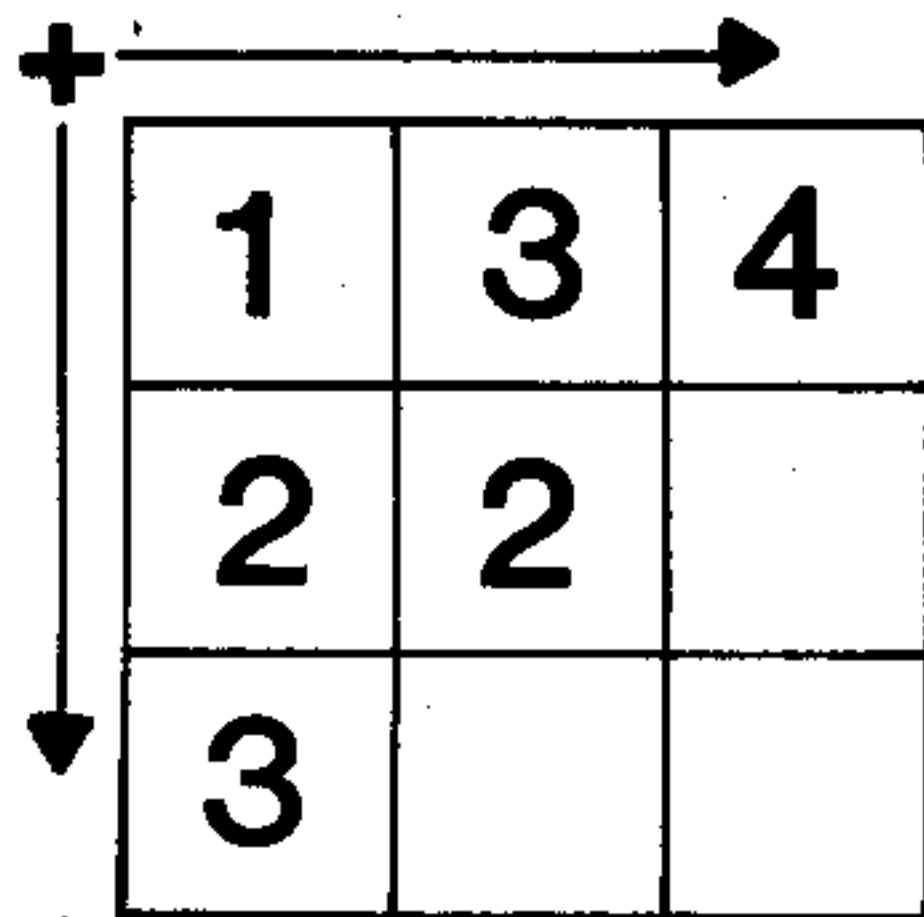
CAI software has been traditionally weak in its reliance on instructions and operating procedures that, unfortunately, have been designed for a much higher level of understanding than the intended users actually possess. The developers of **Addition and Subtraction 1** must have been well aware of this pitfall, because the module is so easy for a child to operate. Being able to read, or having an adult constantly standing alongside is *not* a requirement. Of course, the fact that the program resides in an “instant loading” Command Module and employs synthetic speech is a big plus in its favor, but these features on their own just wouldn’t be enough. It’s the numerous, unsung “hidden” features that have been “human engineered” into the software that make a world of difference. Take, for example, the operation of the control keys: The child doesn’t have to be able to read the nine-item activity menu or remember what is involved in each. Instead, if the period key (.) is pressed, the module goes into “overview mode” and actually shows brief, speeded-up demonstrations (sequentially from 1 to 9) of what each activity involves. When the desired activity is reached, the child releases the key and the tutorial exercise for that activity begins.

In a similar way, a few other keys are used to put the child in control of the speed and direction he or she wishes to travel with the module’s activities. For example, ENTER eliminates pauses (by deactivating the built-in response-time monitoring); the SPACE BAR stops and starts any activity; W(BEGIN) returns to the activity menu; R(REDO) or Z (BACK) returns to the beginning of the activity in progress, and A(AID) returns to the beginning of the *section* of the activity in progress. In all cases, the SHIFT or FUNCTION key is not required. All it takes is a single keystroke (with a built-in one second delay to guard against the child pressing the key accidentally).

Users of this module will find a conspicuous absence of text instructions that have been traditionally imbedded within CAI software frames. The developers have substituted animated displays of what is expected of the users—demonstration tutorials in a graphic language that children can easily understand. Any child who has ever seen *Sesame Street* or *Electric Company* will feel right at home with this software.

As I expected, the module is indeed “crash proof”—the

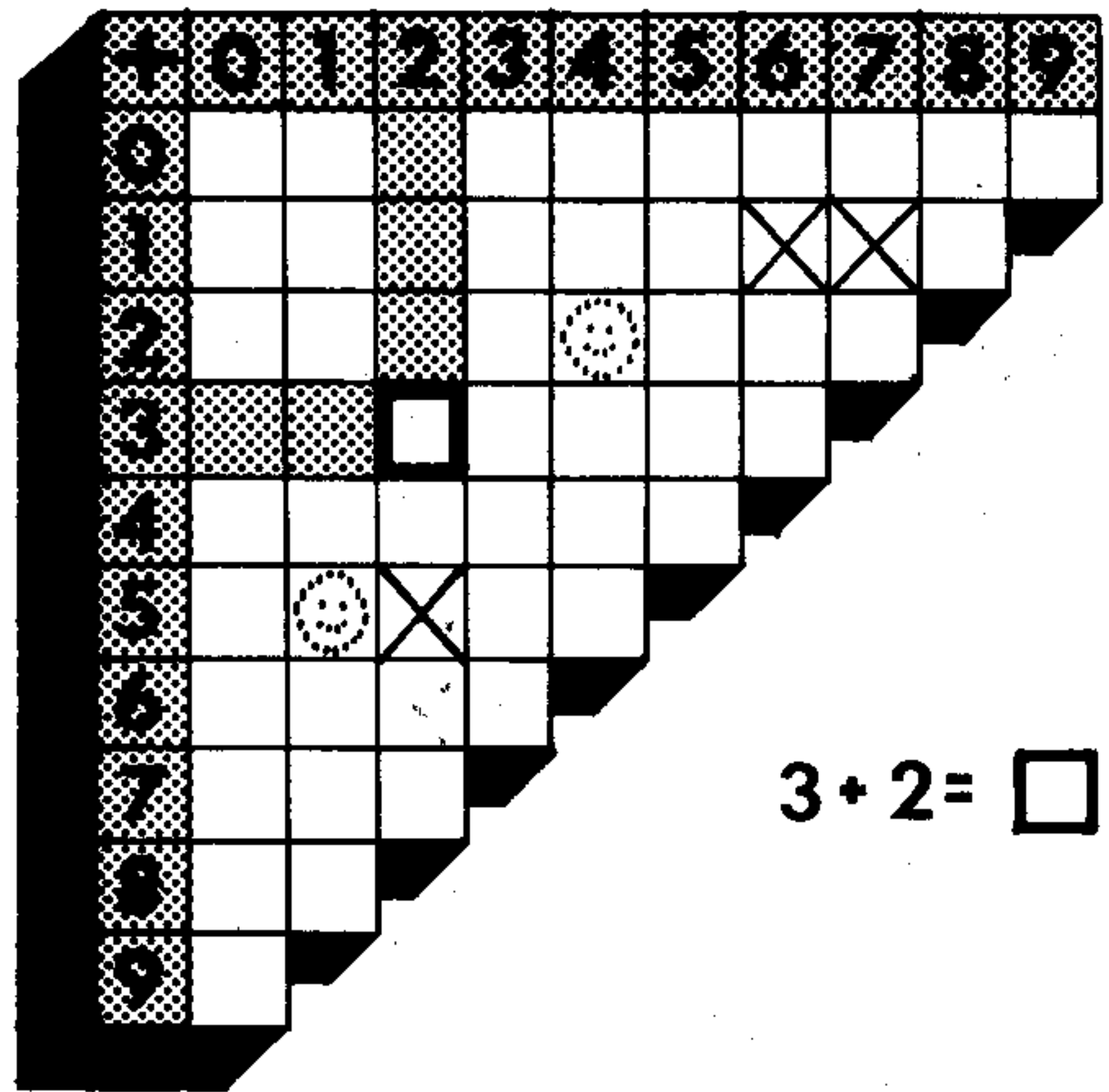
only way to halt the program is with a SHIFT Q or FUNCTION Q. This is a necessity in a class environment where the unexpected always seems to happen. Another big plus in a school (or home, for that matter) environment is the module's *adaptability*: The activities adapt to the user, rather than vice versa—letting children of different abilities proceed along at their own comfortable pace. The feedback each child receives is useful, immediate, and frequent. The error signals were designed to be low-key and nonintimidating—a red “X” momentarily appears and an “uh-oh” sound is heard; the incorrect response and red “X” are then erased, and the computer’s synthetic voice tells the child to “Try again.” If the second response is correct, the child gets one of the three visual and musical rewards (not too long to be repetitively boring); if incorrect, another red “X” appears and the computer works through the problem step-by-step, showing how to arrive at the answer. In Activities 8 & 9 (the module’s review activities) the switch to this explanatory mode is made after only *one* incorrect response. Then, following the completion of these activities, a list of up to 32 incorrectly answered problems appears with the message “Study these for next time!”



Motivation & Contribution to Learning

Right from the opening music, the module grabs the attention of its users—children or curious adults—and keeps them involved. *Involvement* is the key word here; there’s always something happening to provide sensory input. The program keeps “showing and telling” without freezing up—regardless of how unresponsive a timid child first exposed to the computer is bound to be. It does this by monitoring response time and repeating or continuously branching at a pre-determined pace unless overridden by a child pressing one of the keys. In this way, young users can participate in “Sesame Street Mode” until they perceive the computer as a friendly device and are both willing and able to interact with it. As protection for the TV, the screen goes blank after ten minutes of inactivity, but the speech does continue. In my opinion it would have been a nice feature at this point if the developers had included the “spoken” message: “Please press any key to help me get my picture back.”

The use of color animation, music, and sound effects provide effective rewards for answering correctly and advancing through the activities, as well as keeping a user’s attention focused on the screen during a tutorial sequence. This is especially important after a problem is missed, and the computer demonstrates how the correct answer was reached. Although I found all these motivating elements entirely satisfactory, I do believe that *passive* motivation such as this is not in keeping with the other outstanding design elements in the software. Perhaps more *active* motivation could be achieved by incorporating the user’s name as part of a display. And going one step beyond that, *interactive* motivation could be built in by providing users with a graphic game or art exercise as a reward for a certain level of achievement. Without being a part of the actual programming team, however, it’s difficult to say whether these additions would have been economically feasible: The additional memory overhead might have required an extra chip in the Command Module which could have made the price too expensive for the



software’s intended market.

The greatest strength of this software is that it contributes significantly to the learning process. This program is definitely *not* just a “page turner” like so much of the CAI courseware presently available for other microcomputers. The amount of material presented in each segment or “frame” is just large enough to provide diversity without frustration, and just small enough to ensure a high success ratio without boredom. The sequence of material also appears to be optimized. All of this demonstrates to me a great amount of “fine tuning” and user testing in the development process.

Computer Utilization & Hardware Justification

At the risk of being repetitive, I must say that this mathematics software *fully* utilizes the impressive array of capabilities built into the Texas Instruments computer and its speech peripheral. When evaluating a piece of software in regard to how well it makes use of a computer’s features, it’s important to ask yourself how well that same learning process could be duplicated with manual methods if the computer weren’t available. In the case of **Addition and Subtraction 1**, the process *couldn’t be duplicated* by static means alone. This is a good indication that the module’s use of interactive video is indeed revolutionary in the world of personalized CAI.

Once again, I offer a suggestion (that may or may not be economically feasible in the context of production and marketing): A printed record of a child’s performance and progress would be nice for adults to have. Accordingly, a print option for the TI thermal printer could possibly be built in.

Finally, to be complete in any educational software evaluation, we should look at what *hardware* it takes to run the software. Here is where the Scott-Foresman package really shines, since all that is required is the basic TI-99/4(A) console—in contrast to other computer systems that frequently require disk drives and expansion memory boards to run anything meaningful. The combination of low-cost TI hardware and Scott, Foresman software certainly appears to be setting the standards for the practical CAI environment that educators and concerned parents will be seeking during the decade of the 1980s.

NEED CASSETTE TAPES ?

For High Quality & Low Cost
See Inside Back Cover (page 95)

Moving with You into the '80s

Microcomputer Courseware From Scott, Foresman, the Education Expert

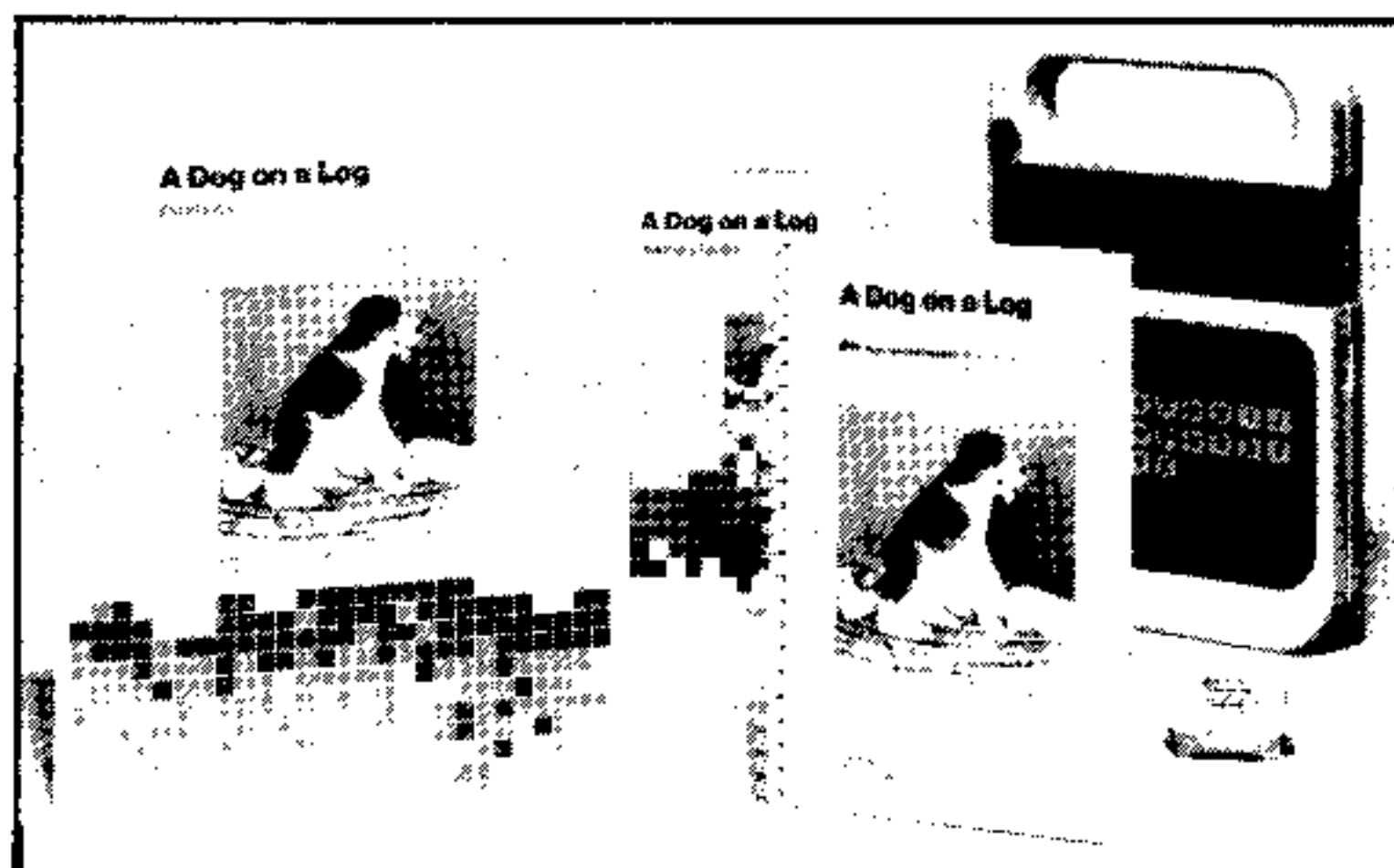


Reading Skills Courseware Series (Grades K-6) and Mathematics Courseware Series (Grades K-8) feature:

- content correlated with basal series—to supplement instruction, provide enrichment
- individualizing capabilities that meet each pupil's learning needs
- motivating animation, color graphics, music, and speech
- comprehensive Teacher's Guides

School Management Applications for administrators feature:

- fourteen modules, each programmed to streamline a specific area of school administrative work
- easy information storage and access, clearly organized printed reports
- program instructions in plain English, helpful Reference Manuals



Versatile microcomputer hardware systems are also available for both classroom and administrative use.

Scott, Foresman Module Packages (Grades 1-3) for Speak & Read™ Educational Model, complete with Pupil Books, Teacher's Guides, and optional Workbooks, can enhance any reading curriculum by reinforcing essential skills.

Scott, Foresman and Company
Electronic Publishing
 Glenview, Illinois 60025

Learn more. Mail this coupon to receive free information.

To Scott, Foresman Electronic Publishing

Please send me free, full-color brochures on:

- Reading Skills Courseware Series and Mathematics Courseware Series (30621-6)
- School Management Applications (30661-5)
- Scott, Foresman Module Packages for Speak & Read™ (30692-5)

NN10

School _____

Name _____

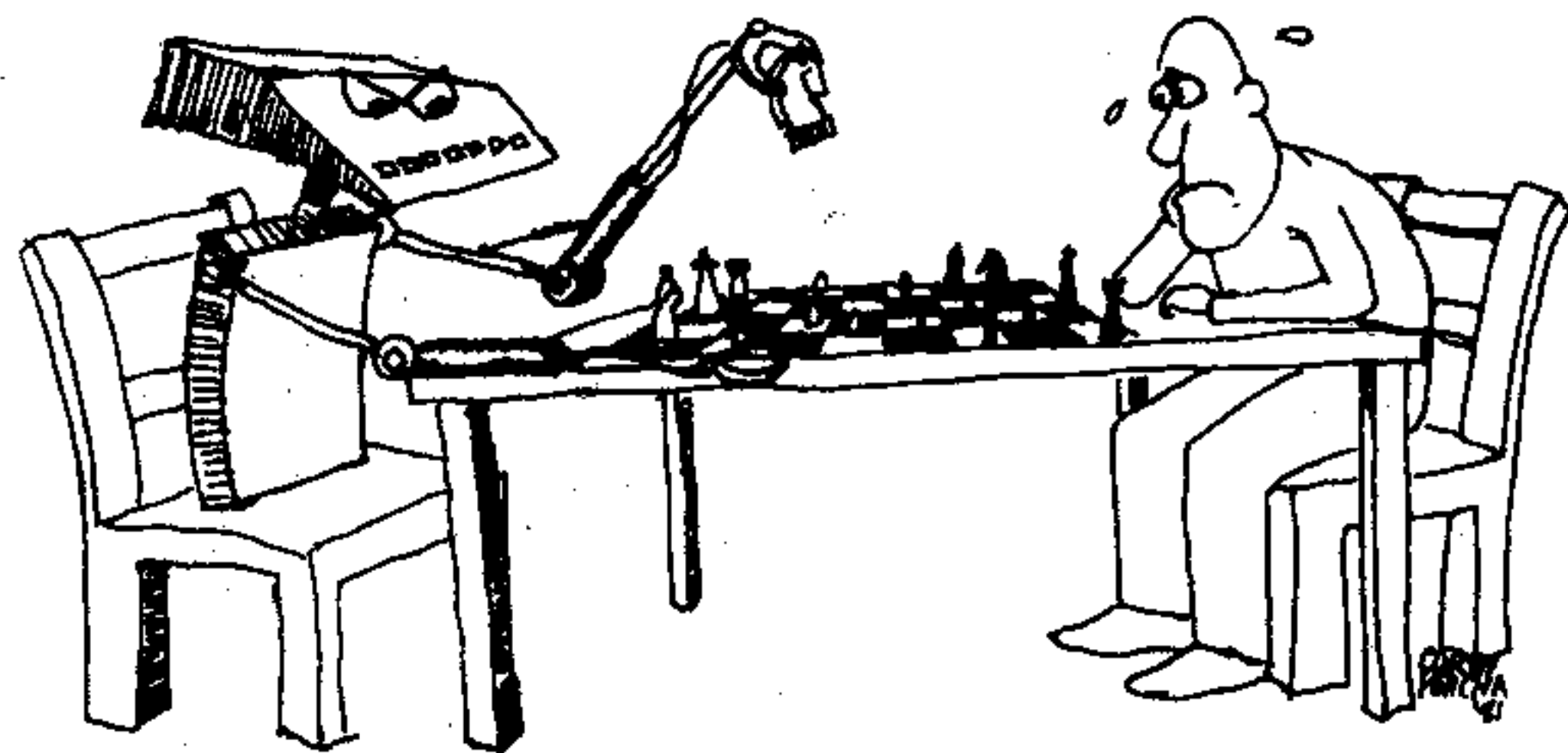
Position _____

School Address _____

City _____ State _____ ZIP Code _____

COMPUTER CHESS

CORNER



By Jerry Wolfe

Last time, we discussed the relationship between chess programs and artificial intelligence, and examined some general characteristics of chess playing programs—both strengths and weaknesses. In this article, I'm going to illustrate some of these characteristics through an actual game played between the *TI Video Chess* program and myself. The game was played with the program set on intermediate level, normal mode, with 200 seconds per move allowed.

White: J. Wolfe Black: TI-99/4 with
Video Chess

1. E2 - E4 G7 - G6
2. D2 - D4 F8 - G7

These first two moves constitute the Pirc-Robatsch defense to the opening move E2 - E4. You may have noticed in your play that the program often makes the first several moves quickly and then slows down. This is because certain standard opening sequences are stored in the program and played automatically in the appropriate situation. As soon as these run out or as soon as the position is no longer standard, the program reverts to its main programming and hence slows down.

3. B1 - C3 C7 - C6

The main purpose of the opening part of the game is to bring out the pieces and to get a reasonable foothold in the central part of the board. (More precisely, the "center" is the square region whose corners are C3, C6, F6, and F3. The squares D4, D5, E4, and E5 are

About the Author

Jerry Wolfe is a professor of mathematics at the University of Oregon in Eugene, Oregon. He has been playing chess since the age of eleven and began playing in chess tournaments at the age of fourteen. He is the 1979 Oregon Open champion and has won numerous other local tournaments in the Pacific Northwest during his chess "career." Currently he holds the official rating of candidate master.

especially crucial.) Long experience has shown that the success of future maneuvers depends on an adequate control of this area. The last moves for each side fit well into this plan. White brings out a knight that bears down on the center while black prepares to play D7 - D5 establishing his own foothold there.

4. F1 - C4 B7 - B5
5. C4 - B3

White develops a piece and temporarily prevents D7 - D5. Black responds by driving back white's bishop and preparing a later pawn advance on the "queen-side" (i.e., the left-hand portion of the board).

5. . . . D7 - D6 6. . . . D6 - D5

"Such programs (with look-ahead capability of nine moves) will be virtually impossible to trap . . . the human player will most likely be the victim."

This is a weak move because of the following tactic.

6. C3 - B5

Black cannot capture the knight because white then plays 7. B3 - D5 and captures the rook at A8 coming out with a two unit gain in material. (Recall that a rook is worth 5 units and a bishop 3 units. These units represent the relative strength of the two pieces. This is what is meant when we say a rook is worth more than a bishop.)

You might be wondering why the program missed such a short sequence of moves. Well, the reason is fairly complex. The program has two basic features: The first is a *static evaluation feature* which takes a given position and evaluates it to decide which side is better and by how much. This is done by

assigning numerical values to certain features of the position and summing these values to get a numerical value for the position. For example, being a pawn ahead in material might be worth, say, 75 points, while not being able to castle (ever) might be "worth" minus 15 points, etc. The program does this for both sides, and the side with the largest score is judged to have the best position. In this evaluation scheme, material advantage is given the largest positive weight by far.

The second basic feature of the program is a *searching procedure*. When combined with the static evaluation program, it allows the program to evaluate the consequences of various moves and to pick what it deduces to be the optimum one. Unfortunately, time and memory considerations limit the number of moves the program can look ahead (i.e., its "search horizon") and can also limit the number of moves that are considered in response to a contemplated move.

Thus, in examining the position after 5. . . . D7 - D6, white has 38 legal moves. In deciding which moves to consider first as possible replies by white, the program will *not* begin with moves that result in immediate material loss by white. This again is due to the heavy weight assigned to material superiority. Thus the continuation 6: C3 - B5 might not even be reached in the search within the time limit. Sacrifices of material are difficult for all but the most advanced and powerful programs to either make or predict.

This is a good move and is the other side of the argument above. The program finds the only possible way to regain the lost pawn. Here the emphasis on material is helpful to the program.

7. B5 - C3 D5 - E4
8. C3 - E4 G7 - D4
9. G1 - F3

Thus black has not lost a pawn after all. However, black's position now has two unpleasant features: First, his pawns at A7 and C6 are weakened since they cannot be protected by pawns if attacked, but must be protected by pieces. This can tie down black's pieces and will make the pawns vulnerable especially in the later part of the game when fewer pieces remain. Secondly, to regain the pawn, black has exposed his bishop to attack. Thus white can develop a piece

(G1 - F3) and at the same time force black to waste a move either guarding or retreating his bishop. Note that white has three pieces developed and no pawn weaknesses while black has only one developed and definite pawn weaknesses. White already has a distinct advantage.

9. ... C6 - C5

This is another weak move. Black cannot retreat the bishop to G7 or F6 because of the B3 - F7 check winning the queen, but D4-B6 was possible—preserving material equality. There is some evidence from this game and others I have played that the search horizon of Video Chess is about two moves in complicated positions. This would explain why C6 - C5 (so as not to waste a move retreating) was considered best.

10. C2 - C3 D4 - F2 check

Now looking ahead two moves, the program *apparently* can see that if D4 - G7, then B3 - F7 check and white wins the black queen on the next move. However, later when I set up the position after D4 - G7 and asked the program to play white, it played D1 - D8 winning only two pawns (the one on F7 and then the one on C5), leaving white two units ahead. Since giving up a bishop for a pawn also leaves black two units behind without having to trade queens, the move 10. ... D4 - F2 check was chosen. Thus it appears that

the program made the right move for the wrong reason!

11. E1 - F2 G8 - F6

Again the program does not see the third move in the coming sequence.

12. E4 - F6 check E7 - F6
13. D1 - D8 check E8 - D8
14. B3 - D5

Thus white wins another piece.

14. ... B8 - C6
15. D5 - C6 A8 - B8

White is so far ahead in material that winning is simple. Accordingly, I will relate the rest of the game with little comment.

16. B2 - B4

This move allows white to play C1 - F4 without an annoying check at B2 by the black rook.

16. ... C5 - B4
17. C1 - F4 B8 - B6
18. A1 - D1 check D1 - E2
19. F4 - D6 check E7 - D8

19. ... E7 - E6 leads to a quick mate after 20. H1 - E1 check E6 - F5; 21. D1 - D5 check F5 - G4; 22. H2 - H3 mate.

20. D6 - C5 check D8 - C7
21. C5 - B6 check A7 - B6
22. C6 - D5 B4 - C3
23. H1 - E1 C7 - B8
24. D5 - F7 C8 - G4
25. E1 - E7 G4 - F3
26. G2 - F3 F6 - F5
27. D1 - D7 B8 - C8
28. F7 - E6 B6 - B5
29. D7 - A7 check C8 - D8
30. E7 - D7 check D8 - E8
31. A7 - A8 checkmate

Currently, the most powerful chess programs can look ahead about six moves in fairly complicated positions. Advancements in hardware should extend that capability to *nine* moves. This is about twice as many moves as chess masters can look ahead in complicated positions. Such programs will be virtually impossible to trap in simple tactical sequences and, in fact, the human player will most likely be the victim. To defeat such a program will require superior application of chess theory and strategy, as well as avoidance of open tactical situations where an eight or nine move look-ahead program would be at its best.

SOLUTIONS TO THE PROBLEMS IN THE LAST ISSUE:

Problem No. 1:

1. H5 - H7 check!! G8 - H7
2. E4 - F6 Double check H7 - H6 (else E5 - G6 mate)
3. E5 - G4 check
4. F2 - F4 check
 - (a) 4. ... G5 - F4
 5. G2 - G3 check F4 - G5
 6. H2 - H4 check mate or
 5. ... F4 - F3
 6. 0 - 0 checkmate.
 - (b) 4. ... G5 - H4
 5. G2 - G3 check. H4 - H3
 6. D3 - F1 check B7 - G2
 7. G4 - F2 checkmate.

Problem No. 2:

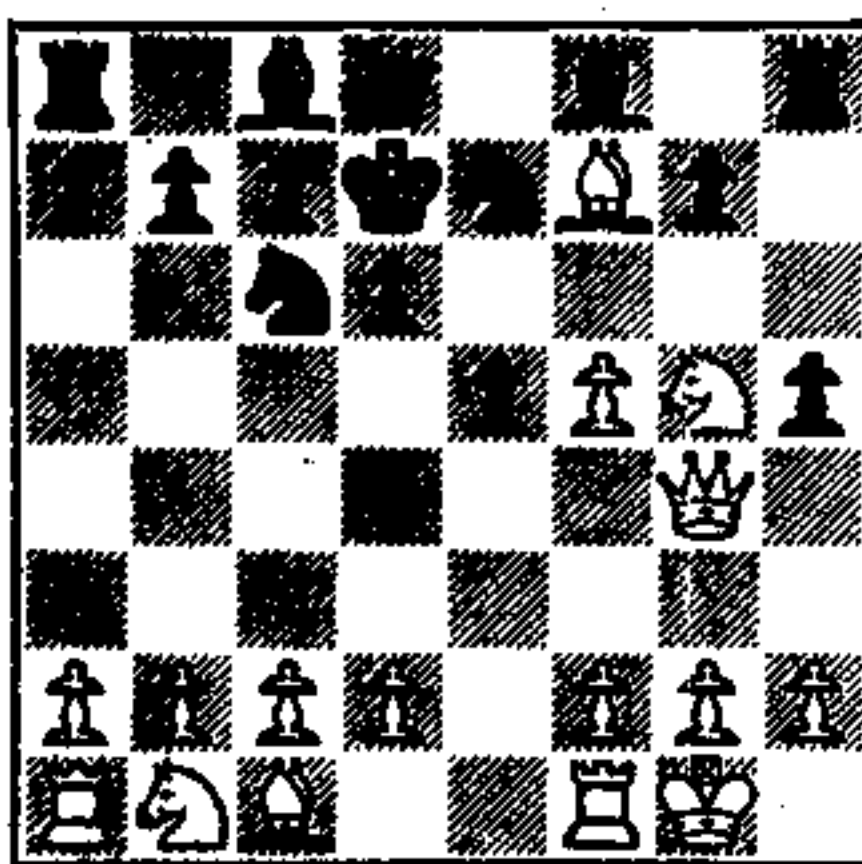
1. ... F7 - F1 check
2. E3 - G1 F1 - F3 check!
3. E4 - F3 C6 - F3 checkmate.

New Problems (The solutions will appear in our next issue.)

PROBLEM NO. 1

White: Pawns: A2, B2, C2, D2, F2, F5, G2, H2
Knights: G1, G5
Bishops: C1, F7
Rooks: A1, F1
Queen: G4
King: G1.

Black: Pawns: A7, B7, C7, D6, E5, G7, H5
Knights: C6, E7
Bishops: C8, F8
Rooks: A8, H8
Queen: D8
King: D7

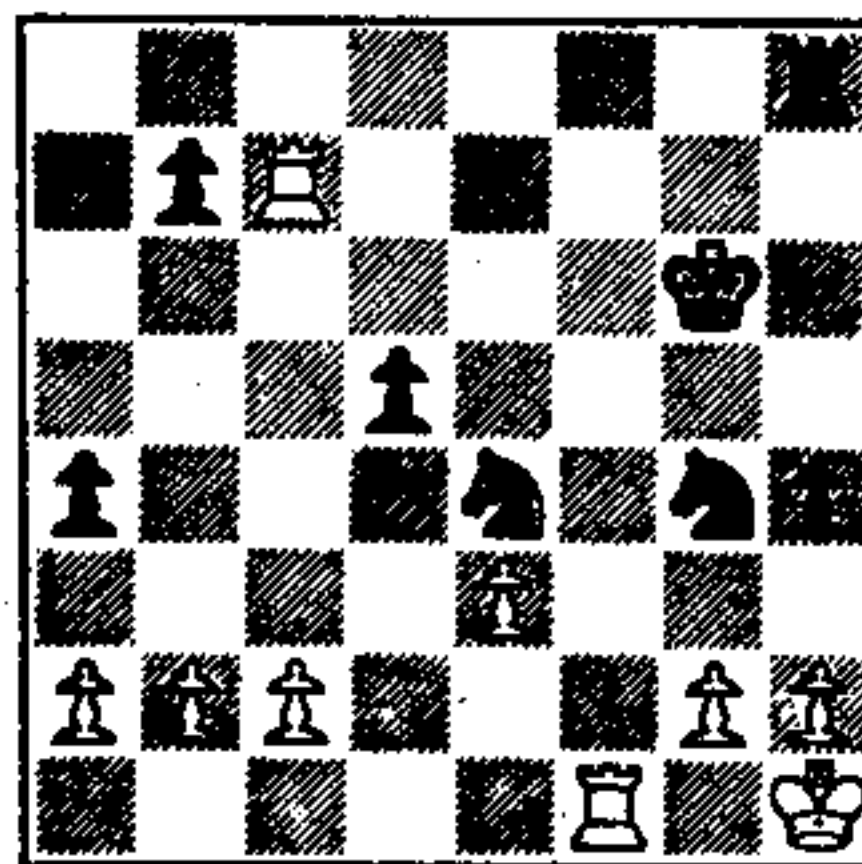


White to move and mate in at most three moves.

PROBLEM NO. 2

White: Pawns: A2, B2, C2, E3, G2, H2
Knights: NONE
Bishops: NONE
Rooks: F1, C7
Queen: NONE
King: H1

Black: Pawns: A4, B7, D5, H4
Knights: E4, G4
Bishops: NONE
Rooks: H8
Queen: NONE
King: G6



Black to move. There is no forced mate but black has a decisive sequence of moves available. Try and find it!

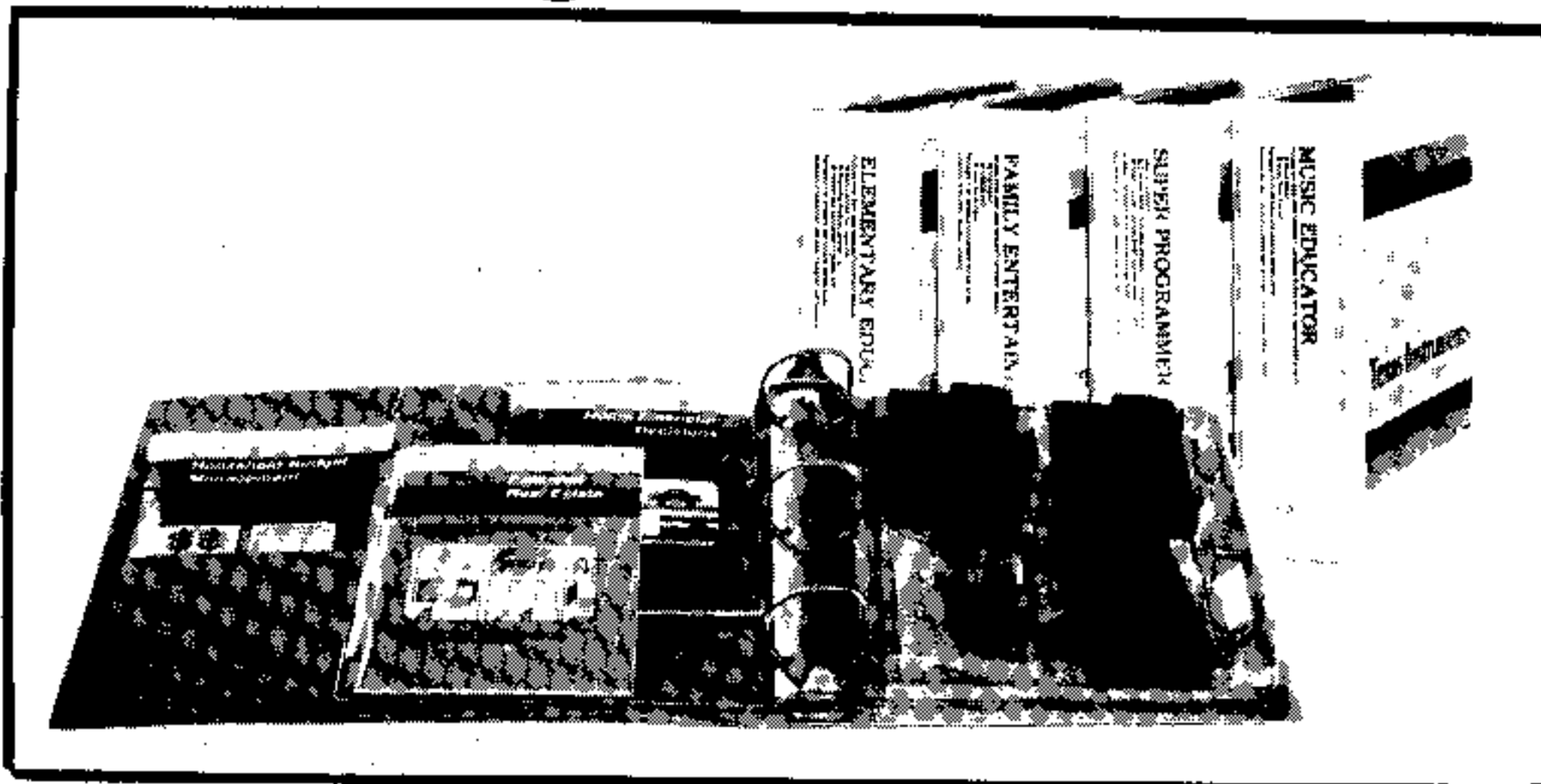
ATTENTION

**DEALERS,
DISTRIBUTORS,
& BOOKSTORES**

CALL OR WRITE US
ABOUT SELLING
SINGLE COPIES OF

99'er
magazine™

New! Specialized Software Assortment Packages from Texas Instruments.



A Money Saving Way To Get TI Home Computer Software.

Many of TI's most popular software offerings are being packaged and sold in the assortments listed below. Each software assortment package comes in a handy storage case consisting of a three-ring binder with special inserts to hold the command modules (or cassettes) and instruction booklets. The storage case alone is a \$15 value and it's FREE with the purchase of an assortment package. What's more, Unisource Electronics is offering you these packages on an introductory basis at EVEN GREATER SAVINGS. Just look!

HOME FINANCIAL MANAGER — Can help you gain better control of your family's financial planning, and can help guide decisions regarding your budget, investments, major purchases (such as a car or home), loans and savings. Three modules included are Home Financial Decisions, Household Budget Management and Personal Real Estate. (Note: A cassette tape recorder and interface cable are needed to maximize the record-keeping features of this package.)

Sug. Ret. of Modules\$139.85 **YOUR PRICE** **\$99.95**
 Storage Case 15.00 **A \$54.90 savings!**
 Total Value\$154.85

THE ELEMENTARY EDUCATOR — Can make your TI Home Computer an effective instructional tool. The activities teach and reinforce basic language or math skills and raise your child's computer-awareness level. Activities also feature music, sound effects and full-colored graphics. Three modules included are Early Learning Fun, Addition and Subtraction I and Beginning Grammar. Activities for preschool through 5th grade level.

Sug. Ret. of Modules\$ 99.85 **YOUR PRICE** **\$79.95**
 Storage Case 15.00 **A \$34.90 savings!**
 Total Value\$114.85

FAMILY ENTERTAINER — The whole family can enjoy limitless hours of fun, intrigue and challenge. Whether you're destroying aliens, racing through mazes eluding cats or hunting the Wumpus while avoiding slime pits, you're in for some action-packed entertainment. Three modules included are The Attack, A-Maze-Ing and Hunt the Wumpus.

Sug. Ret. of Modules\$ 89.85 **YOUR PRICE** **\$69.95**
 Storage Case 15.00 **A \$34.90 savings!**
 Total Value\$104.85

TI SUPER PROGRAMMER — Combines the elements you need to become proficient in TI Extended BASIC programming. Just insert the TI Extended BASIC module in your console and the Teach Yourself Extended BASIC cassette in a cassette player (not included) and the features of a high-level programming language are at your fingertips. Available late November.

Sug. Ret. of Software\$119.90 **YOUR PRICE** **\$99.95**
 Storage Case 15.00 **A \$34.95 savings!**
 Total Value\$134.90 Limited Availability.

MUSIC EDUCATOR — The more you learn about music the more you enjoy it. Whether you understand the intricacies of chords or learn to compose your own melody. TI's Music Educator brings you a bright new note in improving your musical skills. Includes Music Maker command module and Music Skills Trainer cassette-based program (cassette player not included).

Sug. Ret. of Software\$ 64.90 **YOUR PRICE** **\$54.95**
 Storage Case 15.00 **A \$24.95 savings!**
 Total Value\$ 79.90

And a special offering from Unisource ... a complete TI LOGO Add-on System for just \$489.00. There's no better way for your kids to learn the basics of programming a home computer. TI LOGO is a child-appropriate, computer language that lets students of all levels of ability communicate with the TI Home Computer using an easy-to-understand language. You get both the TI LOGO command module and Memory Expansion Unit (32K RAM) — a total suggested retail value of \$699.90 — at a savings of over \$200.00!

CALL TODAY!

The simplest way to place your order is to call us, TOLL FREE:

1-800-858-4580

In Texas, call 1-806-745-8835
 Phone lines open 8 a.m. - 6 p.m. CST.



UNISOURCE ELECTRONICS, INC.

Or, choose from the complete selection of TI software.

Package Title	Command Module	Diskette	Cassette
Home Management/Personal Finance			
Home Financial Decisions	\$29.95	—	—
Household Budget Management	39.95	—	—
Securities Analysis	54.95	—	—
Personal Record Keeping	49.95	—	—
Tax/Investment Record Keeping	69.95	—	—
Personal Real Estate	69.95	—	—
Mailing List	—	69.95	—
Personal Financial Aids	—	19.95	—
Checkbook Manager	—	19.95	—
Bus. Aids Lib. - Inventory Mgmt.	—	69.95	—
Bus. Aids Lib. - Invoice Mgmt.	—	69.95	—
Bus. Aids Lib. - Cash Mgmt.	—	39.95	—
Personal Financial Aids	—	—	14.95
Education/Personal Enrichment			
Early Learning Fun	29.95	—	—
Beginning Grammar	29.95	—	—
Number Magic	19.95	—	—
Video Graphs	19.95	—	—
Video Chess	69.95	—	—
Physical Fitness	29.95	—	—
Early Reading †	54.95	—	—
Music Maker	39.95	—	—
Weight Control and Nutrition	59.95	—	—
Addition and Subtraction I †	39.95	—	—
Addition and Subtraction II †	39.95	—	—
Multiplication I †	39.95	—	—
TI LOGO (Requires Mem. Exp. Unit)	199.95	—	—
Teach Yourself BASIC	—	34.95	29.95
Music Skills Trainer	—	29.95	24.95
Computer Music Box	—	19.95	14.95
Market Simulation	—	19.95	14.95
Teach Yourself Extended BASIC	—	24.95	19.95
Music Maker Demonstration	—	14.95	—
Bridge Bidding I	—	29.95	24.95
Speak & Spell Program	—	29.95	—
Speak & Math Program	—	29.95	—
Bridge Bidding II	—	29.95	24.95
Bridge Bidding III	—	29.95	24.95
Spellwriter	—	29.95	24.95
Entertainment			
Football	29.95	—	—
Video Games I	29.95	—	—
Hunt The Wumpus	24.95	—	—
Indoor Soccer	29.95	—	—
Mind Challengers	24.95	—	—
A-Maze-Ing	24.95	—	—
The Attack ††	39.95	—	—
Blasto ††	24.95	—	—
Blackjack and Poker	24.95	—	—
Hustle ††	24.95	—	—
Zero Zap ††	19.95	—	—
Hangman ††	19.95	—	—
Connect Four ††	19.95	—	—
Yahtzee ††	24.95	—	—
TI-Trek (w/speech)	—	14.95	—
Mystery Melody	—	14.95	9.95
Oldies But Goodies - Games I	—	19.95	14.95
Oldies But Goodies - Games II	—	24.95	19.95
Saturday Night Bingo	—	29.95	24.95
Draw Poker	—	24.95	19.95
Others			
Diagnostic	29.95	—	—
Demonstration	69.95	—	—
Speech Editor	44.95	—	—
Statistics	44.95	—	—
Extended BASIC	99.95	—	—
Terminal Emulator II	49.95	—	—
Programming Aids I	—	14.95	9.95
Programming Aids II	—	24.95	—
Math Routine Library	—	29.95	24.95
Electrical Engineering Library	—	29.95	24.95
Programming Aids III	—	19.95	—
Graphing Package	—	19.95	14.95
Structural Engineering Library	—	29.95	24.95

† Developed by Scott Foresman

†† Developed by Milton Bradley

ORDER BY MAIL.

Send to: Unisource Electronics, Inc., Box 64240, Lubbock, Texas 79464

Name _____

Address _____

City _____ State _____ Zip _____

Please send me: _____

Subtotal \$ _____

Texas residents add 5% sales tax\$ _____

Shipping and handling\$ 2.00

TOTAL ORDER \$ _____

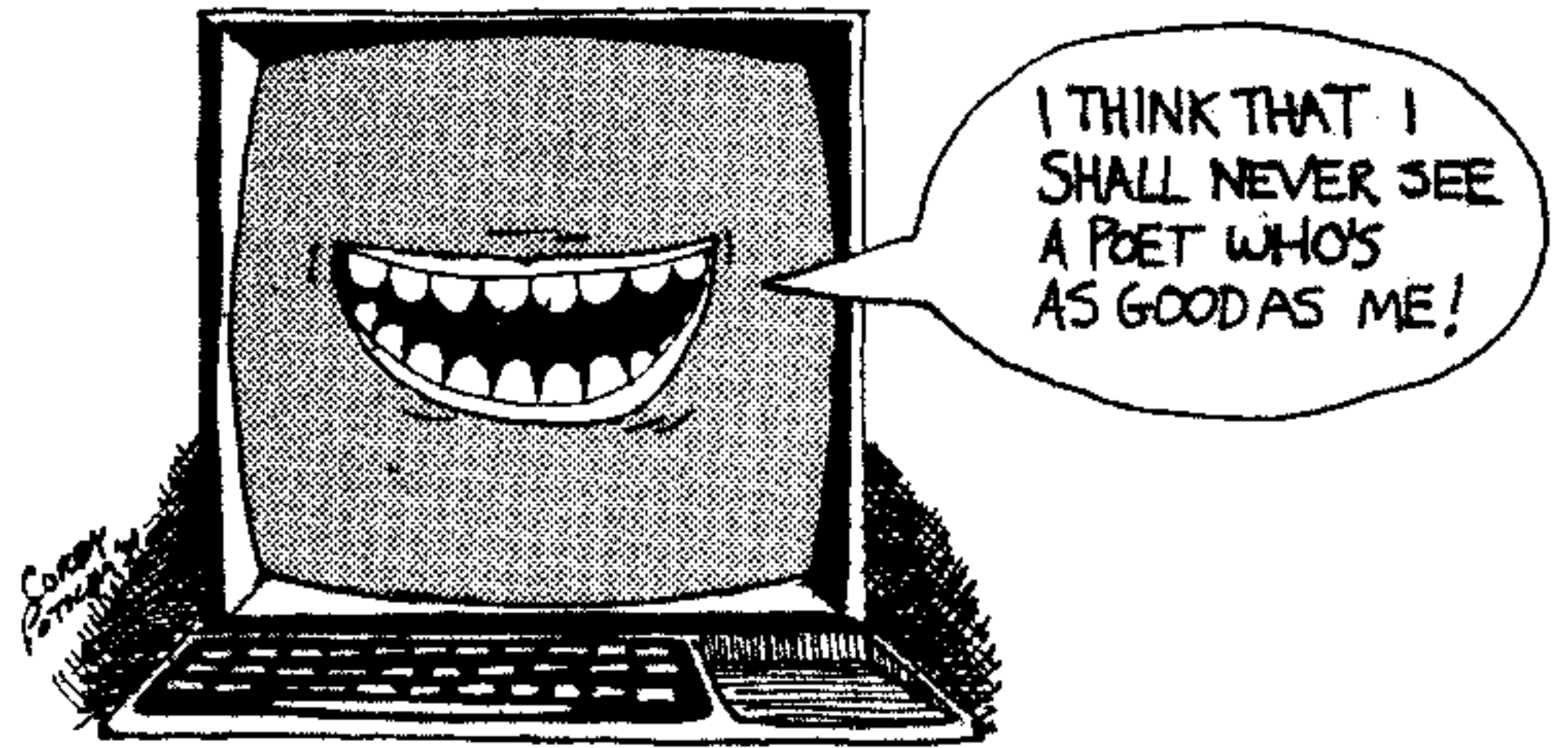
Check or money order enclosed.

Visa MasterCard Card No. _____

Exp. Date _____ Your Signature _____

The LOGO Poet:

USING RECURSION FOR LIST HANDLING



By Henry Gorman

Since TI LOGO's graphics capabilities are so vast and so easily used, there is a tendency to overlook its other features. List handling is a case in point: By combining some of LOGO's list primitives—such operations as FIRST, BUTFIRST (or the converse LAST, BUTLAST)—with recursive [see the adjoining *A Primer on Recursion and List Primitives*] OUTPUT lines, we can easily write programs to reverse a list, alphabetize a list, or even compose poetry. The several examples that follow will, I hope, demonstrate to you the powerful simplicity and list-manipulation potential of the language.

A commonly encountered problem in list processing is that of verifying the presence or absence of a word in a list of words. The MIT LOGO group refers to this as the "MEMBER?" problem because the program is to answer the question, "Is a specified word in a specified list?" Some aspects of the program are obvious, for example, once the answer is obtained (whether TRUE or FALSE) it should OUTPUT to the user or program which called for the answer. It is also obvious that if the list is empty, the word was not in the list. Given just this much information, it is possible to frame a MEMBER? program:

```
TO MEMBER? WORD :LIST
IF :LIST = [ ] OUTPUT "FALSE
:
:
END
```

Papert, following Polya [see *Mindstorms: Children, Computers, and Powerful Ideas* by Seymour Papert—available from the 99'er Bookstore], notes that one way of solving a complex problem is to ignore the complex whole and focus on those parts which can easily be solved. Here, with this problem, if the *first* word in the list were the target word, then it would be easy to detect it and solve the problem:

```
TO MEMBER? :WORD :LIST
IF :LIST = [ ] OUTPUT "FALSE
IF FIRST :LIST = :WORD OUTPUT "TRUE
:
:
END
```

Now all that remains is solving for those cases in which the word is in an *interior* position or is *absent* from the list. Were the word *second* in the list, the problem would be solved by adding a line:

```
IF FIRST BUTFIRST :LIST = :WORD OUTPUT "TRUE
```

since the *second* word in the list is the *first* word in a list which excludes the first word. Similarly, the third word becomes the FIRST of the BUTFIRST of the BUTFIRST of the list, the fourth word is the FIRST of the BUTFIRST of

the BUTFIRST of the BUTFIRST of the list. It would be possible to write a separate line for each of those positions as well as the fifth, sixth, seventh or any other potential word position. However, a program that did this would quickly grow ponderous. Fortunately, in LOGO this is unnecessary. Notice that for each position an additional BUTFIRST is all that is needed. The problem therefore requires only a *single* recursion line to complete the program:

```
TO MEMBER? :LIST :WORD
IF :LIST = [ ] OUTPUT "FALSE
IF FIRST :LIST = :WORD OUTPUT "TRUE
OUTPUT MEMBER? BUTFIRST :LIST :WORD
END
```

Now when we run the program by typing MEMBER? [A QUICK BROWN FOX] "FOX, the first stack checks to see if the list is empty or if the first word in the list matches the target word, FOX. Then it awaits the results of a second stack which runs MEMBER? with the truncated list and the target word. The second stack then awaits the results of a third stack which runs MEMBER? on BROWN FOX and "FOX. That stack then awaits the results of MEMBER? FOX "FOX which returns "TRUE (from the match in the second line). "TRUE is returned to the second stack which outputs "TRUE to the first stack which outputs "TRUE to the program which first called it (or to top level). In event that there were no matches, one of the stacks would eventually run MEMBER? on an empty list and would output "FALSE.

FIRST returns the first word in a list of words, or the first letter in a list of words, or the first letter in a word. LAST returns the last letter in a list of words, or the last letter in a word. BUTFIRST returns all but the first word in a list of words, or all but the first letter in a word. BUTLAST returns all but the last word in a list of words, or all but the last letter in a word.

Another common problem is to count the number of words in a list of words. As before, the way to solve this problem is to outline the obvious elements of the solution and the simplest case.

```
TO COUNT "LIST
OUTPUT some number
END
```

The simplest case occurs when the list is empty.

```
TO COUNT :LIST
IF :LIST = [ ] OUTPUT 0
OUTPUT some number
END
```

When a list has just *one* word in it, the program should recognize that and OUTPUT 1. Since a list with just one word is one word away from an empty list, The LOGO operation BUTFIRST applied to that list would yield the empty list. If there were *two* words in a list, then obviously the list is just *two* words away from an empty list. If a recursive line were put into the program which (a) applied BUTFIRST and (b)

added 1 to the count for every application of BUTFIRST, the program would count the words in the list.

```
TO COUNT :LIST
IF :LIST = [ ] OUTPUT 0
OUTPUT (COUNT BUTFIRST :LIST) +1
END
```

For another example, consider a program which will reverse a list. The simplest case would be a list with no words.

```
TO REVERSE :LIST
IF :LIST = [ ] OUTPUT [ ]
END
```

The next simplest case would be a list with just one word. For such a list we could have the program OUTPUT the SENTENCE or the word and an empty list.

```
TO REVERSE :LIST
IF :LIST = [ ] OUTPUT [ ]
OUTPUT SENTENCE (LAST :LIST) (REVERSE
  BUTLAST :LIST)
END
```

This solution can be applied to longer lists as well!

For a final example, let's use LOGO to "write" random poetry. As a first effort at LOGO poetry, we'll attempt some "free verse" by instructing our poet to randomly string words together from a list we select. First, we will need a program like SELECT to output a selected item from a list.

```
TO SELECT :N :LIST
IF :N = 1 OUTPUT FIRST :LIST
OUTPUT SELECT :N - 1 BUTFIRST :LIST
END
```

Then we need a program to generate random numbers for SELECT. Since LOGO's RANDOM primitive provides the integers through nine, if our list is less than ten, we can get a COUNT of it and use that COUNT.

```
TO NUMB :LENGTH
CALL RANDOM "N
TEST BOTH :N > 0 :N < (:LENGTH + 1)
IFT OUTPUT :N
IFF OUTPUT NUMB :LENGTH
END
```

By first typing

```
CALL COUNT :LIST "LENGTH
```

we can then use NUMB for the value of LENGTH. If we then type

```
TYPE SELECT (NUMB :LENGTH) [a list of words]
```

the computer types one of the words in the list. We can write that as a program:

```
TO VERSE :LIST
TYPE SELECT (NUMB :LENGTH) :LIST
END
```

To turn this into a line of poetry, we should have a random number of such randomly picked words with a random number of spaces between words (E. E. Cummings' style) and then a carriage return:

```
TO SPACE
REPEAT RANDOM [PRINTCHAR 32]
END
TO LINE :LIST
REPEAT RANDOM [SPACE VERSE :LIST]
PRINT SELECT (NUMB :LENGTH) :LIST
END
```

Note: PRINTCHAR 32 puts the character with ASCII code 32, a space, on the screen.

If we want continuing lines of poetry, we could write a recursive program:

A PRIMER ON RECURSION AND LIST PRIMITIVES

It is easier to understand recursion in LOGO if one imagines that each LOGO program is a job for a contractor to perform. Each contractor is a specialist and can only do one job. Every contractor follows strict working rules; these rules say that when the contractor sees STOP, he must stop, when he sees OUTPUT, he must pass back some information and then stop. Of course, when a contractor reaches an END, he also stops. When a contractor sees the name of any LOGO program inside of the program he is completing, he subcontracts that job out to another contractor. Thus, in COUNT [A B C], the first contractor reads the first line of the program, but the condition isn't met, so he moves to line two. There he is told to OUTPUT 1+ the COUNT of [B C]. Since he can't do another program, he subcontracts the job. The subcontractor reads line 1 of COUNT, since it doesn't apply, he reads line 2. He is told to OUTPUT 1+ the COUNT of [C]. He can't do that, so he also subcontracts the job. The third contractor notes that line 1 doesn't apply and line 2 tells him to OUTPUT 1+ the COUNT of []. He also must subcontract the job out, and so the fourth contractor reads line 1 of COUNT. Since the list is empty, he OUTPUTs 0 and passes the job back to the third contractor; he, in turn, adds 1 and then OUTPUTs 1. In a similar way, the second contractor adds 1 to that and OUTPUTs 2. Then, the first contractor adds 1 to that and OUTPUTs 3, which is the correct answer. With this explanation, you should now be able to analyze a program which gives you the answer to a number X raised to N power.

```
TO EXPONENT :X :N
```

```
...
```

```
END
```

```
TO EXPONENT :X :N
```

```
IF :N = 0 OUTPUT 1
```

```
...
```

```
END
```

```
TO EXPONENT :X :N
```

```
IF :N = 0 OUTPUT 1
```

```
IF :N = 1 OUTPUT :X
```

```
OUTPUT (EXPONENT :X :N-1) * :X
```

```
END
```

```
TO LINES :LIST
```

```
LINE :LIST
```

```
LINES :LIST
```

```
END
```

Now, putting this all together we get:

```
TO POET :LIST
```

```
CALL COUNT :LIST "LENGTH
```

```
LINES :LIST
```

```
END
```

As an exercise, see if you can figure out how to write a LOGO POET that can compose blank verse, rhyming verse, or a set number of lines of verse. We'll examine some solutions next time.

YPLA MAKES ITS DEBUT

The Young People's LOGO Association (YPLA) is a new nonprofit corporation formed to promote a better understanding and appreciation by young people, their parents, and their teachers, of the versatility and utility of the personal computer. As an integral part of this program, YPLA will promote the use of personal computers to aid in the education of the learning disabled, and the physically and otherwise developmentally disabled. YPLA's software exchange provides a catalog of User-developed TI LOGO procedures. Young people 18 years old and under can join at no charge. Adults are asked to contribute \$15 per year. For additional information contact James H. Muller, President, YPLA, 1208 Hillside Drive, Richardson, TX 75081. Tel (214) 783-7548.

CORRECTION:

In last issue's article, *LOGO and the Space Shuttle: An Update on Lamplighter Activities*, Kay Murphy's name was inadvertently omitted as co-author. We'd like to apologize to Ms. Murphy of the Lamplighter School in Dallas, Texas.



ARE YOU A TURTLE?

The Young Peoples' LOGO Association wants you to be.

The YPLA is a brand new club with some brand new ideas -- about "turtle graphics," about personal computers. We're young people talking to young people about computers. Across the room, across town, across the country. We're young people sharing the fun and excitement of personal computing, sharing our ideas, sharing software through our own software exchange.

We're parents and teachers that want to make the computer a family affair, a more creative learning center for the home and the classroom. We're people that want to see the computer aid in the education of the learning disabled and the handicapped. We want to provide people with a central library of information on where and how this work is being done.

We're one Turtle Learning Center now with a nationwide network of interconnected local centers on the way. Where the young and the young-at-heart can enjoy learning about and using personal computers. YPLA membership is free to those 18 and under. interested adults are asked to contribute \$15.00 per year.

We're young people enjoying all sorts of things with computers, learning TI LOGO and other languages, sharing the fun of it all. learning to make the most of the computer revolution.

C'mon. Join us. Be a turtle!

.....



Young Peoples' LOGO Association
1208 Hillside Drive
Richardson, Texas 75081

I want to be a turtle. Please enroll me as an adult member. I've enclosed my \$15 contribution. I prefer a young person membership.

Name: _____

Address: _____

City: _____ State: _____ ZIP: _____

Interactive Forms . . . from p. 38

```

1220 REM >>>> NO, IT IS THE END OF REPEAT SEQUENCE
      MARK.>>>
1230 REPS=REPS+1
1240 IF REPS=MAXREPS THEN 1290
1250 INPUT "REPEAT ENTRY? (1=YES 0=NO)":MORE
1260 IF MORE=NO THEN 1290
1270 I=REPSTART
1280 GOTO 890
1290 REPEAT=NO
1300 NEXT J
1310 NEXT I
1320 RETURN
1330 REM 3 - PRINT COPIES
1340 REM FORM PRINT SECTION
1350 CALL CLEAR
1360 PRINT "ENTER NUMBER OF COPIES"
1370 INPUT "TO PRINT-":Z
1380 OPEN #3:"RS232/2.CR.EC.DA=8.BA=9600",
      VARIABLE 132
1390 PRINT #3:RESETEPSON*
1400 FOR I=1 TO Z
1410 FOR J=1 TO X
1420 IF SEG$(A$(J),1,2)=BANG$ THEN 1440
1430 PRINT #3:A$(J)
1440 NEXT J
1450 NEXT I
1460 CLOSE #3
1470 RETURN
1480 REM "COMMENT" SUBROUTINE
1490 COMMENT$=""
1500 FOR K=J+1 TO LEN(A$(I))
1510 P$=SEG$(A$(I),K,1)
1520 IF P$=QUOTE$ THEN 1620
1530 P=ASC(P$)
1540 IF P>96 THEN 1550 ELSE 1570
1550 P=P-32
1560 P$=CHR$(P)
1570 COMMENT$=COMMENT$&P$
1580 NEXT K
1590 PRINT "*** ERROR IN LINE #":I
1600 PRINT "*** MISSING QUOTE..."
1610 GOTO 1640
1620 PRINT COMMENT$
1630 PRINT ""
1640 RETURN
1650 REM "FIELDINPUT" SUBROUTINE
1660 FRONT$=""
1670 BACK$=""
1680 REM >>>> DECODE THE FIELD PARAMETERS >>>
1690 GOSUB 2670
1700 PRINT
1710 MIDDLE$=SEG$(A$(LINE),START,LENGTH)
1720 PRINT "***&MIDDLE$&***"
1730 PRINT
1740 INPUT TEXT$
1750 IF SEG$(TEXT$,1,1)=RIGHTARROW$ THEN 1760
      ELSE 1780
1760 RIGHTJUSTIFY=YES
1770 TEXT$=SEG$(TEXT$,2,LEN(TEXT$))
1780 IF LEN(TEXT$)>LENGTH THEN 1790 ELSE 1830
1790 PRINT "-- TEXT STRING TOO LONG..."
1800 PRINT "PLEASE ENTER SHORTER LINE"
1810 GOTO 1700
1820 REM >>>> GO STUFF THE FIELD >>>
1830 GOSUB 2850
1840 RETURN
1850 REM "MATH TERM" SUBROUTINE
1860 J=J+1
1870 ON TERM GOTO 1890,1990,2050
1880 REM >>>> PROCESS FIRST TERM >>>
1890 IF SEG$(A$(I),J,1)=COLON$ THEN 1900 ELSE 1930
1900 GOSUB 2670
1910 FIRSTTERM$=SEG$(A$(LINE),START,LENGTH)
1920 GOTO 1960
1930 ENDFIELD=POS(A$(I),CLOSEPAREN$,J)
1940 FIRSTTERM$=SEG$(A$(I),J,ENDFIELD-J)
1950 K=ENDFIELD
1960 TERM=2
1970 GOTO 2650
1980 REM >>>> PROCESS SECOND TERM >>>
1990 ENDFIELD=POS(A$(I),CLOSEPAREN$,J)
2000 SECONDTERM$=SEG$(A$(I),J,ENDFIELD-J)
2010 TERM=3
2020 K=ENDFIELD
2030 GOTO 2650
2040 REM >>>> PROCESS THIRD TERM AND CALCULATE >>>
2050 IF SECONDTERM$=EQUAL$ THEN 2070 ELSE 2200
2060 REM >>>> PROCESS ANSWER AND STORE AT 3RD TERM
      LOCATION >>>
2070 IF SEG$(A$(I),J,1)=COLON$ THEN 2080 ELSE 2170
2080 IF SEG$(A$(I),J+1,1)=RIGHTARROW$ THEN 2090
      ELSE 2110
2090 RIGHTJUSTIFY=YES
2100 J=J+1
2110 GOSUB 2670
2120 TEXT$=FIRSTTERM$
2130 GOSUB 2850
2140 K=K+1
2150 TERM=1
2160 GOTO 2650
2170 PRINT "ERROR IN MATH- FILE LINE#":I
2180 GOTO 2650
2190 REM >>>> GET THE THIRD TERM .....>>>
2200 IF SEG$(A$(I),J,1)=COLON$ THEN 2210 ELSE 2240
2210 GOSUB 2670
2220 THIRDTERM$=SEG$(A$(LINE),START,LENGTH)
2230 GOTO 2270
2240 ENDFIELD=POS(A$(I),CLOSEPAREN$,J)
2250 THIRDTERM$=SEG$(A$(I),J,ENDFIELD-J)
2260 K=ENDFIELD
2270 TERM=2
2280 IF POS(FIRSTTERM$,DECIMAL$,1)+POS(THIRDTERM$,
      DECIMAL$,1)=0 THEN 2310
2290 ALIGN=YES
2300 GOTO 2330
2310 ALIGN=NO
2320 REM >>>> OK, NOW DO MATH >>>
2330 IF POS(BLANK$,FIRSTTERM$,1)=NO THEN 2350
2340 FIRSTTERM$=ZERO$
2350 IF POS(BLANK$,THIRDTERM$,1)=NO THEN 2370
2360 THIRDTERM$=ZERO$
2370 FIRSTTERM=VAL(FIRSTTERM$)
2380 THIRDTERM=VAL(THIRDTERM$)
2390 IF SECONDDTERM$=PLUS$ THEN 2400 ELSE 2420
2400 TEMPTERM=FIRSTTERM+THIRDTERM
2410 GOTO 2530
2420 IF SECONDDTERM$=MINUS$ THEN 2430 ELSE 2450
2430 TEMPTERM=FIRSTTERM-THIRDTERM
2440 GOTO 2530
2450 IF SECONDDTERM$=MULTIPLY$ THEN 2460 ELSE 2480
2460 TEMPTERM=FIRSTTERM*THIRDTERM
2470 GOTO 2530
2480 IF SECONDDTERM$=DIVIDE$ THEN 2490 ELSE 2510
2490 TEMPTERM=FIRSTTERM/THIRDTERM
2500 GOTO 2530
2510 PRINT "MATH OPERATOR BAD - FILE LINE #":I
2520 GOTO 2650
2530 TEMPTERM=INT(TEMPTERM*100)/100
2540 TEMPTERM$=STR$(TEMPTERM)
2550 IF ALIGN=NO THEN 2640
2560 IF POS(TEMPTERM$,DECIMAL$,1)=0 THEN 2590
2570 ADJUST=1+LEN(TEMPTERM$)-POS(TEMPTERM$,DECIMAL$,1)
2580 ON ADJUST GOTO 2600,2620,2640
2590 FIRSTTERM$=TEMPTERM$&DECIMAL$&ZERO$&ZERO$
2600 TEMPTERM$=TEMPTERM$&DECIMAL$&ZERO$
2610 RETURN
2620 FIRSTTERM$=TEMPTERM$&ZERO$
2630 RETURN
2640 FIRSTTERM$=TEMPTERM$
2650 RETURN
2660 REM GET FIELD DEF. SUBROUTINE
2670 K=J+1
2680 NEXTCOLON=POS(A$(I),COLON$,K)
2690 LINE=VAL(SEG$(A$(I),K,NEXTCOLON-K))
2700 K=NEXTCOLON+1
2710 NEXTCOLON=POS(A$(I),COLON$,K)
2720 START=VAL(SEG$(A$(I),K,NEXTCOLON-K))
2730 K=NEXTCOLON+1
2740 NEXTCOLON=POS(A$(I),COLON$,K)
2750 LEND=VAL(SEG$(A$(I),K,NEXTCOLON-K))
2760 K=NEXTCOLON+1
2770 IF REPEAT=NO THEN 2790
2780 LINE=LINE+REPS
2790 LENGTH=LEND-START+1
2800 IF LENGTH<1 THEN 2810 ELSE 2830
2810 PRINT "*** ERROR IN LINE #":I
2820 PRINT "*** FIELD LENGTH NEGATIVE..."
2830 RETURN
2840 REM FIELD STUFFER SUBROUTINE
2850 IF LEN(TEXT$)=0 THEN 3000
2860 IF RIGHTJUSTIFY=NO THEN 2920
2870 FOR M=LEN(TEXT$) TO LENGTH-1
2880 TEXT$=SPACE$&TEXT$
2890 NEXT M
2900 RIGHTJUSTIFY=NO
2910 GOTO 2950
2920 FOR M=LEN(TEXT$) TO LENGTH-1
2930 TEXT$=TEXT$&SPACE$
2940 NEXT M
2950 IF START=1 THEN 2970

```

Continued on p. 77

Super Language . . . from p. 55

tions Register Unit) lines to the monitor. Under TXMIRA, all peripheral devices are addressed via a fairly complex arrangement of CRU lines. Each device has its own CRU base address and CRU bit assignments, which means that a programmer must have very specific information about each device in order to perform any input or output. On the 99/4 assembler, these difficulties in handling the screen have been eliminated by the Basic Support Utilities. By loading a few registers and invoking the proper utility, a programmer can handle screen I/O in a much simpler way. Here are code segments which might be used for writing the characters AB to the upper left portion of the screen:

The editor/assembler package has no special debugging utility. Personally, I do not find that this is a hardship. Even on systems with extensive debugging packages, I tend to debug without using those special facilities because they are hard to use and harder to understand. Furthermore, every package is different. My method of debugging usually consists of adding assembly language statements to my source program which enable me to isolate the problem, and then removing these statements after the problem has been fixed. For example, if I want a trace of program execution, I just put temporary statements to display meaningful messages at different points in the program. The only danger in this ap-

Larger system (TXMIRA):		
LI	2,0	MOVE 0 TO REGISTER 2 FOR INDEX
LI	12,>CO	SET CRU BASE ADDRESS FOR SCREEN
SBO	>F	SELECT CRU WORD 1
LDCR	@ZERO,11	MOVE CURSOR TO HOME POSITION
SBZ	>F	SELECT CRU WORD 0
LOOP	LDCR @AB(2),7	PUT CHARACTER ON CRU LINE
	SBZ >8	STROBE CHARACTER TO SCREEN
	SBZ >A	INCREMENT CURSOR POSITION
	INC 2	ADD 1 TO INDEX REGISTER
	CI 2,2	COMPARE REGISTER 2 TO 2
	JLT LOOP	LOOP IF MORE CHARACTERS
...		
ZERO	DATA 0	DATA DEFINITIONS
AB	TEXT 'AB'	
99/4 assembler:		
REF	VMBW	EXTERNAL REFERENCE TO BASIC SUPPORT UTILITY
...		
LI	0,0	VDP RAM ADDRESS = 0 FOR HOME POSITION
LI	1,AB	REGISTER 1 POINTS TO FIRST CHARACTER TO DISPLAY
LI	2,2	REGISTER 2 = NUMBER OF BYTES TO WRITE
BLWP	@VMBW	CALL BASIC SUPPORT UTILITY TO WRITE STRING
...		
AB	TEXT 'AB'	DATA DEFINITION

You can see that the Basic Support Utilities really make screen handling easier—by focussing your attention on merely the VDP RAM (the memory associated with the 99/4 monitor) addresses, and not having to worry about the logistics of the move. Furthermore, there is no apparent loss of execution speed in doing it this way.

Another difference between the 99/4 assembler and those for larger TI computers is that the IDLE instruction is not implemented on the 99/4. This causes no great difficulty, but it is useful to know. The IDLE instruction just causes the computer to wait for an interrupt; this can be done via another Basic Support Utility or other means, depending on which device will cause the interrupt.

The optional listing produced by the 99/4 assembler is quite complete. Statement sequence numbers, source statements, and the hexadecimal code generated are all shown clearly. A symbol table can also be given and, of course, the number of errors is shown. Each error is also flagged in the body of the listing with a descriptive message. One very nice—and all too uncommon—feature is that the number of errors is also displayed on the monitor when the assembler is finished.

Running and Debugging

Once a program has been input, edited, and assembled with no errors, it can be loaded and run by choosing this option from the menu. Another menu option (RUN PROGRAM FILE) allows the user to run programs which were assembled on larger Texas Instruments systems running under the DX10 operating system.

proach is that, once debugging is complete, I might forget to remove one of these temporary statements in creating the final version. However, the 99/4's editor can find all these temporary lines for me, as long as I take the simple precaution of putting some unique comment (such as DEBUG) on each temporary line. In this way, I can debug using only the assembly language itself.

In summary, the 99/4 Assembler/Editor package is an excellent programming tool. It is easy to use and powerful, and the object programs produced run with incredible speed. This package is a *must* for serious 9900 assembly language programmers—both software developers and end users who want to access *all* the capabilities of the TI personal computer.

Note: At the time of this writing, the documentation for this 99/4 Assembler/Editor package has not yet been completed. However from the preliminary version we've seen, it is apparent that the final release as planned is intended for programmers who already have experience with assembly language programming. Texas Instruments has confirmed that the package will not be structured as a tutorial for beginners. To fill this information gap, Emerald Valley Publishing Co. (the parent company of *99'er Magazine*) will be publishing a companion book (in mid 1982) that is aimed at taking the BASIC programmer step-by-step into the realm of assembly programming on the TI-99/4(A).

Table 3
Condensed Format Code Table

129 ELSE	171 ???	213 LEN(
130 :	172 ???	214 CHR\$(
131 I	173 ???	215 RND(
132 IF	174 ???	216 SEG\$(
133 GO	175 ???	217 POS(
134 GOTO	176 THEN	218 VAL(
135 GOSUB	177 TO	219 STR\$(
136 RETURN	178 STEP	220 ASC(
137 DEF	179 ,	221 PI
138 DIM	180 ;	222 REC
139 END	181 :	223 MAX(
140 FOR	182)	224 MIN(
141 LET	183 (225 RPTS(
142 BREAK	184 &	226 ???
143 UNBREAK	185 ???	227 ???
144 TRACE	186 OR	228 ???
145 UNTRACE	187 AND	229 ???
146 INPUT	188 XOR	230 ???
147 DATA	189 NOT	231 ???
148 RESTORE	190 =	232 NUMERIC
149 RANDOMIZE	191 <	233 DIGIT
150 NEXT	192 >	234 UALPHA
151 READ	193 +	235 SIZE
152 STOP	194 -	236 ALL
153 DELETE	195 *	237 USING
154 REM	196 /	238 BEEP
155 ON	197 \	239 ERASE
156 PRINT	198 ???	240 AT
157 CALL	199 ???	241 BASE
158 OPTION	200 ???	242 ???
159 OPEN	201 ???	243 VARIABLE
160 CLOSE	202 EOF	244 RELATIVE
161 SUB	203 ABS(245 INTERNAL
162 DISPLAY	204 ATN(246 SEQUENTIAL
163 IMAGE	205 COS(247 OUTPUT
164 ACCEPT	206 EXP(248 UPDATE
165 ERROR	207 INT(249 APPEND
166 WARNING	208 LOG(250 FIXED
167 SUBEXIT	209 SGN(251 PERMANENT
168 SUBEND	210 SIN(252 TAB
169 RUN	211 SQR(253 (# files)
170 INPUT	212 TAN(254 VALIDATE

Table 4
Condensed Record Structure

OPEN #1:"DSK1.BASIC",INPUT,DISPLAY,VARIABLE 163

ASCII CODE FOR LINE 100

3 159* OPEN	23 179* COMM A
4 253* #	24 162* DISPLAY
5 200* NUMBER FOLLOWS	25 179* COMM A
6 1 AND. COMMENTS	26 243* VARIABLE
7 49 1	27 200* NUMBER FOLLOWS
8 181* COLOR	28 3 NO. OF NUMBER.
9 199* SPACE FOLLOWS	29 49 1
10 10 NO. OF STAMPS	30 54 6
11 68 D	31 51 3
12 83 S	32 0 END OF LINE
13 75 K	
14 49 1	
15 46 .	
16 66 B	
17 65 A	
18 83 S	
19 73 I	
20 67 C	
21 79* COMM A	
22 146* INPUT	

PRESS ANY KEY TO CONTINUE

Continued on p. 78

TOP QUALITY Professional Programming for your TI-99/4

ACCOUNTING PACKAGE

Runs on cassette; no disks or printers needed. Enter journal in any order; output is journal in date order; general ledger in date within account no. order, balance sheet and profit & loss statement. Copy from screen to books. RAM sort limits max. 100 journal entries per run. Takes about 1½ hours. Saves account balances for next run. **\$50.00**

EARLY READING PACKAGE

Your words or ours are displayed in giant lower case letters. Child must type word on keyboard—each letter is checked when typed. Child has another chance if wrong; child's letter displayed under the original if right. Musical tune is rewarded when word correctly completed. Speech Synthesizer optional. **\$15.00**

WORD LIST PACKAGE

Can be used alone, or with Early Reading. Type in your child's short story. Computer sorts words, deletes duplicates and saves on tape a list of words to be learned for reading that story. **\$10.00**

CONCENTRATION

A version of pelmanism using your choice of 10 three letter words—each scattered randomly two times on a 4x5 grid. Words are displayed in place then replaced by numbers. Object is to remember which numbers covers which word and find matches. Computer keeps score and also decides first player. Speech Synthesizer optional. **\$10.00**

PERCENTAGE COMPARISONS

Enter and save current and historical values of stocks, currency, commodities, etc. Calculates, sorts, and lists by percentage changed. **\$7.50**

Other Programs Available.
Write or Call for Free Catalog.

Custom programming
& dealer inquiries invited

In USA & Canada, send check or Money Order. Elsewhere send International Money Order. (Prices are in U. S. dollars. Canadians add 20% for CAN\$ Orders; Ontario residents add further 7% P. S. T.)

Master Charge & VISA Accepted for mail & phone orders worldwide
(Include all information embossed on card.)

Programs Sent Via Airmail
on Tape Cassette

ANTHISTLE SYSTEMS & PROGRAMMING LTD.,

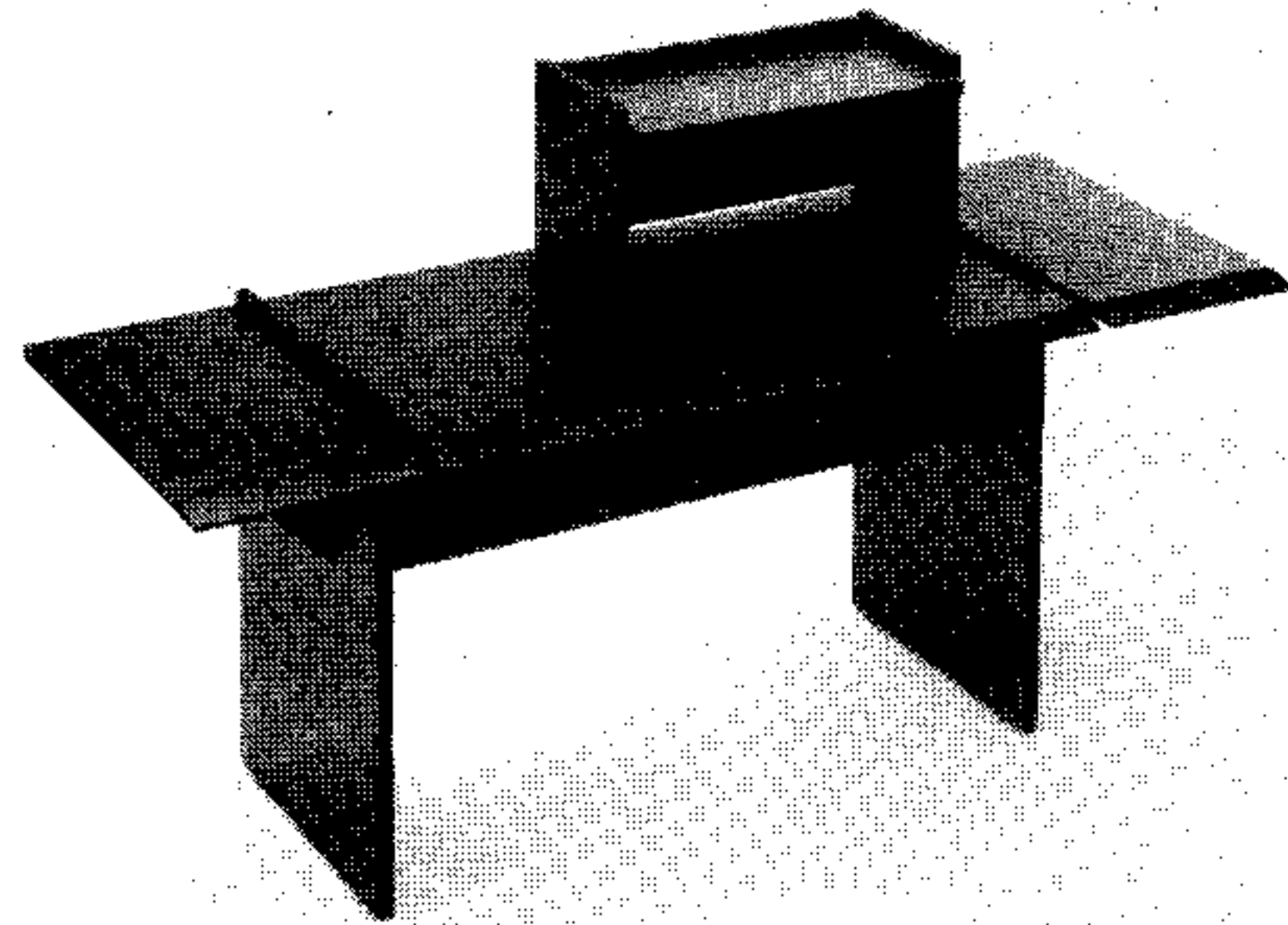
563 Patricia Drive
Oakville, Ontario
Canada, L6K 1M4
Tel. (416) 845-7959

Interactive Forms . . . from p. 75

```
2960 FRONT$=SEG$(A$(LINE),1,START-1)
2970 IF LEND=LEN(A$(LINE)) THEN 2990
2980 BACK$=SEG$(A$(LINE),LEND+1,LEN(A$(LINE)))
2990 A$(LINE)=FRONT$&TEXT$&BACK$
3000 RETURN
3010 END
```

CUSTOM DESK FOR THE 99/4

This attractive, wooden desk with a laminated walnut grain finish is specifically designed for your 99/4 System. 24" Deep, and 40" Wide (64" Wide if you add both extension shelves), allows you to organize your computer and all of your accessories. You gain even more space by adding this monitor shelf pictured above. A six plug power strip is available for your convenience. The desk assembles in minutes without tools. Freight paid by Computer Roomers. Please allow 2 weeks for delivery.



ORDER FORM

_____ BASIC DESK @ \$145
1 or 2 EXTENSION SHELVES (12")
 @ \$20 each
 _____ MONITOR SHELF @ \$35
 _____ PLUG STRIP @ \$30

COMPUTER ROOMERS, INC.
 1250 MAJESTY DRIVE, DALLAS, TX 75247
 (214) 630-0280

CHECK ENCLOSED
 (ADD 5% FOR TEXAS DELIVERIES)
 MASTERCARD# _____ EXP. _____
 VISA# _____ EXP. _____

SHIP TO _____

At the same time, several codes are reconstructed into things which can't be understood directly (e.g., 171-175, 185, and 198-201). It is apparent that some of the ASCII codes are used for purposes other than direct translation to BASIC. Some might be used as descriptors of subsequent bytes (i.e., for purposes of identifying trailing bytes as numeric data, line number references, string data, etc.) while other of these ASCII codes may not be assigned at all.

```

100 REM *****
110 REM *
120 REM * CONDENSED FORMAT *
130 REM * CODE TABLE *
140 REM *
150 REM *****
160 REM
170 REM BY JOHN CLULOW
180 REM 99'ER VERSION 9.81.1XB
190 REM
200 REM *****
210 REM
220 REM OPEN OUTPUT FILE
230 REM "DSK1.FILENAME"
240 REM USING PARAMETERS OF
250 REM MERGED FILE FORMAT
260 REM
270 REM *****
280 REM
290 OPEN #1:"DSK1.FILENAME",
    DISPLAY ,OUTPUT,VARIABLE 163
300 REM
310 REM *****
320 REM
330 REM BEGIN COUNTING (I)
340 REM WITH HIGH BIT ON
350 REM I.E. 129 130 ... 254
360 REM
370 REM *****
  
```

```

380 REM
390 FOR I=129 TO 254
400 REM
410 REM *****
420 REM
430 REM CALCULATE VALUES FOR
440 REM FIRST TWO BYTES TO
450 REM REPRESENT LINE NO.S
460 REM SO THAT LINE NO WILL
470 REM EQUAL ASCII CODE
480 REM
490 REM *****
500 REM
510 LNBYTE1=INT(I/256)
520 LNBYTE2=I-256*LNBYTE1
530 REM
540 REM *****
550 REM
560 REM WRITE RECORD:
570 REM
580 REM BYTE#1&2=LINE NUMBER
590 REM BYTE#3=CODED BASIC
600 REM BYTE#4=END OF LINE
610 REM
620 REM *****
630 REM
640 PRINT #1:CHR$(LNBYTE1)&CHR$(LNBYTE2)&CHR$(I)&CHR$(0)
650 REM
660 REM *****
670 REM
680 REM REPEAT LOOP FOR NEXT
690 REM ASCII CODE.
700 REM
710 REM *****
720 REM
730 NEXT I
740 REM
750 REM *****
760 REM
770 REM WRITE END OF FILE
780 REM MARK = LINE NUMBER
790 REM OF 65535
800 REM
810 REM *****
  
```

```

820 REM
830 PRINT #1:CHR$(255)&CHR$(255)
840 REM
850 REM *****
860 REM
870 REM CLOSE FILE AND STOP
880 REM
890 REM *****
900 REM
910 CLOSE #1
920 STOP
  
```

*ALSO SHOULD AS
 DSK1.EXT BASLIST
 SAVED AS DSK1.FILENAME*

Putting this question aside for the moment, let us see how we could write a program that would remove all REM statements from another program. The ASCII code for REMARK (REM) is found in Table 3 to be 154. If we assume that the ASCII character with code 154 will be found in the third position of a REM statement in condensed format (following the line-number bytes), we can write a "REM Remover" program very simply. Such a program would need to read a record from a program file saved with the MERGE option, see if the third byte is CHR\$(154), and if not, print the record in a second file. That is what the following program does. To use it with the "Condensed Format Code Table" program, save that program with the MERGE option (SAVE DSK1.CODE, MERGE), run the "REM Remover," and load the output file, DSK1.REMFREE, with the MERGE command (MERGE DSK1.REMFREE). Presto, Chango! LISTing the program shows it to be "REMless," and this

version may now be saved in the usual way under a new file name.

```

100 REM *****
110 REM *   REM REMOVER   *
120 REM *****
130 REM
140 REM BY JOHN CLULOW
150 REM 99'ER VERSION 9.81.1XB
160 REM
170 PRINT "ENTER FILE NAME"
180 INPUT "DSK1.XYZ" - ":X$
190 OPEN #1:X$,DISPLAY ,INPUT ,
    VARIABLE 163
200 OPEN #2:"DSK1.REMFREE",
    DISPLAY ,OUTPUT,VARIABLE 163
210 EOF%=CHR$(255)&CHR$(255)
220 LINPUT #1:X$
230 IF SEG$(X$,1,2)=EOF% THEN 270
240 IF SEG$(X$,3,1)=CHR$(154)
    THEN 260
250 PRINT #2:X$
260 GOTO 220
270 PRINT #2:CHR$(255)&CHR$(255)
280 CLOSE #1
290 CLOSE #2
300 STOP

```

Of course, more complex applications require a more detailed knowledge of condensed format structure. The "Condensed Record Structure" program listed below will allow you to examine the condensed structure of every line in any BASIC program. With such a representation and the list of codes in Table

```

100 REM *****
110 REM *   CONDENSED RECORD   *
120 REM *   STRUCTURE         *
130 REM *   *
140 REM *   *
150 REM *****
160 REM
170 REM BY JOHN CLULOW
180 REM 99'ER VERSION 9.81.1XB
190 REM
200 OPEN #1:"DSK1.BASIC",INPUT ,DISPLAY ,VARIABLE 163
210 LINPUT #1:X$
220 BYTE1=ASC(SEG$(X$,1,1))
230 BYTE2=ASC(SEG$(X$,2,1))
240 LINENUM=BYTE1*256+BYTE2
250 IF LINENUM=65535 THEN 430
260 DISPLAY AT(1,3)ERASE ALL:"ASCII CODE FOR LINE "&STR$(LINENUM)
270 COL=1 :: J=0
280 FOR I=3 TO LEN(X$)
290 IF I>62 THEN 400
300 ROW=I-2*(COL-1)
310 J=J+1
320 DISPLAY AT(ROW,COL):STR$(I)
330 Y=ASC(SEG$(X$,I,1))
340 DISPLAY AT(ROW,COL+3):STR$(Y)
350 IF Y>128 THEN DISPLAY AT(ROW,COL+6):"*"
360 IF Y>31 AND Y<91 THEN DISPLAY AT(ROW,COL+6):CHR$(Y)
370 IF J<20 THEN 390
380 COL=COL+10 :: J=1
390 NEXT I
400 DISPLAY AT(24,2)BEEP:"PRESS ANY KEY TO CONTINUE"
410 CALL KEY(0,K,S):: IF S=0 THEN 410
420 GOTO 210
430 STOP

```

3, a great deal of additional information can be deduced.

For purposes of illustration, let us treat the "Record Structure" program itself as the program to be analyzed. First enter the program without the REM statements, and then save it as DSK1.BASIC, MERGE. Now enter RUN to display the code structure of each line. The display for the first line is shown in Table 4.

The first column in each pair of columns shows the position of the byte code. The first position displayed is 3

because 1 and 2 are used for the line number. An asterisk has been placed beside all ASCII codes which exceed 128 to easily identify them as "instruction" codes. Codes which are between 32 and 94 are followed by their corresponding ASCII character representations.

Since it is known that the first line of the program is OPEN #1:"DSK1.BASIC", INPUT, DISPLAY, VARIABLE 163 let us see what sense can be made of the corresponding condensed code. Codes 159 and 253 correspond to OPEN and

CHANGES IN PROGRAMMING AIDS III

1. Program CREF — This change should be made in the original version of CREF to prevent occasional file errors. Line 816 traps records which are too long, line 817 truncates these records, and lines 818 and 819 inform you that this has occurred and to what keyword, variable name etc.

```

816 IF LEN (E$(II))<254 THEN 820
817 E$(II)=SEG$(E$(II),1,253)
818 PRINT "* LINE TRUNCATED FOR:"
819 PRINT " "&N$(II)

```

2. Users who do not have a printer may want to make the following changes in the CREF program. The original program (without the changes) should be retained in the event of future use of a printer.

- Delete lines: 220, 230, and 820
- Delete the characters "#2" from lines: 400, 520, 580, 620, 660, 700, 730, 750, 860, 880, and 890
- Replace the characters ";CHR\$(13);" with the character ":" in line 520

#. Although the meaning of code 200 is not known, in looking ahead to columns 6 and 7 we might hypothesize that 200 means "A number is about to be encountered, and the next byte will give the number of bytes used to represent that number."

Although 181 is a ":", 199 is another unknown. Looking ahead at positions 10-20, we might again hypothesize that 199 is used for strings, in the way that 200 is used for numbers. The "10" in position 10 is consistent with this hypothesis since DSK1.BASIC is 10 characters long. Next, we encounter the codes for ,INPUT,DISPLAY, VARIABLE . In position 27 another 200 is encountered, and the hypothesis applied earlier to the 200 in position 5 is consistent with what follows—a "3" in position 28 followed by the 3 numbers "163". Finally, a 0 is encountered that indicates end-of-line. By writing program lines specifically for the purpose, you can use the "Condensed Record Structure" program to deduce additional information about condensed format.

Part II of this series will discuss the writing of more sophisticated utility programs and "generator" programs —i.e., BASIC programs that can write other BASIC programs.

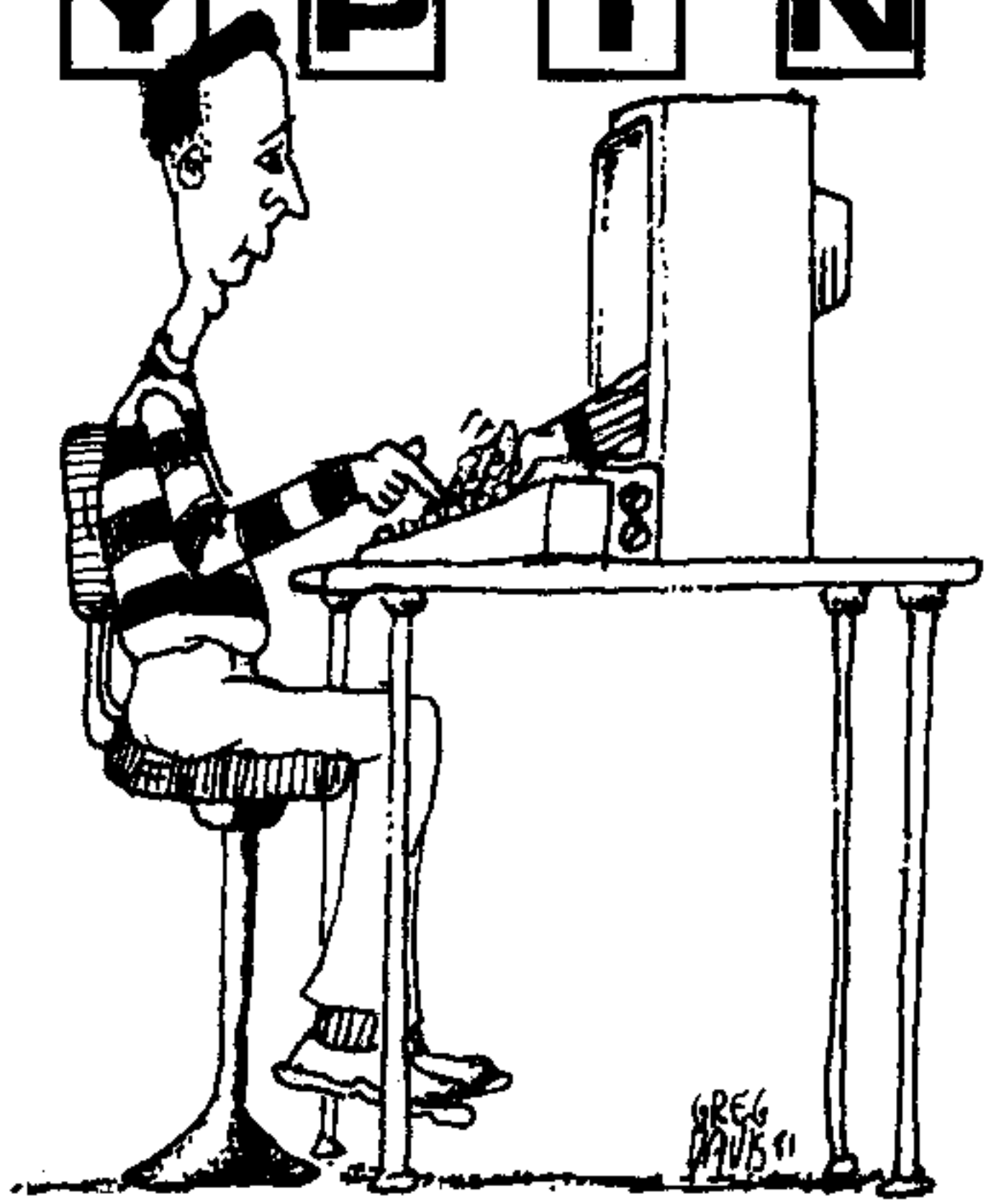
References

Thurlow, D., *TMS9900 machine and assembly language: Binary number system*, 99'er Magazine, 1981, vol 1, no 1, p. 48-51.

Whitsitt, R. E., II. *TI Extended BASIC for the 99/4 Home Computer*. Dallas: Texas Instruments, 1981.

TYPING

U T O R



Part 2: Typing for Accuracy

By Regena

Typing for Accuracy provides a student with practice in typing words after he or she has learned the basic keyboard layout. The words used in the drill are in categories depending on where the keys are located on the keyboard.

Seven finger-placement categories using different typewriter keys are offered: home keys; home row; top row, middle finger; top row, pointer finger; ring finger; little finger; and bottom row. A typist may choose one of the categories for each drill.

The program uses graphics and sound effects to liven up the drill: A rocket appears on the screen, and a word is printed on the rocket while a 1.5-second tone sounds. A student then types and enters the word. If it has been typed *incorrectly*, the rocket blasts; if it has been typed *correctly*, a second tone is sounded and the score is incremented. The rocket then moves upward (with gases trailing behind) and a different word appears.

At the end of ten words the student's score is tabulated and displayed as a percent accuracy rating. The student may

then choose from the seven drills or may exit the program.

This drill is not meant to be a speed drill since beginning typing students need to gain accuracy and familiarity with the keyboard before working on speed. However, if the student wants a time test, an approximate words-per-minute rate can be estimated using the tones—i.e., if words are entered as the tone ends, the rate is 40 wpm.

EXPLANATION OF THE PROGRAM *Typing for Accuracy*

Line Nos.	Description	Line Nos.	Description
170	Dimensions the array A\$ to allow for twenty words.	610-630	Prints the word to be typed on the rocket.
180	Sets the y-coordinate for drawing the rocket.	640	Awaits the student's typed word.
190-250	Words used in the drills.	650-690	Compares the student's word with the given word. If it is incorrect, a white noise is sounded; if it is correct, a tone sounds and the score is incremented.
260	Prints the title screen.	700-750	If it is the first word, draws the bottom of the rocket.
270	Prints instructions.	760-780	If it is the second word, completes the fins of the rocket.
280	Prints the menu screen of the seven categories.	790-830	The rocket moves up and has a trail under it. The words are cleared.
290-450	Awaits the student's choice. Depending on the category chosen, a certain DATA statement is RESTORED which contains the words for that particular category.	840	A\$ set to zero so the word cannot be used again.
460-490	Draws the rocket.	850	Returns for next word.
500-530	Reads the number of words in the category and stores the words in the A\$ array.	860-940	Prints score and waits until student is ready to continue.
540	Initializes the score.	960	END.
550-580	Randomly chooses a word. Once a word is chosen it is not used again in the drill.	Subroutines	
590	Calculates the coordinate for printing the word.	970-1580	Prints title screen with music.
600	Sounds the tone for 1.5 seconds.	1590-1730	Prints instructions.
		1740-1930	Prints menu screen of seven categories.
		1940-2160	Draws the rocket.

```

100 REM *****
110 REM * TYPING FOR ACCURACY *
120 REM *****
130 REM 99'ER VERSION 9.81.1
140 REM BY REGENA
150 REM
160 REM
170 DIM A$(20)
180 Y=20
190 DATA 15, FAD, A, AS, DAD, AD, SAD, LAD, FALL, ALFALFA, SASS, LASS, DADS, LADS, FALLS, FADS
    
```



Typing Tutor . . .

```

200 DATA 16, HAD, HAS, GAS, SAG, HALL, HALLS, LADS, SAGS, HAG, LAG, LAGS, SLAG, SHALL, SASH, DA
    SH, FLASH
210 DATA 17, DEAF, DEED, SEED, FEED, HEED, LIKE, KILL, FILL, FEEL, FEES, LIED, DIAL, SLIDE, FL
    IES, LIFE, SLID, GLIDE
220 DATA 17, TAG, HAT, TALL, THY, DAY, HAY, JAY, GAY, LAY, TAR, RAT, STAR, STAFF, FAST, TRY, SAY
    , YARD
230 DATA 20, WISH, EXAM, EXACT, TEXT, TWO, WON, SOW, WASH, WORLD, OWE, WORD, LOOK, LOSE, SOD, W
    OW, TOW, TEXAS, OXEN, MIX, WORSE
240 DATA 16, QUAKE, QUIZ, QUIP, ZAP, QUIT, PIQUE, PLAQUE, PUZZLE, PLAZA, SAP, ZIPPER, PRIZE,
    QUICK, SQUEEZE, ZEAL, ZIP
250 DATA 18, CALM, CAN, MEN, NIMBLE, EXACT, EXAM, MIX, NIX, BUZZ, ZOOM, NAVY, CAB, BACK, BOMB,
    ZOMBIE, CAVE, VACATE, YARMINT
260 GOSUB 970
270 GOSUB 1590
280 GOSUB 1740
290 CALL KEY(0, KEY, STS)
300 IF KEY<49 THEN 290
310 IF KEY>56 THEN 290
320 ON (KEY-48) GOTO 330, 350, 370, 390, 410, 430, 450, 960
330 RESTORE 190
340 GOTO 460
350 RESTORE 200
360 GOTO 460
370 RESTORE 210
380 GOTO 460
390 RESTORE 220
400 GOTO 460
410 RESTORE 230
420 GOTO 460
430 RESTORE 240
440 GOTO 460
450 RESTORE 250
460 CALL CLEAR
470 CALL COLOR(1, 2, 1)
480 CALL SCREEN(8)
490 GOSUB 1940
500 READ N
510 FOR I=1 TO N
520 READ A$(I)
530 NEXT I
540 SCORE=0
550 RANDOMIZE
560 FOR K=1 TO 10
570 W=INT(N*RND)+1
580 IF A$(W)="0" THEN 570
590 XC=24-K
600 CALL SOUND(1500, -1, 2)
610 FOR J=1 TO LEN(A$(W))
620 CALL HCHAR(XC, J+17, ASC(SEG$(A$(W), J, 1)))
630 NEXT J
640 INPUT B$
650 IF B$=A$(W) THEN 680
660 CALL SOUND(1000, -7, 1)
670 GOTO 700
680 CALL SOUND(1000, -2, 1)
690 SCORE=SCORE+1
700 IF K<>1 THEN 760
710 CALL HCHAR(23, Y-4, 98)
720 CALL HCHAR(23, Y+4, 98)
730 CALL HCHAR(23, Y-3, 99)
740 CALL HCHAR(23, Y+3, 100)
750 GOTO 790
760 IF K<>2 THEN 790
770 CALL HCHAR(23, Y-4, 99)
780 CALL HCHAR(23, Y+4, 100)
790 CALL HCHAR(23, Y-1, 105, 3)
800 CALL SOUND(1, 44000, 30)
810 CALL HCHAR(XC-1, Y-2, 98, 7)
820 CALL HCHAR(XC-1, Y+5, 32, 3)
830 CALL HCHAR(23, 1, 32, 15)
840 A$(W)="0"
850 NEXT K
860 CALL CLEAR
870 FOR I=2 TO 8
880 CALL COLOR(1, 2, 1)
890 NEXT I
900 SC=10*SCORE
910 PRINT :::::"YOUR SCORE IS"; SC, "PERCENT ACCURACY."
920 PRINT :::"PRESS ENTER TO CONTINUE."
930 CALL KEY(0, KEY, ST)
940 IF KEY<>13 THEN 930
950 GOTO 280
960 END
970 CALL CLEAR
980 CALL SCREEN(5)
990 T=500
1000 CALL SOUND(T, 880, 3, 698, 8, 294, 10)
1010 PRINT :::TAB(11); "T Y P I N G"
1020 CALL CHAR(104, "FFFF00FFFF00FFFF")
1030 CALL SOUND(T, 932, 3, 784, 8, 196, 11)
1040 PRINT :::TAB(15); "FOR"
1050 CALL COLOR(10, 16, 6)
1060 CALL COLOR(11, 13, 1)
1070 CALL SOUND(T, 784, 3, 659, 8, 262, 10)
1080 PRINT :::TAB(12); "ACCURACY"
1090 CALL CHAR(112, "00E0FBFEFFFFFFF")
1100 CALL SOUND(T, 880, 3, 698, 8, 175, 12)
1110 CALL CHAR(113, "00000000B0E0FBFE")
1120 CALL CHAR(114, "B0E0FCFFFFCE0B")
1130 CALL CHAR(115, "FEFB0E0B")
1140 CALL SOUND(T, 698, 3, 587, 8, 233, 10)
1150 PRINT :::TAB(15); "BY"
1160 PRINT :TAB(13); "REGENA"
1170 CALL CHAR(116, "FFFFFFFFFEFB")
1180 CALL SOUND(T/2, 784, 3, 165, 10)
1190 CALL CHAR(96, "B0C0E0FBFE0C0B")
1200 CALL SOUND(T/2, 698, 3, 165, 10)
1210 CALL CHAR(97, "B0C0E0F0FBFCFEFF")
1220 CALL SOUND(T/2, 659, 3, 277, 10)
1230 CALL CHAR(98, "FFFFFFFFFFFFFFFF")
1240 CALL CHAR(117, "FFFFFFFFFFFFFFFF")
1250 CALL SOUND(T/2, 784, 3, 277, 10)
1260 CALL CHAR(99, "FFFEFCFBFE0C0B")
1270 CALL SOUND(T*2, 698, 2, 587, 8, 147, 12)
1280 CALL HCHAR(15, 7, 117, 7)
1290 CALL HCHAR(16, 7, 117, 9)
1300 CALL HCHAR(17, 7, 117, 7)
1310 CALL HCHAR(16, 16, 114)
1320 CALL HCHAR(15, 15, 113)
1330 CALL HCHAR(17, 15, 115)
1340 CALL HCHAR(15, 14, 112)
1350 CALL HCHAR(17, 14, 116)
1360 CALL SOUND(T, 466, 4, 165, 10)
1370 FOR XX=15 TO 17
1380 CALL HCHAR(XX, 1, 104, 6)
1390 NEXT XX
1400 CALL SOUND(T, 440, 3, 175, 10)
1410 CALL COLOR(2, 16, 1)
1420 FOR YY=18 TO 26 STEP 2
1430 CALL HCHAR(16, YY, 42)
1440 NEXT YY
1450 CALL SOUND(T, 698, 3, 440, 8, 294, 10)
1460 CALL HCHAR(16, 28, 49)
1470 CALL HCHAR(16, 29, 48, 2)
1480 CALL HCHAR(16, 31, 37)
1490 CALL SOUND(T, 784, 3, 587, 8, 233, 10)
1500 CALL CHAR(100, "7F3F1F0F070301")
1510 CALL CHAR(101, "000103070F1F3F7F")
1520 CALL SOUND(T/2, 698, 3, 392, 8, 262, 10)
1530 CALL CHAR(102, "00001B1B3C3C7EFF")
1540 CALL SOUND(T/2, 659, 2, 262, 10)
1550 CALL CHAR(105, "DBDBDBDBDBDBDBDB")
1560 CALL SOUND(4*T, 698, 2, 440, 8, 175, 10)
1570 CALL SOUND(1, 44000, 30)
1580 RETURN
1590 CALL CLEAR
1600 CALL SCREEN(7)
1610 CALL COLOR(2, 2, 1)
1620 PRINT "PICK A TYPING CATEGORY."
1630 PRINT :::"YOU WILL SEE A WORD"
1640 PRINT "IN THE ROCKET."
1650 PRINT : "TYPE AND ENTER IT BEFORE
    THE TONE ENDS."
1660 PRINT : "IF YOU ARE CORRECT,"
1670 PRINT "ANOTHER TONE SOUNDS,"
1680 PRINT : "IF YOU ARE INCORRECT."
1690 PRINT "YOU WILL BE BLASTED." :::
1700 PRINT "PRESS ENTER TO CONTINUE."
1710 CALL KEY(0, KEY, ST)
1720 IF KEY<>13 THEN 1710
1730 RETURN
1740 CALL CLEAR

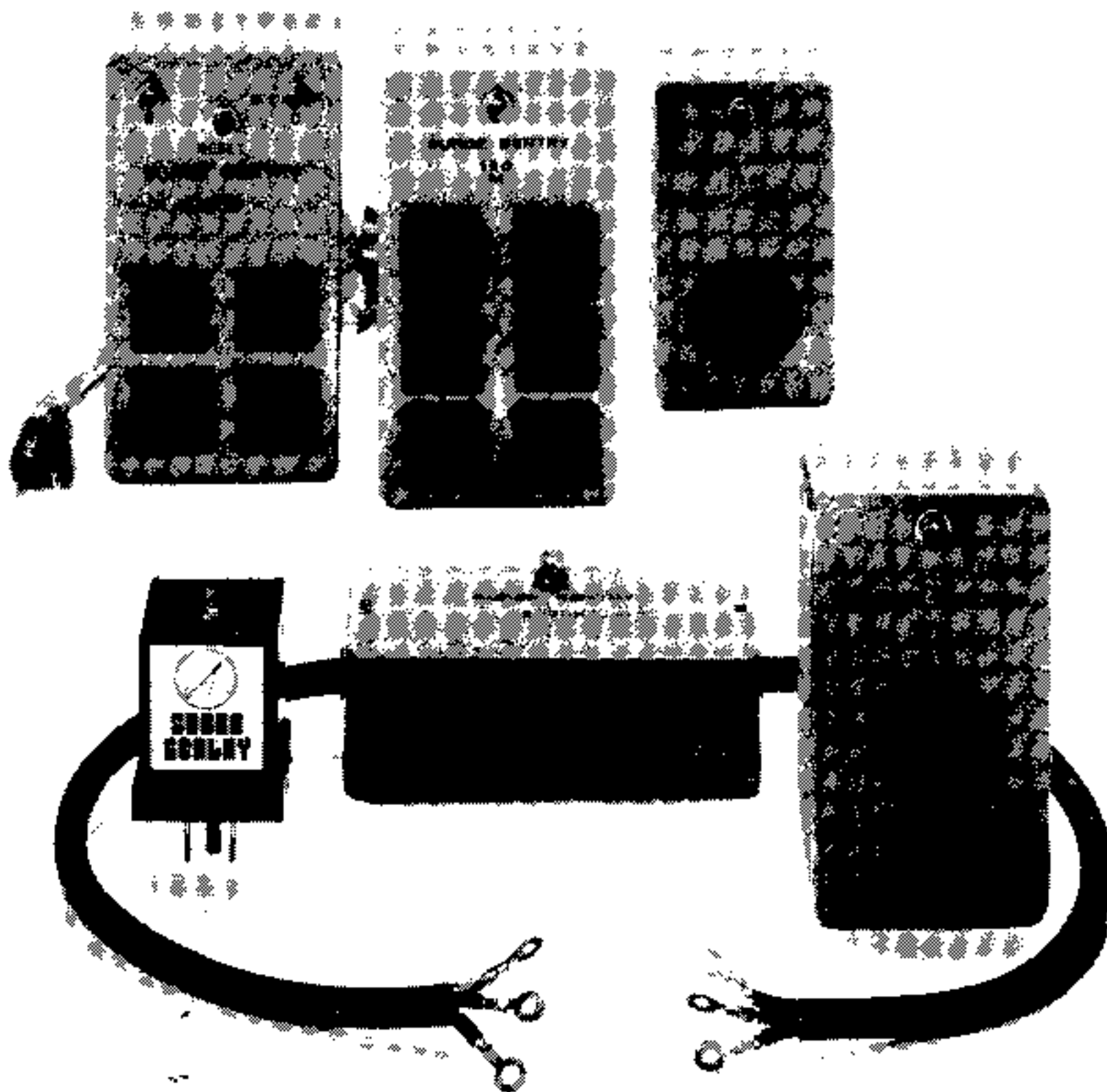
```

KILL SURGES LIKE LIGHTNING!

AC power line surges are destructive, can cost you money, and can't be prevented. But you can stop them from reaching your sensitive electronic equipment with a Surge Sentry.

Surge Sentry acts in picoseconds to dissipate up to a 1,000,000 W, 100 μ second surge. Triggers at 10% above nominal peak voltage. Works in parallel with the power line. Is easy to install for immediate protection. No complicated wiring or special tools required.

Several different models to choose from, including an OEM version. Call or write today for a free brochure.



**RKS
ENTERPRISES, INC.**

643 South 6th Street, San Jose, CA 95112
(408) 288-5565

DEALER INQUIRIES INVITED

Typing Tutor . . . from p. 81

```

1750 CALL SCREEN(12)
1760 CALL COLOR(1,2,12)
1770 FOR I=2 TO 8
1780 CALL COLOR(I,1,12)
1790 NEXT I
1800 PRINT : : : : " CHOOSE ONE" : :
1810 PRINT : " 1 HOME KEYS"
1820 PRINT : " 2 HOME ROW"
1830 PRINT : " 3 TOP ROW, MIDDLE FINGER"
1840 PRINT : " 4 TOP ROW, POINTER FINGER"
1850 PRINT : " 5 RING FINGER"
1860 PRINT : " 6 LITTLE FINGER"
1870 PRINT : " 7 BOTTOM ROW"
1880 PRINT : " 8 END PROGRAM" : : :
1890 CALL SCREEN(5)
1900 FOR I=2 TO 8
1910 CALL COLOR(I,2,12)
1920 NEXT I
1930 RETURN
1940 CALL COLOR(9,1,1)
1950 CALL COLOR(10,1,1)
1960 CALL VCHAR(12,Y,98,13)
1970 CALL VCHAR(13,Y-1,98,12)
1980 CALL VCHAR(13,Y+1,98,12)
1990 CALL VCHAR(14,Y-2,98,11)
2000 CALL VCHAR(14,Y+2,98,11)
2010 CALL VCHAR(13,Y-2,101)
2020 CALL VCHAR(13,Y+2,97)
2030 CALL VCHAR(12,Y-1,101)
2040 CALL VCHAR(12,Y+1,97)
2050 CALL VCHAR(11,Y,102)
2060 CALL VCHAR(22,Y-3,98,3)
2070 CALL VCHAR(22,Y+3,98,3)
2080 CALL VCHAR(23,Y-4,98,2)
2090 CALL VCHAR(23,Y+4,98,2)
2100 CALL VCHAR(22,Y-4,101)
2110 CALL VCHAR(22,Y+4,97)
2120 CALL VCHAR(21,Y-3,101)
2130 CALL VCHAR(21,Y+3,97)
2140 CALL COLOR(9,7,1)
2150 CALL COLOR(10,16,1)
2160 RETURN
2170 END

```

Tax Laws . . . from p. 56

law as under the old, but owners of desktop size computers will be better off—assuming no other equipment purchases in the year.

If the tax deductions won't be available because of other tax deductions or business losses, computer owners will be able to elect to write the equipment off over a 12 year or a 25 year period using a straight line method of depreciation. However, the election to use the slower method is mandatory for each years purchases—meaning you can't change your mind after a year or two. The main reason to use a slow method of depreciation is to avoid the possible loss of deductions during a prolonged start up period due to the existing time limit on offsetting losses of one year against profits of future years. The new law provides substantial relief in this area—which may make the slower depreciation method unnecessary. Previously, business losses could be carried forward for seven years, but the new law extends this to 15 years—retroactive to 1976.

Changes In Rules For Investment Tax Credit

Computer buyers will realize a small increase in the amount of available investment tax credit for purchasing a computer. Under current law, equipment with a five year useful life is eligible for 2/3 of the full 10% tax credit. Equipment with a five year life will now be qualified to claim the full 10% tax credit for equipment that is depreciated over a period of five or more years. If the equipment will have a three year useful life (autos, trucks and certain R & D equipment), the tax credit will be 6% of the cost of the property rather than 10%. These new tax credit rules take effect in 1981 including property that was acquired before the law was passed. (It was signed on 8/13/81.)

There was no specific change relative to claiming the tax

Continued on p. 83

Now What? . . . from p. 36

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

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

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

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

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

Income Tax

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

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

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

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

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



Tax Laws . . . from p. 82

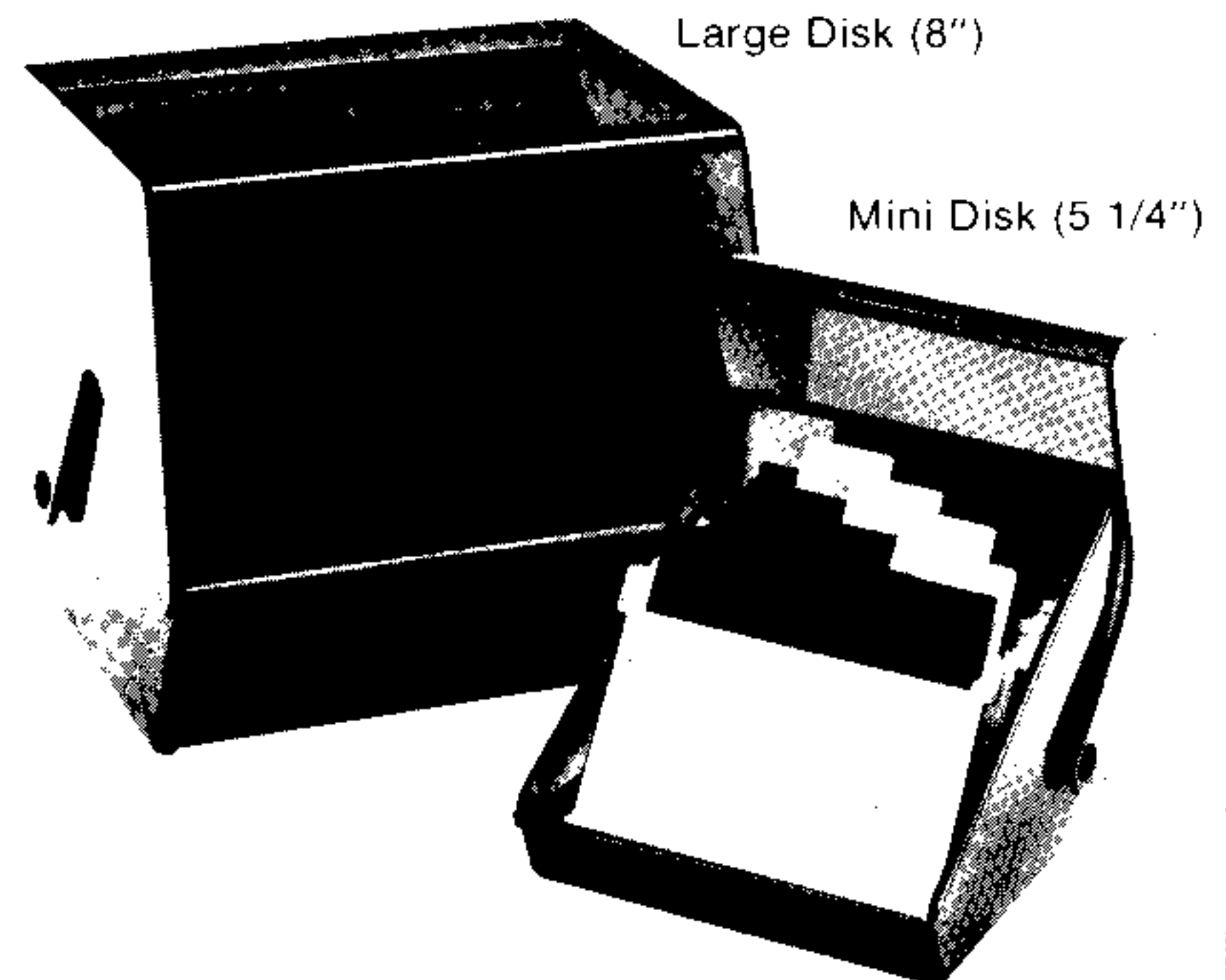
credit on the full cost of a system that included both hardware and software. However, if the tax credit is claimed on the software because the price is combined with the hardware, then the buyer must depreciate the software with the hardware. If the software is purchased separately, and is licensed rather than purchased, then the full software cost can be deducted in the year of acquisition.

New Rules For Defining Leases

Taxpayers and the IRS have been arguing for years about whether a lease is really a lease or just a method of financing an equipment purchase. The new tax law attempts to simplify some of the complex rules that have cropped up in this area of controversy. *Basically*, the parties must clearly agree that the transaction is a lease and the lessee must not acquire ownership of the property at any time during the lease. The lessor must be a corporation and must have an investment of at least 10% that is "at risk" in the investment. Generally, the property must be new property.



For the Intelligent user.



Innovative Concepts, Inc.'s **Flip 'N' File** storage system for flexible disks is both functional and attractive. **The Flip 'N' File** provides your disks with the protection they require.

- **Library Concept**
Easy filing, easy retrieval, ideal storage system.
- **Index Dividers**
Provide easy retrieval—adjustable tabs can be set to desired positions (additional dividers may be ordered separately).
- **Dust Free Protection**
Self contained storage units protect your disks from dust that can lead to loss of important information.
- **Durability**
Each unit is molded from a highly durable plastic.
- **Unique Design**
Transparent, smoke colored, durable plastic is elegant in its simplicity, and enhances any decor.
- **Storage Capacity**
Each unit has a storage capacity of 50-60 disks. The unique lid design provides for easy access and doubles as a carrying handle.

Creators and Manufacturer's of Flip 'N' File and other Fine Products.

IC Innovative Concepts Inc.

2284 Ringwood Avenue, San Jose, California 95131
(408) 262-6680

Now Available Simulation Games for the TI-99/4

- **TI Nuclear Power Plant**
(Extended BASIC 16K) \$24.95
Run a nuclear reactor from the comfort of your own home.
- **Martian Lander**
(Extended BASIC 16K) \$9.95
Landing on the Martian surface with limited fuel is a real challenge.
- **Medieval World**
(Extended BASIC 16K) \$9.95
Increase your wealth and become King while fighting off enemies.

Holiday Special: All 3 for \$34.95

For additional information, write or call:
Galactic Software
P. O. Box 99345
Cleveland, OH 44199
Tel. (216) 241-5100
Ext. 907/908

SOFTWARE SERVICES TI-99/4

CUSTOM PROGRAMMING

- Specializing in Tutorials for the Elementary Grades
- Personalized Consulting for the Compleat Novice
- Down-to-Earth Prices

For more information write:

Pablo Diablo*
P. O. Box 4863
Santa Clara, CA 95054

SOURCE ID TCV774

*The Legendary Evil Kiteflyer

MICROGRADE I&II

★ Revolutionizes Grade Keeping ★

Features :

- Change in Entry
- Delete an Entry
- Alphabetize
- Individual or Group Profile
- Selected Listings
- Hardcopy

Extended BASIC Required

MICROGRADE I (disk) \$26.95

MICROGRADE II (disk) \$59.95

(Version II requires 32K Memory Exp.)

To Order, or For Detailed Catalog, Write:

Charles Morreale
105 Meadowbrook
Venetia, PA 15367

PROGRAMMING HINTS



HINTS... BY REGENA

WHO IS REGENA???

Twice each month, like clockwork, a peculiar event occurs. An air freight delivery truck rolls up to our door and drops off a mysterious parcel. Each time, the outside wrapper carries markings from a different country. One thing, however, always remains the same—the two-word return address: "From Regena."

The contents of each package are similar—manuscripts and tapes of articles and programs for and about the TI-99/4 computer, plus a one-line note: "Hope you can use this—Regena." We've never been able to contact this mysterious programmer. Since we presume that he or she gets to see the magazine, it would seem logical that Regena must be a subscriber. But we can't be certain, as large numbers of issues also get sent in bulk to dealers and distributors all over the U. S. and abroad.

Also, there's the matter of the formal letter we received one day from a Zurich bank, with instructions to mail Regena's payment checks to a numbered Swiss account. All very secret and strange indeed . . .

Here at the magazine, we've tried to figure it out: Why would anyone who goes to all this trouble to keep their identity and whereabouts such a secret be writing for a computer magazine? Unless, of course . . . But no, that's the stuff that spy thrillers and James Bond movies are made of . . .

Regena's manuscripts (each typed on a different machine, on paper bearing different watermarks) and program tapes (long strips of reel-to-reel recording tape, rolled up tightly and inserted into 35 mm film cans) have been thoroughly analyzed for clues. It is true, we have found a few, but we suspect, however, that they were deliberate "red herrings." Perhaps you, our readers, have detected some clues in the many articles and programs of Regena that we've published. Or perhaps one of you has actually made contact with our mysterious programmer. In any case, please send us your ideas or information.

And Regena—if you're reading these words—won't you please "come in from the cold . . ."

1: Graphic String Display

Suppose you have a variable you would like to print during a graphics display. CALL HCHAR or VCHAR allows you to print only one character at a time at a specific location. If the value to be printed contains a variable number of digits, then let the computer do the work by using this routine.

```
10 INPUT A
20 A$=STRS(A)
30 FOR I=1 TO LEN(A$)
40 CALL HCHAR(12,I+5,ASC(SEGS(A$,I)))
50 NEXT I
60 GOTO 10
```

Line 10 "A" is the number you want printed and ordinarily would be calculated by your program. For this test program you may key in a trial number.

Line 20 Let A\$ be the string variable of A.

Line 30 Starting with 1 go to the length of A\$ (that is, the number of digits).

Line 40 Here is the CALL HCHAR(x,y,c) statement. This program will start printing in the 12th row and 6th column. ASC finds the ASCII code of the digit. SEG\$ is picking up the Ith digit of A\$ one at a time.

Line 50 Increment I until you have all the digits.

Line 60 Try another number.

#2: Real Time Delay

You may want a delay for a specific length of time in your program. An accurate way to do this is to use CALL SOUND for the number of milliseconds you need. Use 30 for the volume level and a very high frequency so you won't hear anything. For example, if you have a game in which the rocket is blasting off and you want a countdown for each second, use:

```
FOR I=10 TO 1 STEP -1
CALL SOUND(1000,44000,30)
PRINT I
NEXT I
CALL SOUND(1,44000,30)
PRINT "BLASTOFF"
```

While a SOUND is being performed, the program can be doing graphics or computations. As soon as another SOUND statement is encountered the computer waits for the previous one to be executed for its duration. The duration of 1 for the last SOUND statement signals the end of the previous second and allows the program to continue. It forces the computer to wait the final second before "BLASTOFF" is printed.

#3: RND Function . . . How Fair Is RANDOM?

The RND function is used to choose random numbers. It can be used, for example, to simulate the throw of dice, choosing random numbers from one through six. But does one number seem to be picked more often than another?

This program was run to pick a random number from 1 through 10 ten thousand times and to count how many times each number was chosen. The probability for each number being chosen is 1/10, so for 10,000 trials each number should be picked 1000 times—if RND is fair.

```
10 DIM N(10)
20 RANDOMIZE
30 FOR I=1 TO 10000
40 J=INT(10*RND)+1
50 N(J)=N(J)+1
60 NEXT I
70 FOR I=1 TO 10
80 PRINT I;N(I)
90 NEXT I
100 END
```

The results of four runs are as follows. N is the number chosen randomly. The numbers in the columns for each run are the number of times N was chosen.

N	First Run	Second Run	Third Run	Fourth Run
1	1002	1008	1008	1002
2	993	997	992	989
3	1036	1035	1007	1019
4	1025	1037	1028	1034
5	980	984	989	986
6	1011	1007	1002	1020
7	1005	982	1002	1011
8	1025	1020	1029	1024
9	966	959	987	956
10	957	971	956	959

The distribution is fairly even and within 5 percent of 1000 for each number picked. The number chosen the fewest number of times was chosen 956 times, and the highest a number was chosen was 1037. It is interesting to note, however, that for these four runs, the same numbers were chosen more often or less often in each run as in each other run, with the exception of N=7 in the second run.



TI-Asteroids

- 99/4 version of Popular Arcade Game
- Finest Extended BASIC Game Available
- Rated ****Four Stars**** By National Users Group
- Four Skill Levels
- "High Score To Date" Feature in Both Versions

Cassette Version \$22.95

Diskette Version \$25.95

(NJ residents add 5% tax)

To order or for information write :

FFF SOFTWARE

P. O. Box 4169

Trenton, NJ 08610

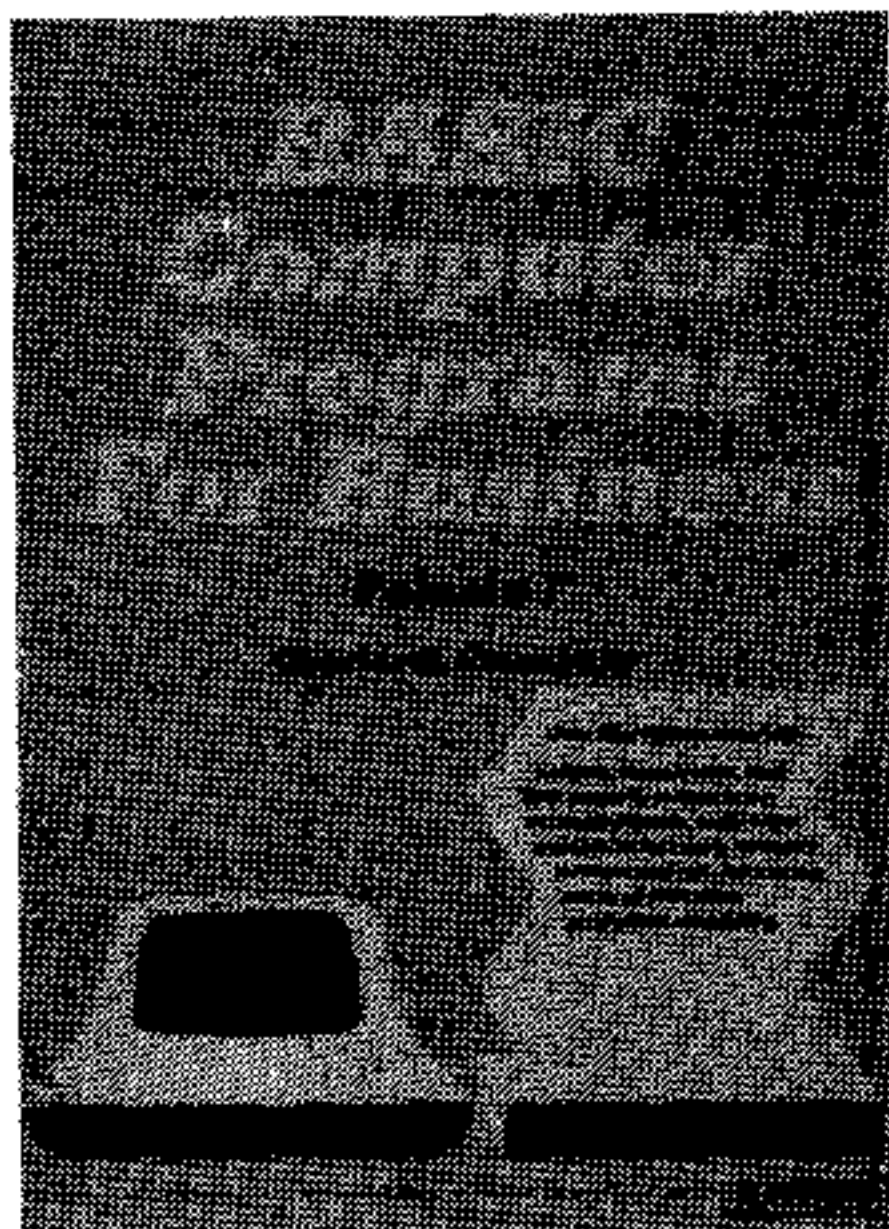
Looking for a Gift?



**The
99'er
Bookstore**

(pages 86 & 87)

99'er BOOKSTORE

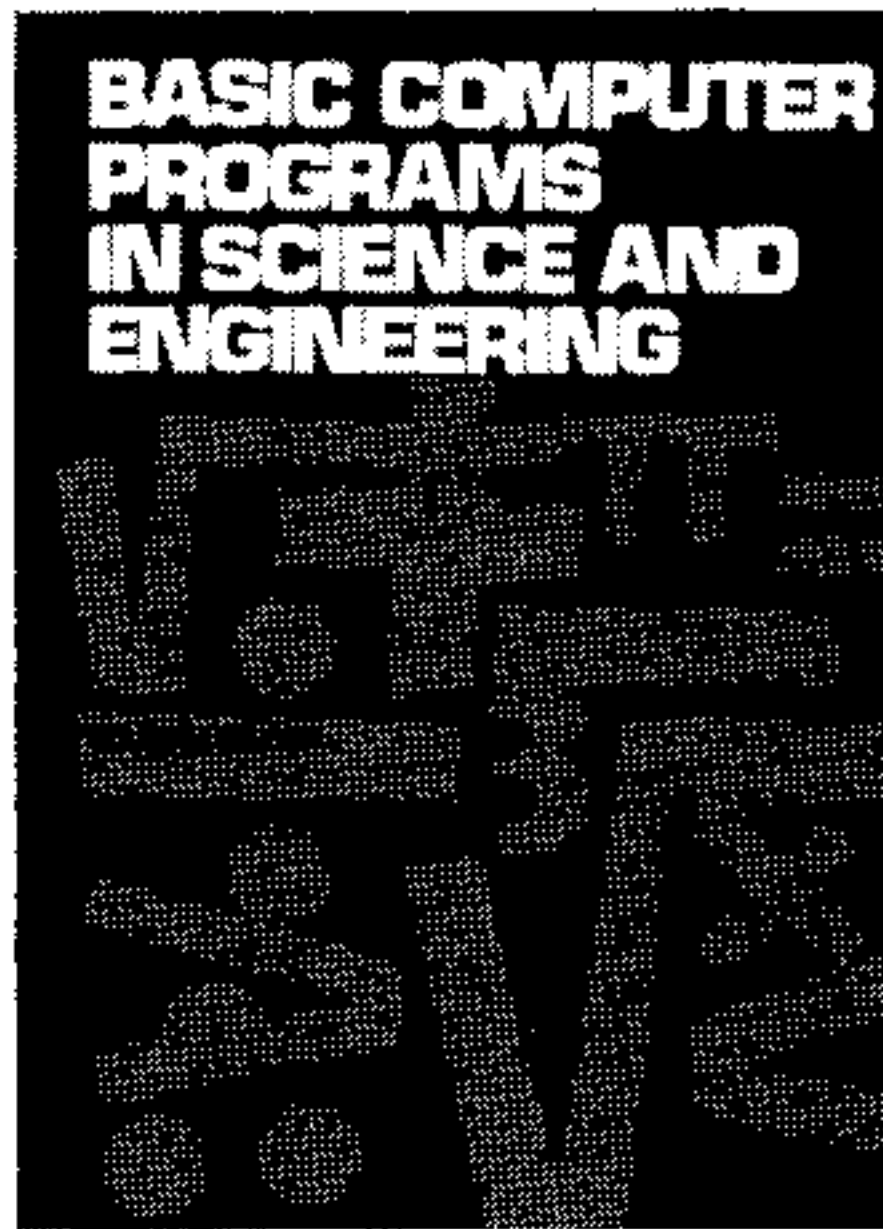


BASIC COMPUTER PROGRAMS FOR BUSINESS : VOL. 1

By Charles D. Sternberg.
A must for small businesses utilizing micros as well as for entrepreneurs, these two volumes provide a wealth of practical business applications. Each program is documented with a description of its functions and operation, a listing in BASIC, a symbol table, sample data, and one or more samples.

Volume 1 contains over 35 programs covering: budgets, depreciation, cash flow, property comparisons, accounts payable, order entry, warehouse locations, inventory turnover analysis, job routing, resource allocation, production scheduling, etc.

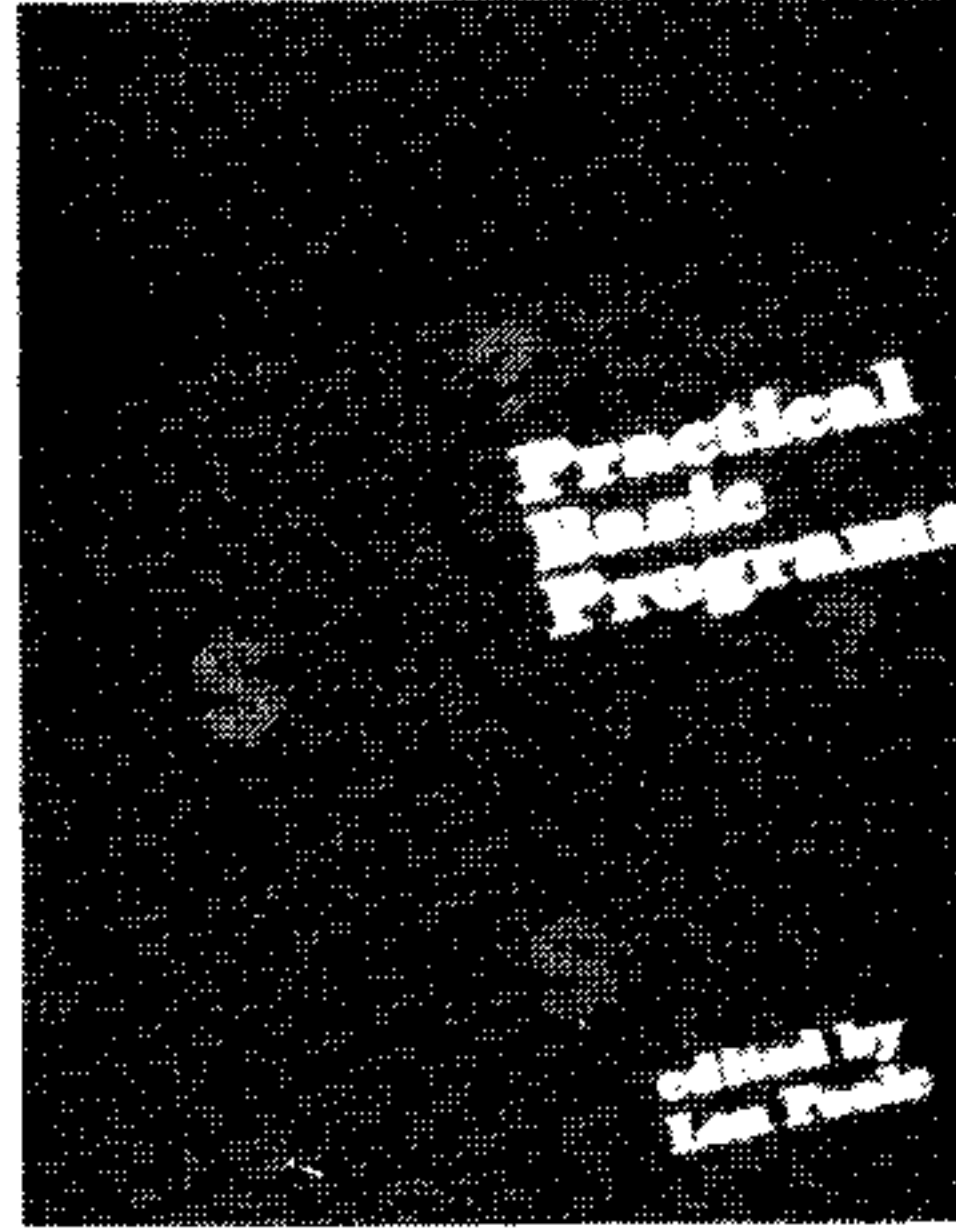
volume 1, paper, \$10.95
1980, 384 pages, 7 x 10



BASIC COMPUTER PROGRAMS IN SCIENCE AND ENGINEERING

By Jules H. Gilder.
Save time and money with this collection of 114 ready-to-run BASIC programs for the hobbyist and engineer. There are programs to do such statistical operations as means, standard deviation averages, curve-fitting, and interpolation. There are programs that design antennas, filters, attenuators, matching networks, plotting, and histogram programs. There is even a justified typing program that can be used in typesetting. All programs in the book have been tested and are fairly universal; so you should have no difficulty running them on your system. You won't find anywhere a more comprehensive collection of usable, ready-to-run BASIC programs!

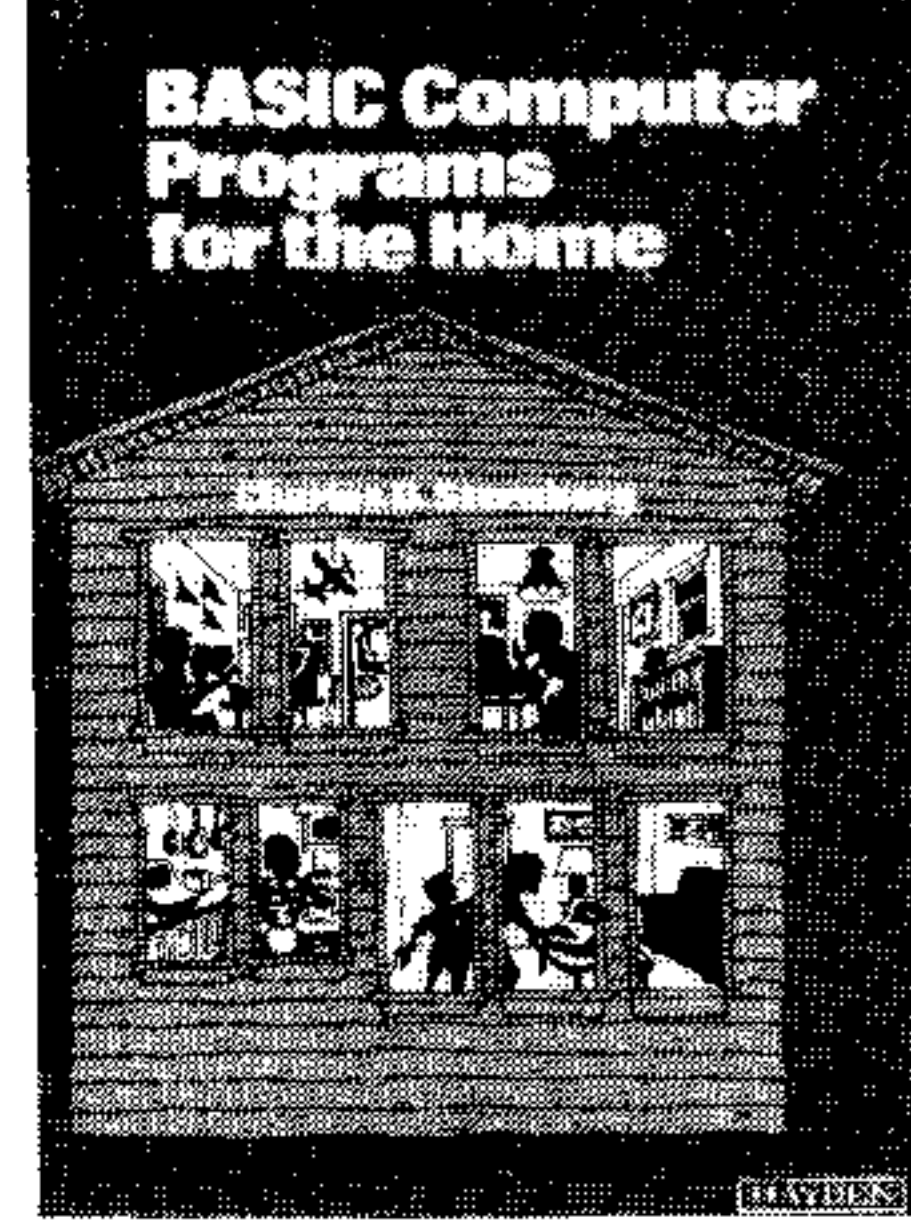
paper, \$9.95
1980, 160 pages, 6 x 9, illus.



PRACTICAL BASIC PROGRAMS

Edited by Lon Poole
Here is a new collection of 40 programs you can easily key in and use on most microcomputers. Each program does something useful. *Practical BASIC Programs* is especially useful in small business applications. It solves problems in finance, management decision, mathematics and statistics. It requires no prior programming knowledge. Each program is thoroughly documented. The book contains sample runs, practical problems, BASIC source listings, and an easy to follow narrative to help you realize the potential uses of each program. This book is a valuable reference for anyone who needs a wide range of useful programs: income averaging, present value of a tax shield, lease/buy decision, financial statement ratio analysis, checkbook reconciliation, home budgeting, non-linear breakeven analysis, Program Evaluation and Review Technique (PERT), statistics, data forecasting divergence, musical transposition, Bayesian decision analysis, etc.

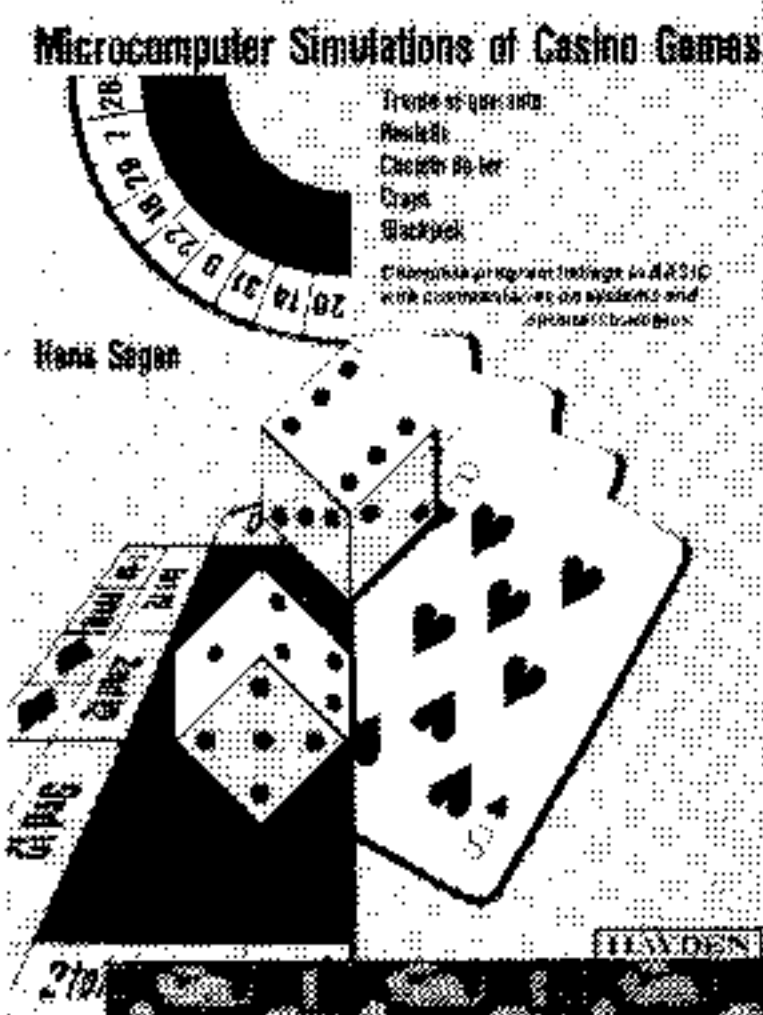
paper, \$15.99
1980, 200 pages, 8 1/2 x 11



BASIC COMPUTER PROGRAMS FOR THE HOME

By Charles D. Sternberg.
An invaluable book containing over 75 practical home application programs that will be helpful to the novice or experienced owner in increasing the usefulness of any home computer. Each program is documented with a description of its functions and operation, a listing in BASIC, a symbol table, sample data, and one or more samples. Programs included are: Home Financial Programs; Automobile Related Programs; Kitchen Helpmates; Scheduling Programs for Home Use; List Programs for Every Purpose; Miscellaneous Programs for the Home; Tutorial Programs for Home Use; Conversion Program; and Hobbyist's Diaries.

paper, \$9.95
1979, 336 pages, 7 x 10



BEAT THE ODDS: MICRO-COMPUTER SIMULATIONS OF CASINO GAMES

By Hans Sagan.
Here's an extremely useful programming guide that provides realistic simulations of five popular Casino games: Trente-et-Quarante (Thirty and Forty), Roulette, Chemin-de-Fer, Craps, and Blackjack. Each of the five chapters has the same structure. It begins with a computer run, displaying facets of the programs, followed by an explanation of the objectives and the physical execution of the game. Acceptable bets and how to place them are discussed and systems and/or strategies laid out. Finally, the computer program is developed and various modifications of the program are detailed.

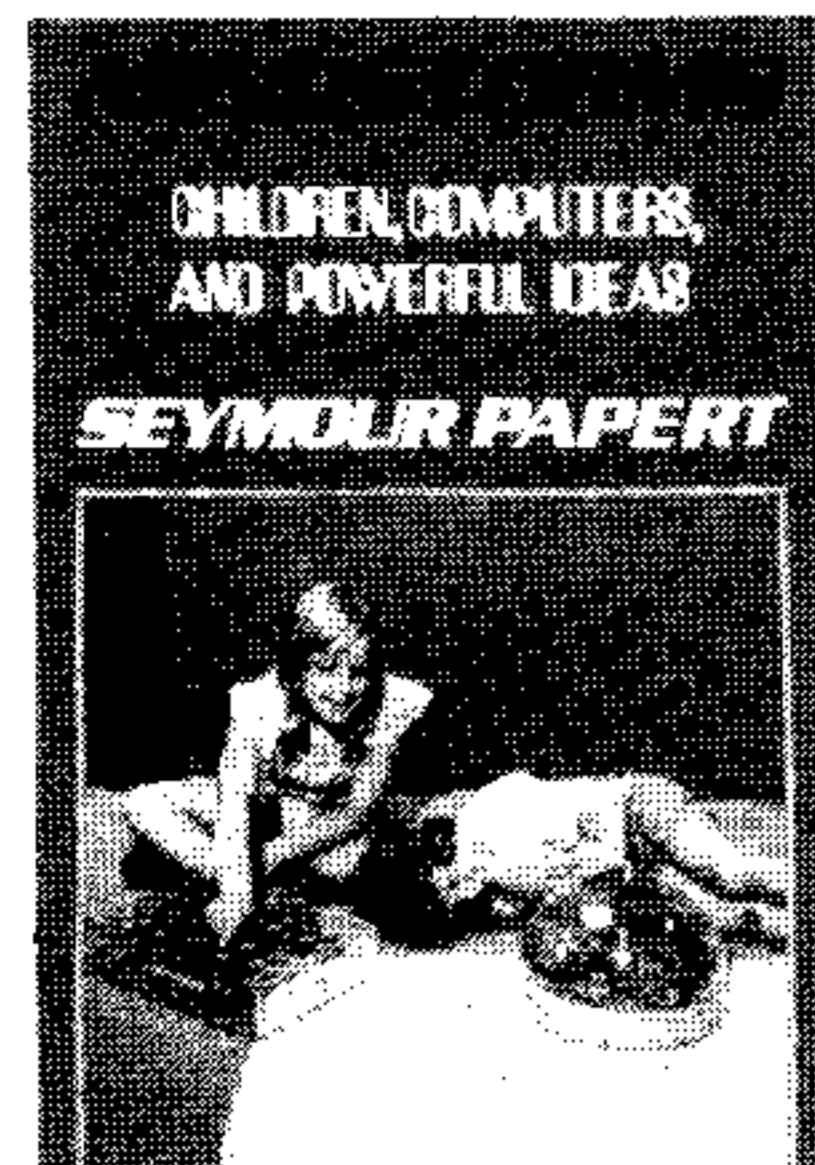
All programs are written in BASIC and heavily REM'd for readability and conversion. A comprehensive bibliography, a glossary of French gambling terms and phrases, and hints on the discrepancies between BASIC dialects are included, as well as a summary of maxims of probability theory.

paper, \$7.95
1980, 128 pages, 6 x 9

GAME PLAYING WITH BASIC

By Donald D. Spencer.
Enjoy the challenge of competition with your computer. Amuse yourself with such games and puzzles as 3-D Tic-tac-toe, Nim, Roulette, Magic Squares, the 15 Puzzle, Baccarat, Knight's Magic Tour, and many others. The writing is nontechnical, allowing almost anyone to understand computerized game playing. The book includes the rules of each game, how each game works, illustrative flowcharts, diagrams, and the output produced by each program. The last chapter contains 26 games for reader solution.

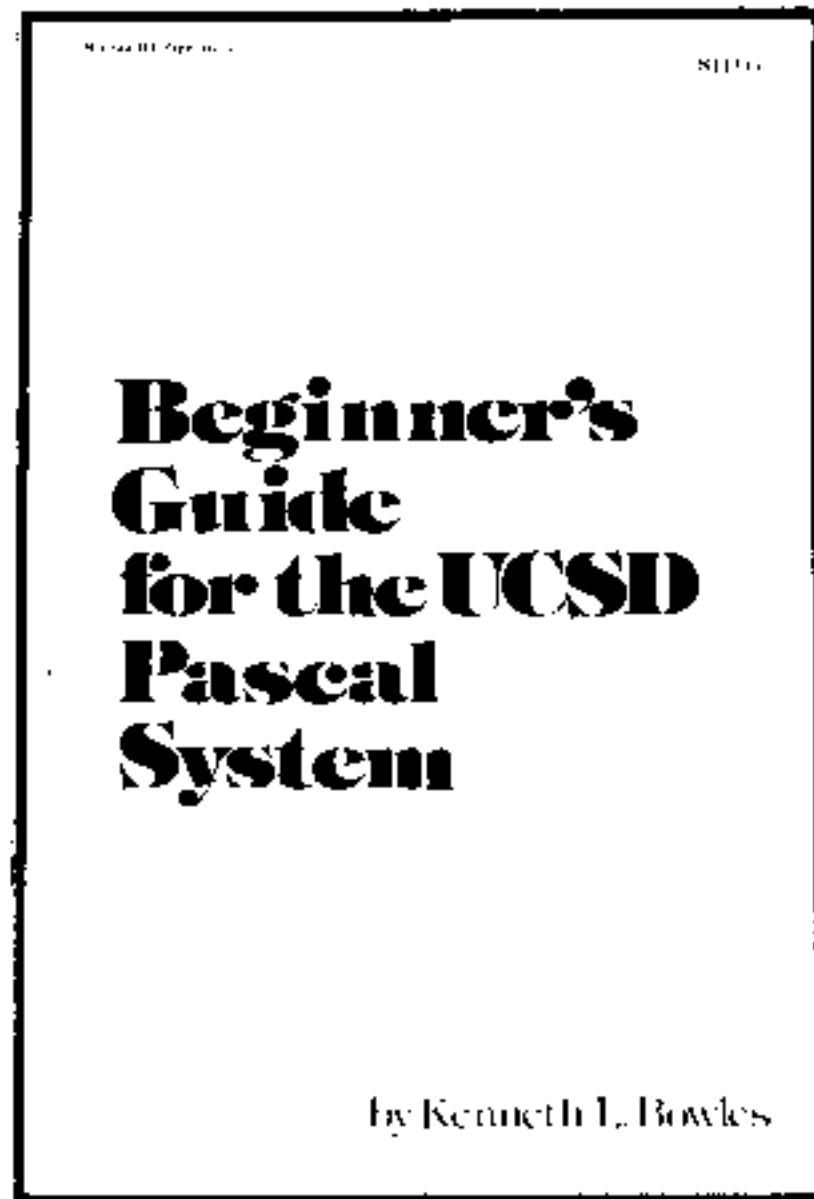
paper, \$9.55
1977, 176 pages, 6 x 9, illus.



MINDSTORMS: CHILDREN, COMPUTERS AND POWERFUL IDEAS

By Seymour Papert
The definitive work on the philosophy behind LOGO. Excerpted in the May/June issue of this magazine.
Hardcover, \$12.95
1980, 230 pages, 6 x 9

99'er BOOKSTORE

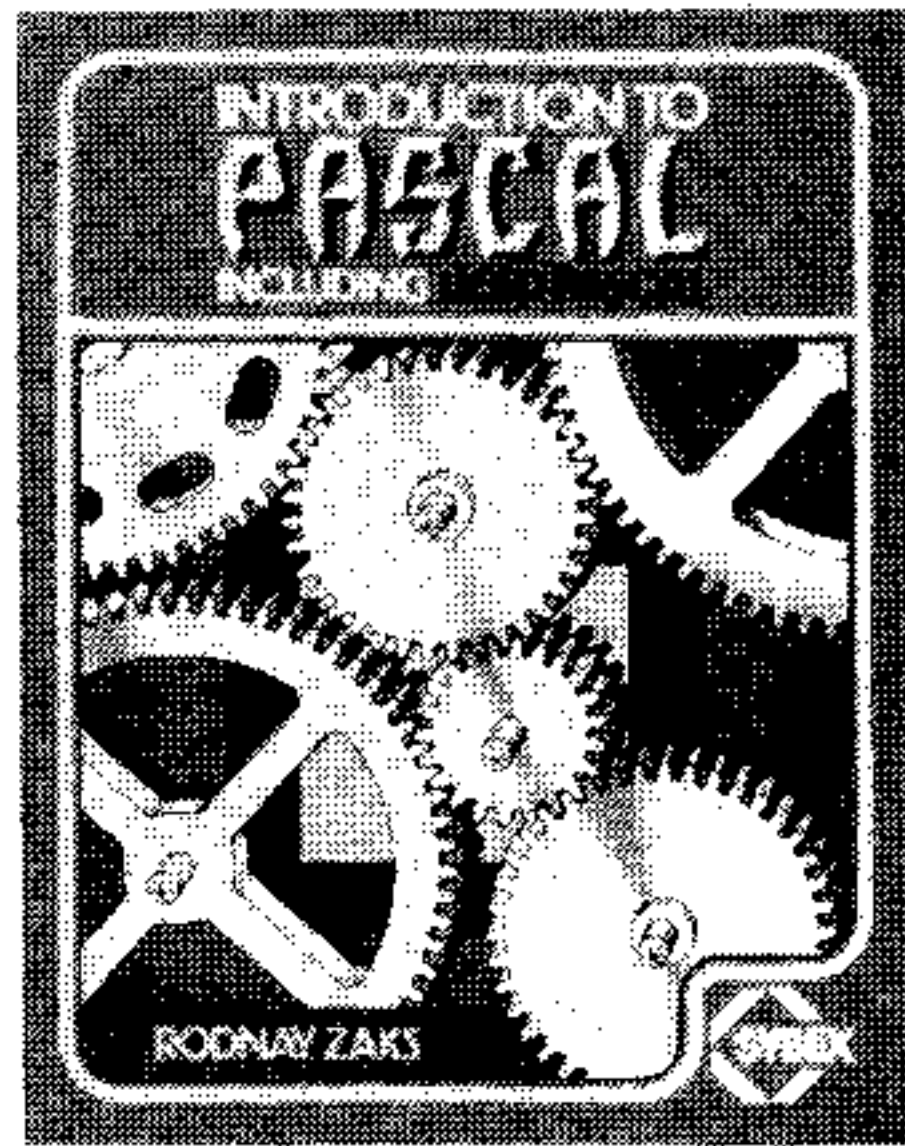


BEGINNER'S GUIDE FOR THE UCSD PASCAL SYSTEM

By Kenneth Bowles

This highly informative book is written by the originator of the UCSD Pascal System. It is designed as an orientation guide for learning to use the UCSD Pascal System, and features tutorial examples of programming tasks in the form of self-study quiz programs. Once familiar with the system you will find the guide an invaluable reference tool for creating advanced applications.

paper, \$11.95
1980, 204 pages, 6 x 9

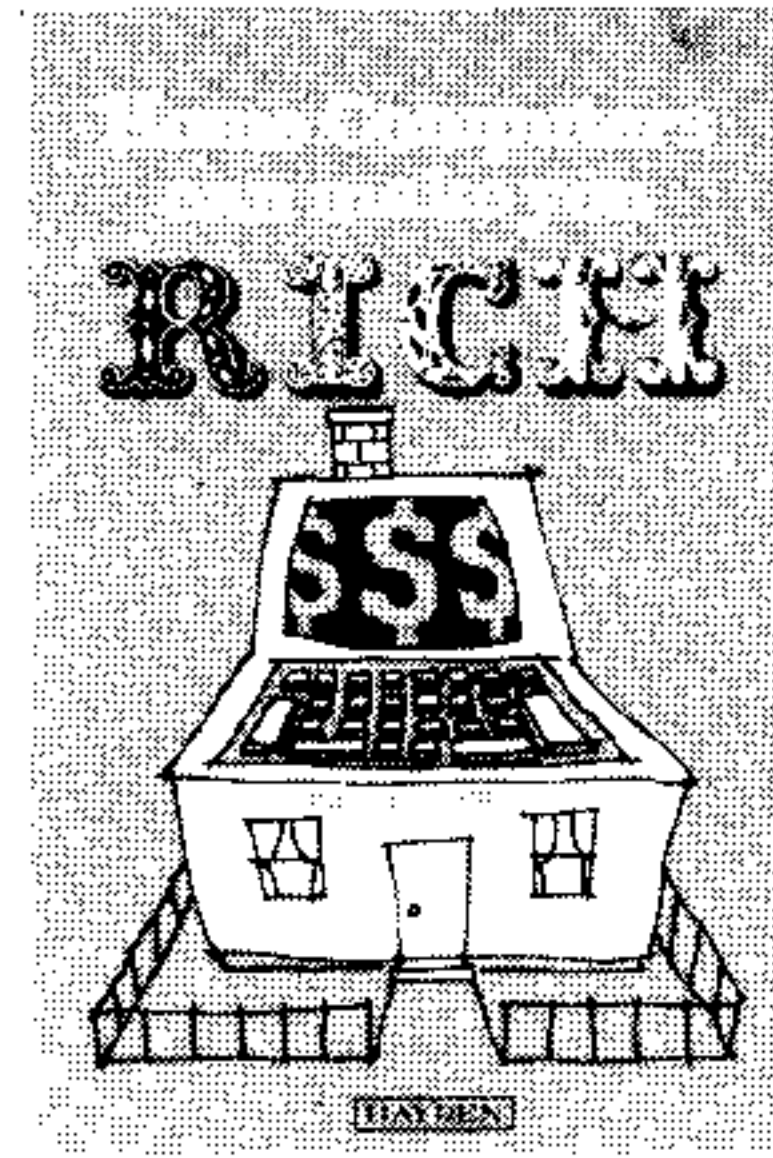


INTRODUCTION TO PASCAL (INCLUDING UCSD PASCAL)

By Rodnay Zaks

This is the first book on Pascal that can be used by persons who have never programmed before, but more generally it is a simple and comprehensive introduction to standard and UCSD Pascal for anyone—beginner to experienced programmer—who wants to learn the language rapidly. The logical progression and graduated exercises—designed to provide practice as well as test skill and comprehension—enable the reader to begin writing simple programs almost immediately. This book presents all concepts and techniques in a clear and simple style, making it accessible to beginners and useful to experienced programmers. All Pascal features are covered in detail, from basic definitions to complex data structures. An extensive appendix section presents a listing of all symbols, keywords and rules of syntax for programming in Pascal, providing a concise summary and important reference tool.

paper, \$14.95
1981, 440 pages, 7 x 9



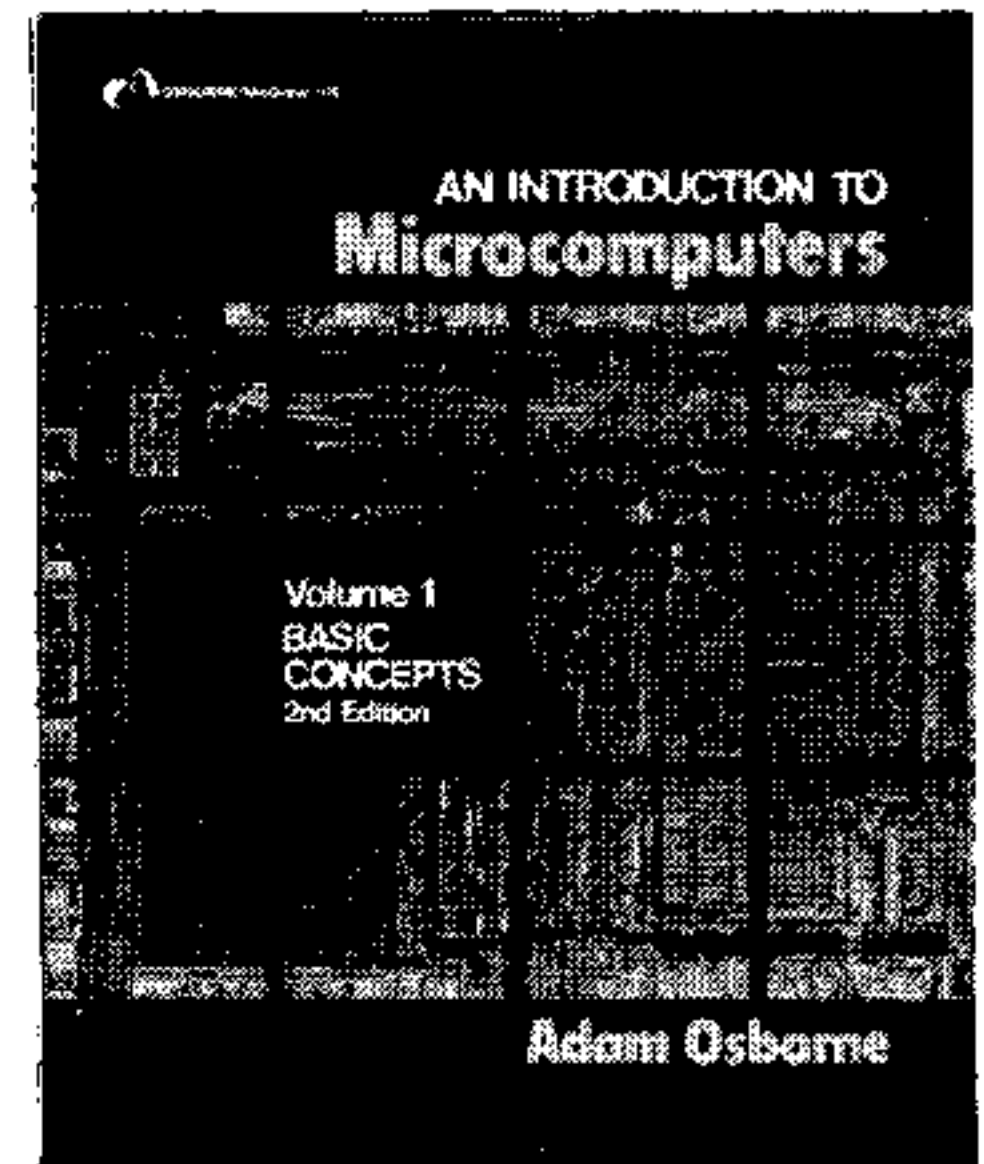
HOME COMPUTERS CAN MAKE YOU RICH

By Joe Weisbecker

Here's a valuable text for every home computer owner and non-owner interested in spare-time income opportunities. You'll be introduced to the microcomputer industry, and the types of people involved in it. You'll find out how to learn more about this new industry. Discussed are basic principles of making money, freelance writing, programming, consulting, inventing, computer-made products, investing, and much more.

CONTENTS: The Microcomputer Industry. What you Need to Know About Making Money. Resources You Can Use. Choosing Your Hardware. Writing for Money. Creating and Selling Programs. Services for Sale. Use Your Imagination. Invent Your Way to Success. Making Your Money Grow. Working at Home.

paper, \$6.50
1980, 128 pages, 6 x 9



AN INTRODUCTION TO MICROCOMPUTERS - VOLUME 1 - BASIC CONCEPTS

By Adam Osborne

Using concepts that are common to all microprocessor systems, *Volume 1* develops a detailed picture of what a microcomputer can do, how it does what it does, and how the capabilities of microcomputers can best be applied. Basic Concepts presents the fundamental logic framework upon which microcomputer systems are built, so that the reader can evaluate the applicability of microcomputers to any practical problem. This new revised edition incorporates all recent microprocessor developments. Concepts are discussed in terms of modern hardware configurations, and examples of common applications are drawn from today's most popular devices. For example, the logic instructions and programming concepts of the new 16-bit microprocessors are discussed in detail, and current logic distribution configurations are used throughout the text with numerous illustrations and examples. Programming mnemonics conform to the newly proposed IEEE standard. This is the first book in print to use them.

paper, \$12.99
1980, 320 pages, 7 x 9

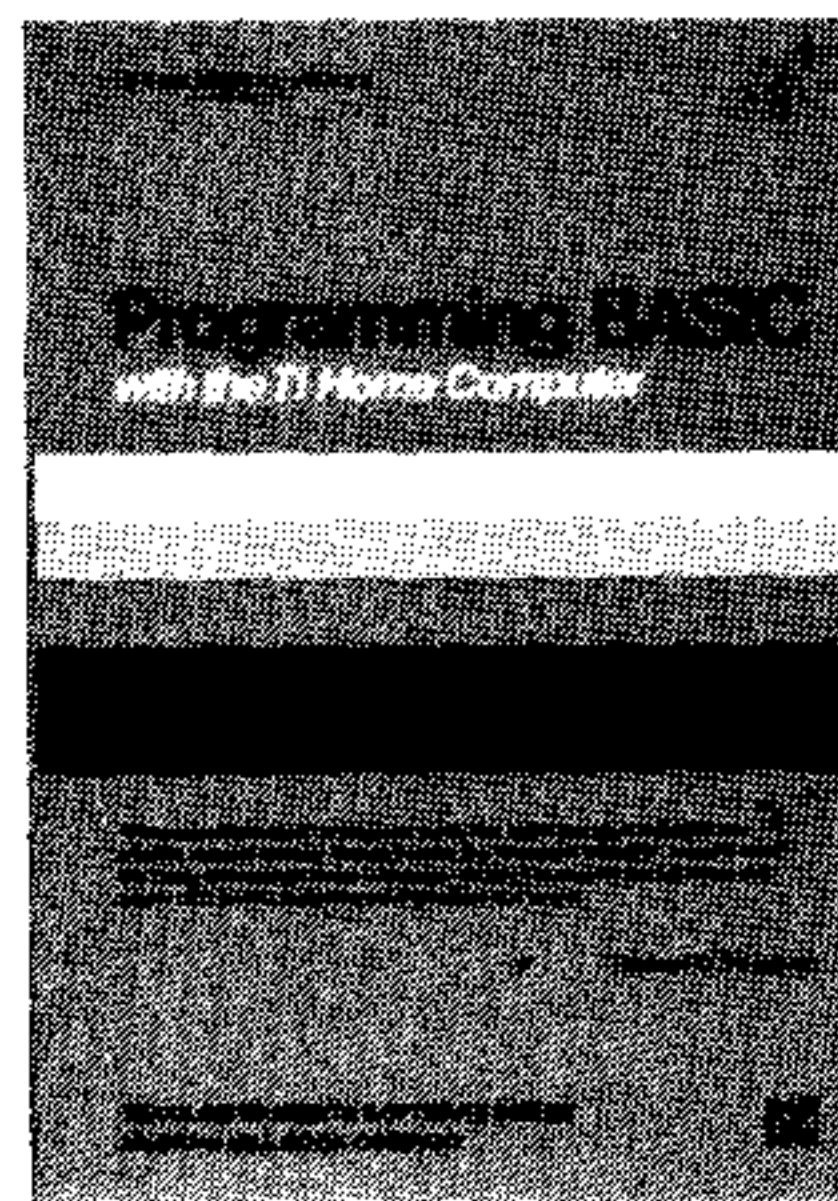
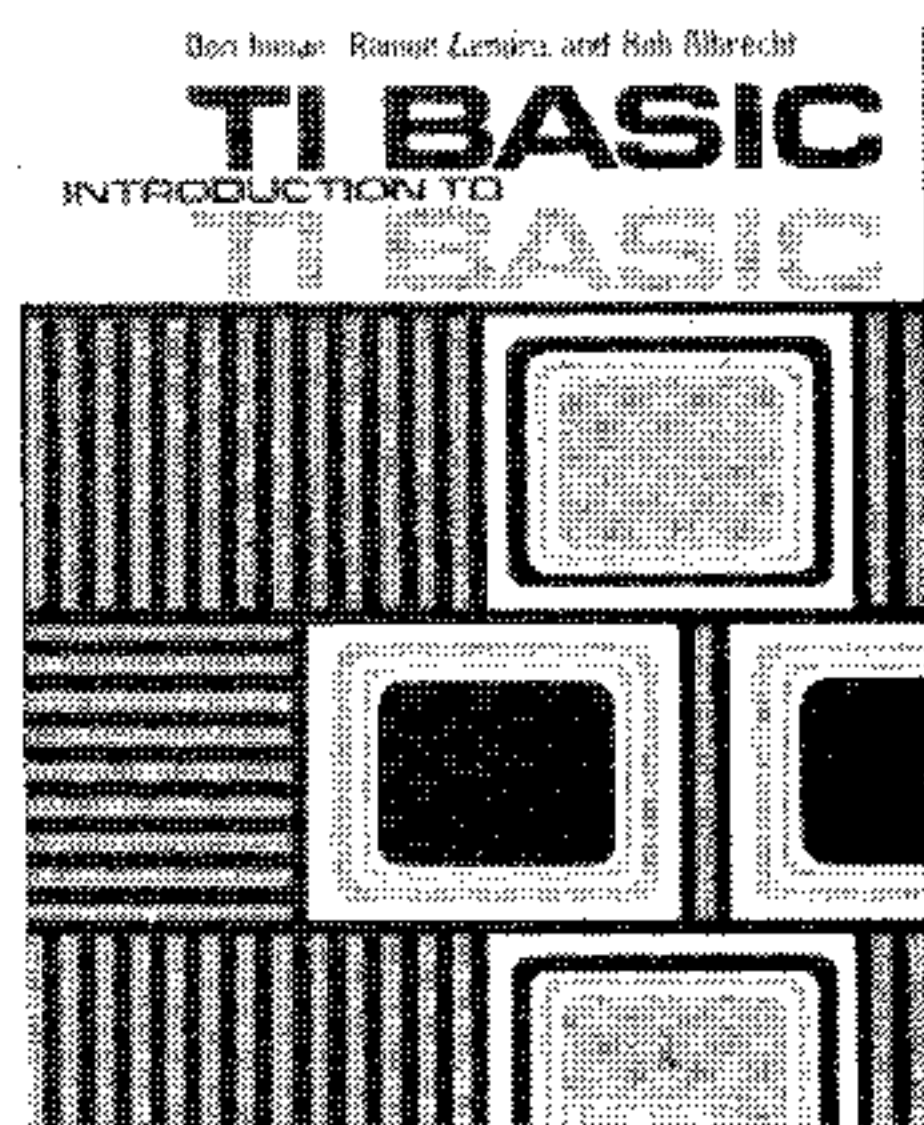
Increase Your Knowledge. Read A Book Today!

INTRODUCTION TO TI BASIC

By D. Inman, R. Zamora, and R. Albrecht

This comprehensive work will teach you all about computers and BASIC for use with the Texas Instruments Home Computer. Even if you've never worked with a computer, you can now teach yourself how to use, program and enjoy the TI Home Computer with this entertaining, and easy-to-read work. The authors have carefully constructed this introduction so that you will soon be writing BASIC programs and exploiting all of the excellent features of the TI machines. Its 14 chapters and Appendices cover all of the essential programming statements and machine features.

paper, \$10.95
1980, 320 pages, 7 x 10



PROGRAMMING BASIC WITH THE TI HOME COMPUTER

By Herbert D. Peckham

A tutorial guide that helps you learn TI BASIC in a friendly, relaxed manner. It goes beyond *Beginner's BASIC* furnished with the TI-99/4, and introduces the full range of TI BASIC features including color graphics and sound. Its 11 chapters are written in a complete-the-blanks, programmed instruction format.

paper, \$10.95
1979, 306 pages, 6 x 9

Use the order card in the back of this magazine, or itemize your order on a separate piece of paper and mail to: 99'er Magazine/Book Dept., 2715 Terrace View Drive, Eugene, OR 97405. Be sure to include check or detailed credit card information. No. C.O.D. orders accepted. Add \$1.50 postage & handling for 1 book, \$2.00 for 2 books, or \$2.50 for 3 or more books. Please allow 4-6 weeks for delivery. If there is a question regarding your order please write to Customer Service at the above address. PRICES SUBJECT TO CHANGE WITHOUT NOTICE.

DEALER DIRECTORY

Goleta, CA

See our complete line of personal and business computers including: Texas Instruments, Atari, Apple, Altos, Data General. Also All accessories, software and supplies. Plus all that's new in personal electronics. **Personal Electronics, 5674 Calle Real Center, Goleta, CA 93117, (805) 967-5322.**

Los Angeles, CA

A software/systems house primarily engaged in the sale of software for TI and Polymorphic computers. Hardware sales are usually included in a combined turnkey package. Prospective customers are seen by prior telephone appointment only. **AAAA Discount Computer How's, 13022 Pso-mas Way, Los Angeles, CA 90066, (213) 391-8777.**

Waterloo, IA

Texas Instruments Large selection of Accessories and Software for 99/4 at Dhein's True Value, 7 W. Airline Hwy, Waterloo, IA 50701. Sale prices at store are also given on mail orders. Write for price list. **Dhein's True Value, 7 W. Air-line Highway, Waterloo, IA 50701, (319) 232-6225.**

Evansville, IN

Independent Register 2414 N. Governor (across from Town Center Mall) is the tri-state location for TI-99/4 Hardware/Software, Supplies etc. We supply the tri-state with popular magazines and software for other units. Open Mon - Sat. 8-5. **Independent Register, 2414 N. Governor, Evansville, IN 47711. (812) 424-8246.**

Lexington, KY

A complete line Texas Instruments dealer. We stock all Texas Instruments calculators and accessories, learning aids, home computer products, accessories, peripherals and software. Atari computers and Hewlett Packard calculators in stock. **CBM Incorporated, 198 Moore Drive, Lexington, KY 40503, (606) 276-1519.**

Springfield, MO

A full-line Texas Instruments TI-99/4A Computer dealer—Large selection of peripherals and software—Custom programming for the TI-99/4A—Complete price list on request. **G & G Data Processing, 209 E. Walnut, Springfield, MO 65806, (417) 869-1489.**

Marlow Heights, MD

Specializing in Texas Instruments 99/4 computer and accessories. We have a large selection of Software for the TI-99/4, TRS-80, Apple, and Atari. We also stock Lobo Drives and Epson Printers. **Wills Computer Store, 4410 Stamp Road, Marlow Heights, MD 20031, (301) 423-4525.**

Minneapolis, MN

Authorized Texas Instruments TI-99/4 computer dealer offering you the best prices in the U.S.A. on the computer and all peripherals. Also, a wide variety of printers. CRT's available at lowest prices. No collect calls, please. **Calculators, Inc., 7409 Fremont Avenue South, Minneapolis, MN 55423, (612) 866-8908.**

Minneapolis, MN

Authorized TI-99/4A & Scott, Foresman dealer for educational electronic materials in the 5-state area. We have qualified educators to help with your educational microcomputer needs. We are also a full sales, service and leasing organization. **Tele-Terminals, Inc., 7216 Boone Ave. North, Brooklyn Park, MN 55428. (612) 535-5330.**

Great Falls, MT

We've got the best prices because we want your business. Complete line of TI products and Epson printers. Monthly SALES on all items. Call now or write: **IRCA Inc., N.E. Bypass Hwy. 87, P. O. Box 6125, Great Falls, MT 59406, (406) 727-8835.**

Call a Dealer Today

New York, NY

"Your Personal Guide to Personal Technology." We carry the complete TI-99/4 System. For demonstration of the TI-99/4 at home, school or office, or more information call Next Tech, Inc. today. **Next Tech, Inc., 350 Fifth Avenue, Suite 3308 New York, NY 10001, (212) 792-8768.**

Olean, NY

Texas Instruments TI-99/4 Home Computer Systems, software, hardware, and accessories at discount prices. **APF-Imagination Machine. Video Sales and Service. BLOISE Electronics, 114 Center Street, Olean, NY 14760. (716) 372-8063.**

Rye, NY

Wide assortment of TI software, hardware and accessories all at discount prices. Also authorized Scott, Foresman dealer. Write or phone for nine-page catalog with hundreds of items. **Microcomputers Corporation, P. O. Box 191, Rye, NY 10580, (914) 967-8370.**

Dayton, OH

The only full line TI distributor stocking Semiconductors, DSG (700-800 OMNI Products) TM990 Microcomputer Boards, Calculators, complete in depth inventory including 99/4 Products-Software. Ohio 1-800-762-9510. Surrounding area 1-800-543-9550. Will ship. Call us. **Esco Inc., P. O. Box 1166, 221 Crane St. Dayton, OH 45403, (513) 226-1133**

Portland, OR

Friendly pre-programmed computing systems for the home & office. Featuring Texas Instruments, Hewlett Packard, and Xerox equipment. With the TI-99, we offer discounted "Starter Packages" of all configurations. 24-Hour Home Computer Hotline. **The Electronic Cottage, 1836 NW Overton, Portland, OR 97209, (503) 224-9282.**

Call a Dealer Today

Salem, OR

J. Harvey's Video Clubhouse VCR's, Tapes, Home Computers TI-99/4, Atari 400, 800. **J. Harvey's Video Clubhouse 3295 Triangle Drive #142, Salem, OR 97302, (503) 581-1003.**

Electric City, WA

Texas Instruments 99/4 Home Computer Systems — Wholesale prices and not on just the main console! Also good prices on Epson printers and Televideo Terminals. For price list, write to: **Komputar Works, P. O. Box 483, Electric City, WA 99123, (509) 633-2653.**

Green Bay, WI

We specialize in the Texas Instruments 99/4 computer. We handle a complete accessory line for it, at discount prices. Custom programming for the 99/4 also done. Write for a complete price list. **The Micro House, 527 Simonet Street, Green Bay, WI 54301, (414) 432-2871.**

Washington, D. C.

Marinchip systems 9900 and Z800 dealer. Networking software and CPM interface. Technico experience in-house. Epson printers, tele-video CRT's, cables, TI 99/X to S-100 bus card available system customization. Hard disk, tape. Any AMP connectors. **Interface Technology, P O Box 745, College Park, MD 20740, (301) 490-3608.**

RATES

Listings here are \$30 per issue; minimum insertion, 3 issues (6 months). Prepayment of \$90 required. Ads include 35 words describing products and services, plus company name, address and phone. (No merchandise prices, please.) Call Pat at 503-485-8796 or write 99'er Magazine, Ad Department, 2715 Terrace View Drive, Eugene, OR 97405.

Call a Dealer Today

ATTENTION TI DEALERS

Did You Know That You Can Be Reimbursed For Advertising Texas Instruments Products In This Magazine?

Call or Write

99'er Magazine/Ad. Dept.
2715 Terrace View Drive, Eugene, OR 97405
Tel. (503) 485-8796

Talking in class never sounded so good.

With these speaking learning aids, it's the sound of learning. They aren't toys. They were developed, with the help of educational experts, to make learning fun by using sight and sound together.

And that makes them an excellent way for kids to get extra practice in the basics.

But the best side of our speaking learning aids is this: Kids think they're a blast. You know they're an education.

And that's a sound reason to find out more. Contact Customer Relations, P.O. Box 53, Lubbock, Texas 79408.

TEXAS INSTRUMENTS
INCORPORATED

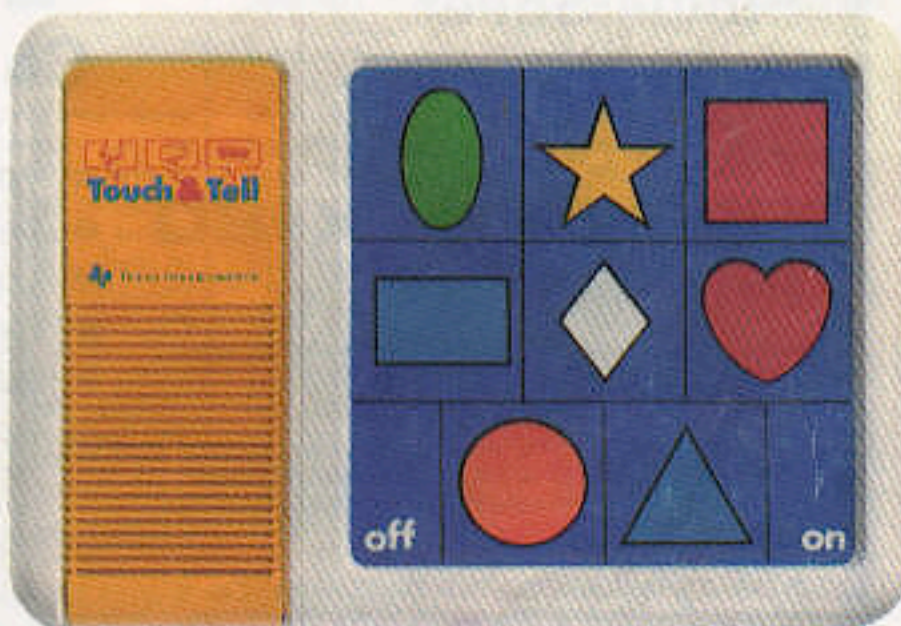


Speak & Math
For grades 1-6
Helps teach basic
math skills.

Speak & Read
For grades 1-3
Helps teach basic
reading skills.*



Speak & Spell
For grades 1-8
Helps teach basic
spelling skills.*



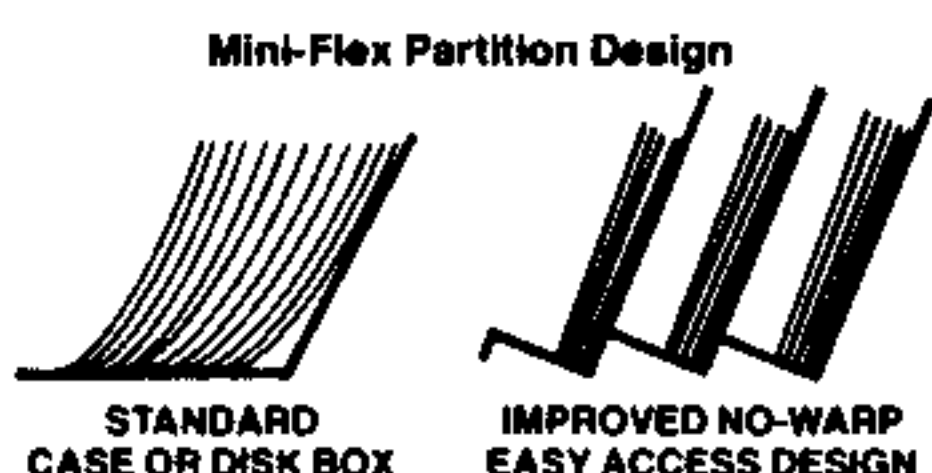
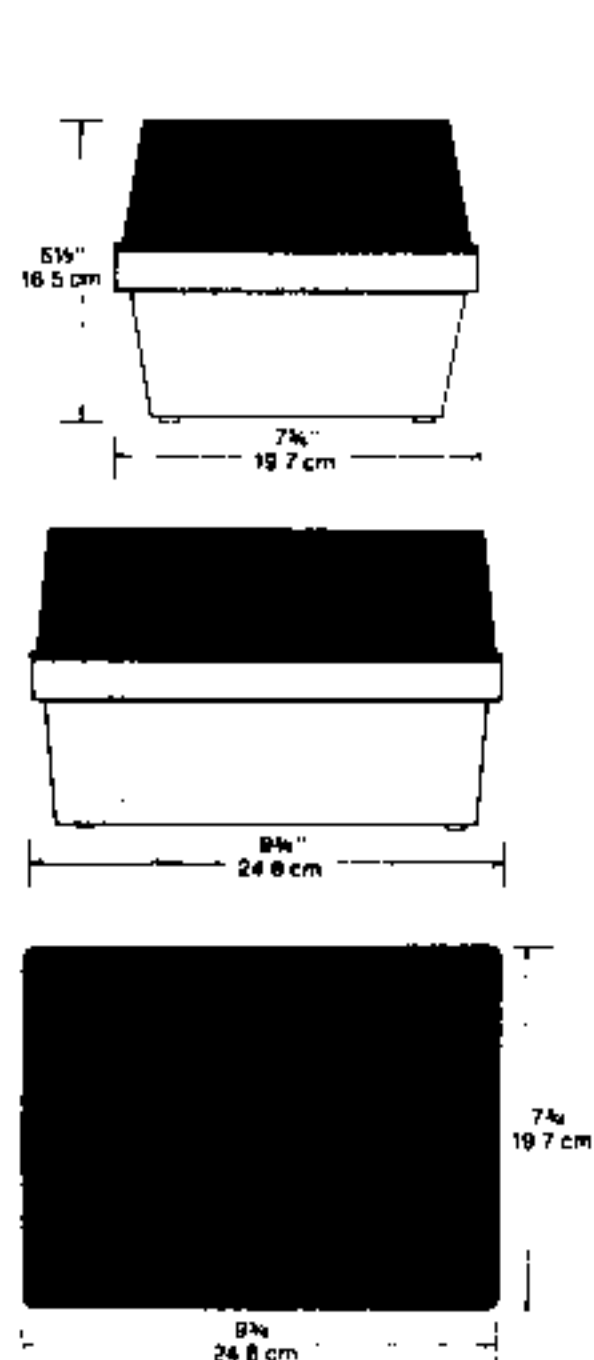
New! Touch & Tell
For ages 2-5
Helps teach basic
pre-school concepts:

Texas Instruments
makes learning what
it should be...Fun.

*Optional plug-in modules allow advancement.
©Texas Instruments, Incorporated

FAST and EASY RETRIEVAL of the DISKETTE YOU NEED . . .

. . . While Guarding Against Data Dropout Caused By Improper Storage and Handling.



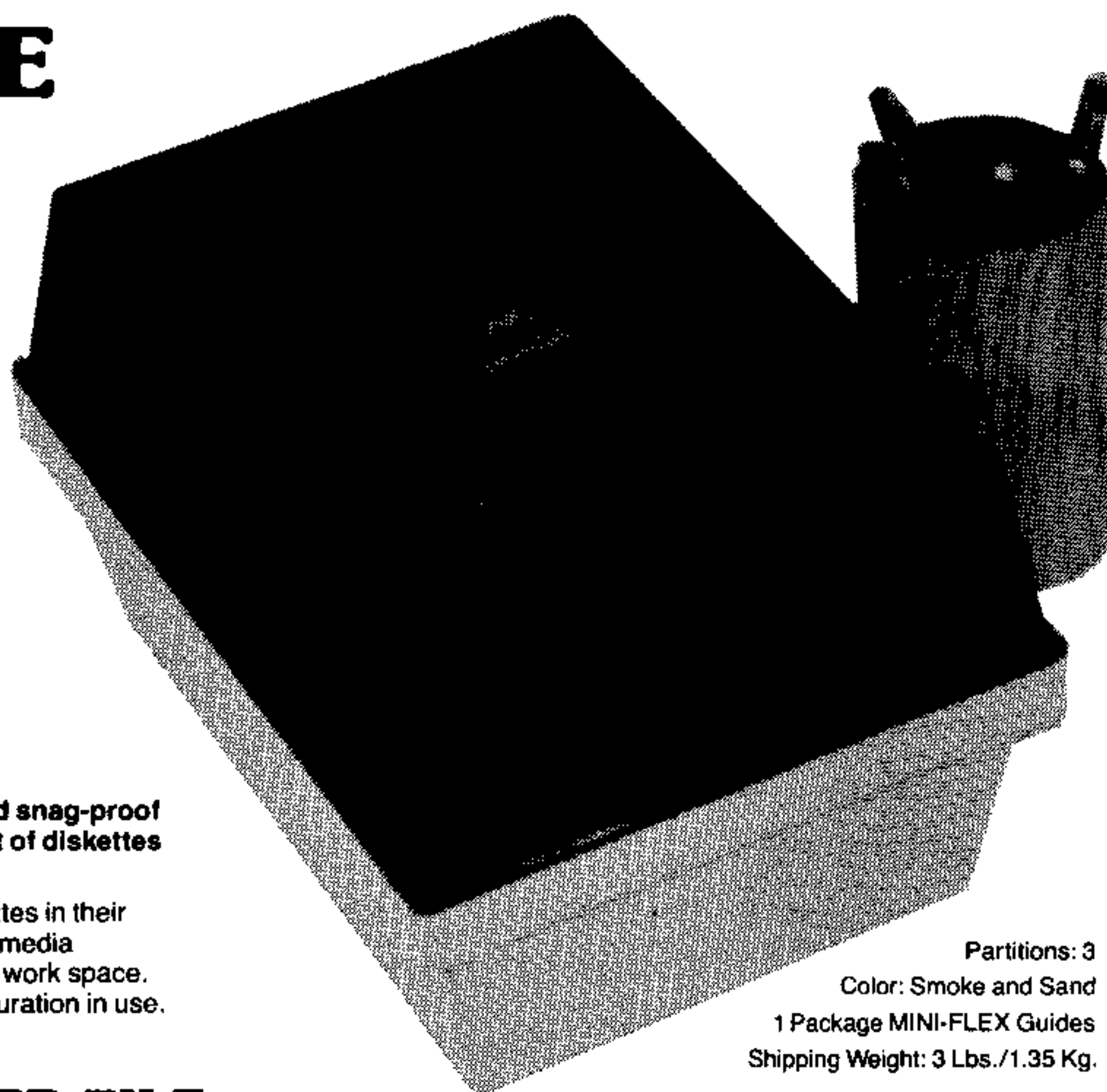
The MINI-FLEX is styled with low sides and snag-proof edges . . . making removal and replacement of diskettes easy and safe.

Spacious enough to store up to 50 mini-diskettes in their protective envelopes as recommended by all media manufacturers, yet small enough to conserve work space. The base nests in the lid for a compact configuration in use.

MINI-FLEX DISKETTE FILE

List Price: \$29.95

★ Special Price \$24.95



Partitions: 3
Color: Smoke and Sand
1 Package MINI-FLEX Guides
Shipping Weight: 3 Lbs./1.35 Kg.

Add \$2.50 shipping and handling. Use the bind-in card in the back of the magazine for your convenience in ordering. Telephone orders accepted if charged to credit cards.

FINALLY - A Cost-Effective Upgrade That EPSON MX-80 Owners Can't Afford To Be Without!

Single-Sheet
& Roll-Feed
For Your
EPSON MX-80
Printer

- Increased Printer Versatility
- Cheaper Paper Supplies
- Less Paper Wastage
- Large Time Savings
- Quickly Pays For Itself

Now You Can:

- Fill In Purchase Orders, Invoices, Statements and Other Business Forms One-At-A-Time, As Needed—Without Having to Wrestle Cartons of Pin-Feed Supplies In & Out of Your Printer.
- Install It Yourself in Less Than 45 Minutes With Only A Small Screwdriver & Penknife.
- Use Your Own Letterheads for Business Correspondence or Colored Paper for Graphics Designs, Posters, and Direct Mail Flyers.
- Use Cheap Roll Paper for Most of Your Printing Needs and Forget About Printing On or Over the Perforations.

Each Kit Comes Complete With:

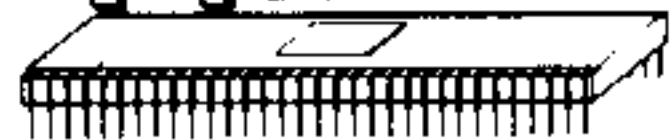
- 2 Acrylic Drive Wheels (With Rubber Friction Rings) Machined to Exacting Tolerances (+.0005, -.0.0 inches) for a Snug Fit
- 1 Weighted Pinch Roller Assembly
- 1 Roll Paper Holder

Special Price:
ONLY \$49.95

Add \$2.00 for Shipping & Handling. Use the bind-in card in the back of the Magazine for your convenience in ordering. Telephone orders accepted if charged to credit cards.

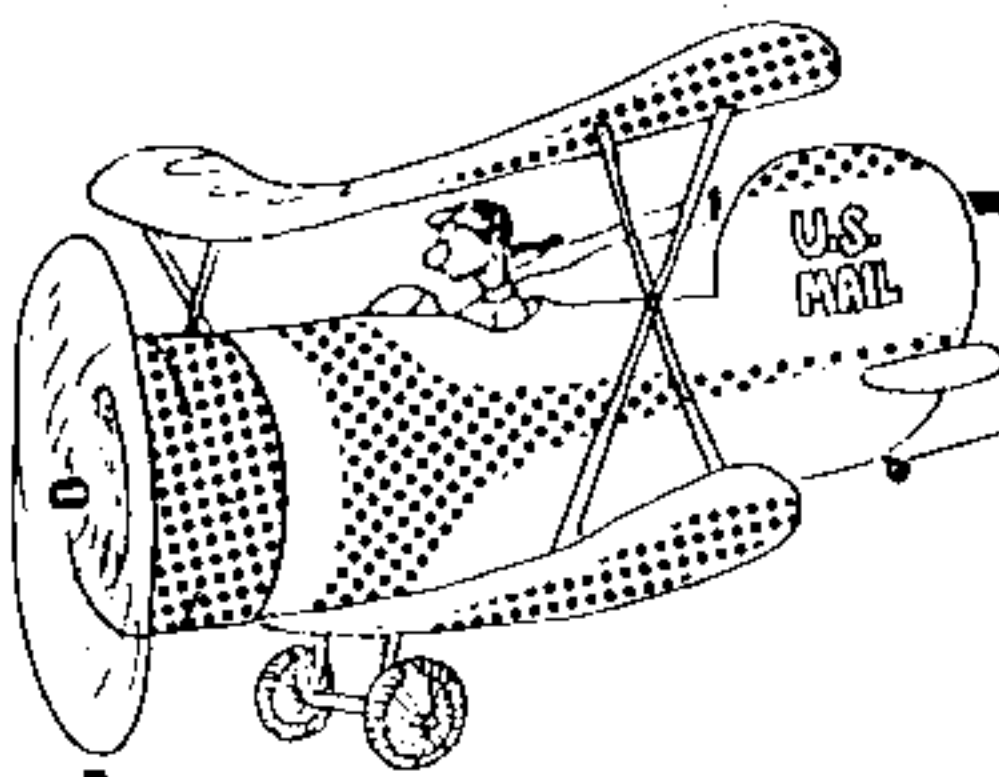


99'er-ware



INNOVATIVE PRODUCTS
FOR TMS9900-BASED
PERSONAL COMPUTING

P.O. Box 5537
Eugene, Oregon 97405
Tel. (503) 485-8796



LETTERS TO THE EDITOR

Dear Sir:

In your July/August issue a reader described a very good idea; a "general purpose" load program which could be used both to display a menu of all programs on a disk and to allow program selection from the menu. While the program he provided did the first task, it could not do the second because TI Extended BASIC does not allow substitution of literals for device-filenames in the RUN statement.

But there is a way around the problem. A program is represented in memory beginning at location -1 (i.e., -1, -2, -3, etc.). Using the PEEK subroutine one can examine the contents of each memory location, and since program lines are represented in condensed code, it is an easy matter to determine the location at which the RUN "DSKx.FILENAME" statement should be. User selected values for x and FILENAME can then be "poked" in using the LOAD subroutine. [All this assumes the user has the 32K Memory Expansion—Ed.]

In the program below, the final statement is a REMark followed by several asterisks. The preceding LOAD statements literally replace this REMark statement with a RUN "DSKx.FILENAME" statement. The program then comes upon that statement and executes it as if it had been there from the beginning. Thus the procedure has the same effect as substitution of literals for device-filenames.

Should the reader wish to try the program, the following instructions must be followed carefully:

1. Make sure the RAM expansion, Extended BASIC, and disk drive(s) are properly connected and turned on in the proper order.
2. Enter the program *exactly* as shown including all REM statements. If any changes are made, addresses for LOAD statements will not be valid and the program will not work.
3. After the program has been entered and corrected, it is essential to insure that all statements are in proper sequence. This is done by saving the program with the MERGE option (e.g., SAVE DSK1.X, MERGE), entering the command NEW, and entering the MERGE command (e.g., MERGE DSK1.X).

```

1 REM *****
2 REM 3 GENERAL PURPOSE 3
3 REM 3 PROGRAM LOADER 3
4 REM 3 BY C.M. EDINGER 3
5 REM *****
6 OPTION BASE 1
7 DIM P$(20)
8 INKEY 00
9 CALL CLEAR
10 DISPLAY AT(12,6)ERASE ALL:"DISK? (1-3): 1";
11 ACCEPT AT(12,19)SIZE(-1)VALIDATE("123");D$
12 OPEN #1:"DSK"&D$&".",INPUT,RELATIVE,INTERNAL
13 INPUT #1:IN$,A,A,A
14 DISPLAY AT(1,8)ERASE ALL:"DSK"&D$&" - "&IN$;
15 I=0
16 FOR I=1 TO 20
17 I=I+1
18 IF I>127 THEN 42
19 INPUT #1:P$,A,B,B
20 IF LEN(P$)=0 THEN 26
21 IF ABS(A)>5 THEN 19
22 DISPLAY AT(X+2,10)USING B:1;
23 DISPLAY AT(X+2,14):P$;
24 P$(I)=P$
25 NEXT X
26 DISPLAY AT(X+2,10)USING B:1;
27 DISPLAY AT(X+2,14):"TERMINATE";
28 DISPLAY AT(X+3,14):"CHOICE? 1";
29 ACCEPT AT(X+3,22)SIZE(-2)VALIDATE(DIGIT);K
30 IF K=X THEN 42
31 IF K<1 OR K>20 THEN 28
32 IF LEN(P$(K))=0 THEN 28
33 CLOSE #1
34 CALL CLEAR
35 REM *****
36 REM 3 HOOKEY POKEY 3
37 REM 3 BY JOHN CLULOW 3
38 REM *****
39 CALL INIT
40 PR#="DSK"&D$&" - "&P$(K)
41 A=-991
42 L=LEN(P$)
43 IF L=13 THEN 48
44 FOR I=1 TO L-1
45 CALL LOAD(A,0)
46 A=A-1
47 NEXT I
48 FOR I=L TO 1 STEP -1
49 B=ASC(SEG$(P$,I,1))
50 CALL LOAD(A,B)
51 A=A-1
52 NEXT I
53 CALL LOAD(A,L)
54 A=A-1
55 CALL LOAD(A,199)
56 A=A-1
57 CALL LOAD(A,169)
58 A=A-1
59 L=L-4
60 CALL LOAD(A,L)
61 REM *****
62 STOP

```

4. If all of the above steps have been done correctly, the program in memory can be saved (SAVE DSK1.LOAD) and subsequently used and copied like any other program. When saved as .LOAD on any disk which contains programs, it will provide a menu of those programs upon entering Extended BASIC, and will allow the user to select a program by pressing a single key.

John Clulow
Perrysburg, OH

Dear Sir:

The program included with the article "Electronic Home Secretary" is very handy indeed. But the information block that shows the telephone frequencies is particularly useful. Anyone who uses one of the modern banking services generally known as "bank-by-phone" has been through the hassle of pushing all the touch-tone buttons on their phone to get the bank's computer to understand what you want to pay. (I need about 60 depressions to make a single transfer of cash from savings to checking.) Mr. Subbaiah's information can speed this up considerably.

By making a small array to hold the frequencies, a person can then develop a short program of inputs which can be interpreted by VAL(SEG\$...) to do the "button pushing" for them (including making the phone call). Just think how creditors will thank Mr. Subbaiah for his article since people will find it so easy to make their payments by phone.

A word of caution however! I suggest the user test the program well before relying on it at the last moment of payment. The program would most likely have to be individually tailored to the user in order to be efficient since different banks use different keys for different functions.

We thank Mr. Subbaiah for the very useful information and hope to see more of the same in your fine magazine.

Jim Schwaller
Cincinnati, OH

Dear Sir:

The August 81 issue of *Interface Age* contains a "benchmark" program, and the times for a number of machines including the 99/4. I thought it would be interesting to try the program in Extended BASIC. The time for TI BASIC is listed as 2479 sec, which I verified. The time for TI Extended BASIC turned out to be 3407 seconds, or 37.43 percent slower!

The news is not all bad, though, because I tried the program again after inserting the commands: CALL INIT :: CALL LOAD (-31878,0) which disable the sprites if one has Memory Expansion. Thus configured, the program took 1853 seconds, which is 25.25 percent faster than TI BASIC, and 45.61 percent faster than unmodified TI Extended BASIC.

I think this information is interesting enough to be included somewhere in your magazine. I am naturally curious about how the LOAD command works, and whether there are other interesting modifications that can be similarly made. TI included the above in their May 81 newsletter. Would it be possible for you to find out more, and tell us the story?

I enjoyed the first issue of 99'er. Good luck.

Roger B. Kirchner
Northfield, MN

Keep your eye on our *Advanced Programming Techniques* column, Roger.

PLEASE DON'T FORGET TO
RETURN THE QUESTIONNAIRE
ON THE FRONT BIND-IN CARD.

Continued on p. 93

Dear Sir:

I have received your magazine and find it to be an absolute masterpiece in the gallery of specialized periodicals. After spending a small fortune subscribing to computer magazines trying to find some information about the 99/4 and/or software for it, the investment has finally realized some fantastic gains with your magazine. Unfortunately, there will have to be some dumping of the turkeys. Fortunately, I shall have the 99'er to sustain my appreciation of the 99/4 and its capabilities.

My little 99/4 system has been a great workhorse for me in many different respects, and I am very happy to have it. Since I spent a long time looking at different systems and prices and software and etc., the decision that was made over a year ago for me to buy the 99/4 is still the best I could have made. The surprising thing to me is that I have seen nothing yet which can adequately compare with the 99/4.

Robert L. Nelson
Suring, WI

Dear Sir:

REF: 99'er, Vol 1, No. 1, page 83—Personal Computing to Aid...

If John Hopkins has another "SEARCH" next year, I hope you will devote more space to this outstanding program.

Thanks to TI and some other fine folks, I was able to enter for *The National Paraplegia Foundation* and feel honored to be asked to demonstrate our modified 99/4 and Paracare programs at NASA. There are no losers in this kind of contest.

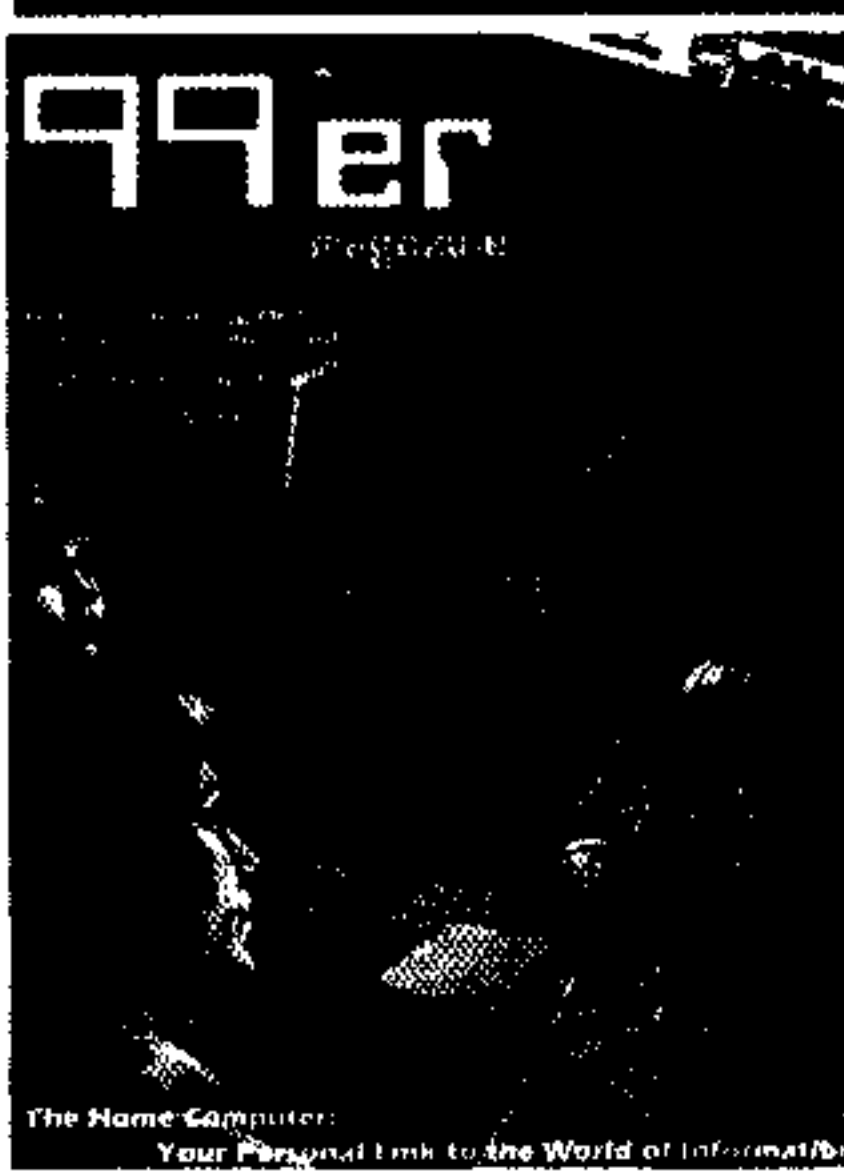
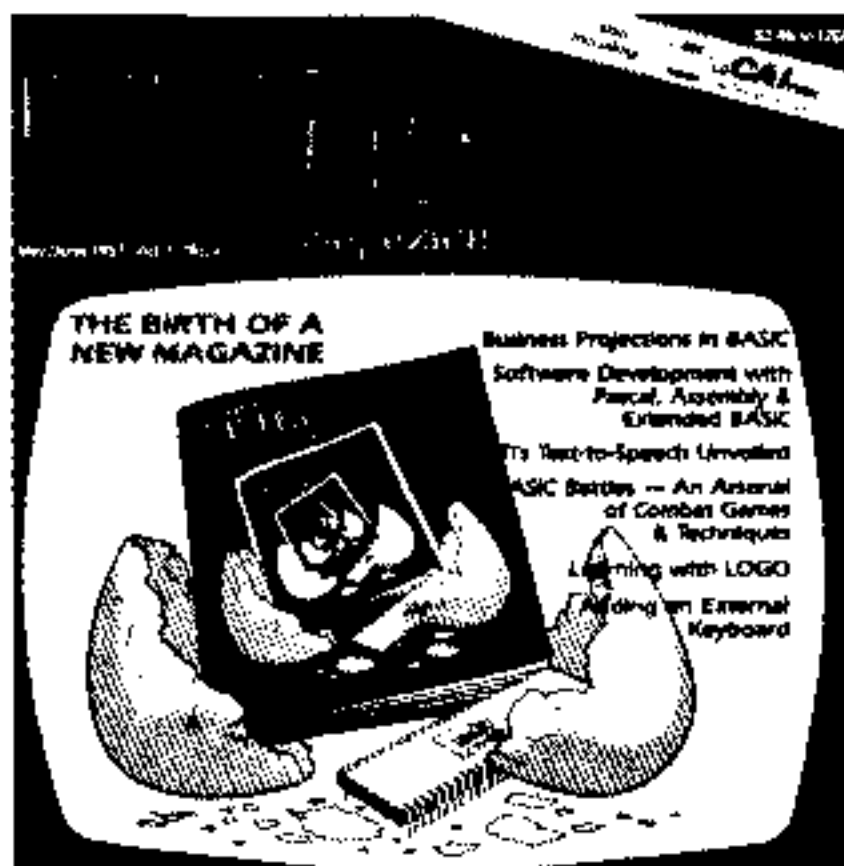
As a beginner, I appreciate getting started with your magazine. It should be very helpful to me.

Jack Kishpaugh
Fort Worth, TX

It's good to hear about the work you're doing, Jack. We will shortly be launching a new column, *Personal Computing for the Handicapped*, and are presently soliciting manuscripts from interested 99'ers.

Back Issues of

99'er
T.M.
magazine



are Still Available . . .
. . . but quantities are limited

so ORDER TODAY!

Each Only \$3.95 postpaid

ISSUE #1 (Partial Contents):

- How To Write Your Own Programs
- An External Keyboard
- Epson MX-80 & Printer Graphics
- Space & Combat Games in TI BASIC
- Text-to-Speech
- TMS9900 Machine & Assembly Language
- Planning & Forecasting in BASIC
- UCSD Pascal & Third-Party Development Systems
- Power Line Problems
- LOGO & Lamplighter
- Music Text Editor
- TM990/189 University Module
- Homework Helper: Fractions
- Computer Chess
- and much, much more . . .

ISSUE #2 (Partial Contents):

- Information Utilities and the Electronic Cottage
- Data Communications
- Marinchip Systems M9900
- Software Conversion: TRS-80 to TI BASIC
- Small Investor and the TI-99/4
- The Electronic Home Secretary
- Typing Tutor
- TI LOGO and the Space Shuttle
- Civil Engineering Fundamentals
- Catch & Match Games in BASIC & Extended BASIC
- Bombs Away on the University Module
- Interfacing a Digitizer
- and much, much more . . .

Coming in the Fall 1981 Issue

- Games & Simulations Galore
- Micro Bartender
- Bit-Plot Printer Graphics
- Computer Choreography
- Speech Concatenation
- Electrical Engineering Fundamentals
- 99'er Matrix Cruncher
- Rule of 78
- Interfacing with Teletypes
- HI-RES Plots & Graphs

★ Plus
And

Programming Differences Between the TI-99/4 & TI-99/4(A)

For programs to be compatible with both console models, a few programming conventions must be observed: First, avoid use of the SHIFT and ENTER keys as fire buttons. Instead, use keys such as "Q" and "P" (as in *Space War*), or "F" and "H" (as in *Dogfight*).

One other major difference concerns the "X" and "M" keys—typically used as directional arrows, indicating a straight downward movement: When you specify a key-unit of 1 or 2 (for a split keyboard scan) in the CALL KEY subroutine, the computer does not return a true 0 (zero) in the return-variable when X is pressed on keyboard 1 or M on keyboard 2. Therefore, if your program checks to see whether the return-variable equals zero, the computer returns an answer of false, and you won't be able to achieve straight-down movement. To illustrate a way around this problem, see lines 2380 and 2620 in *Space War*. Notice that if instead of coding these lines

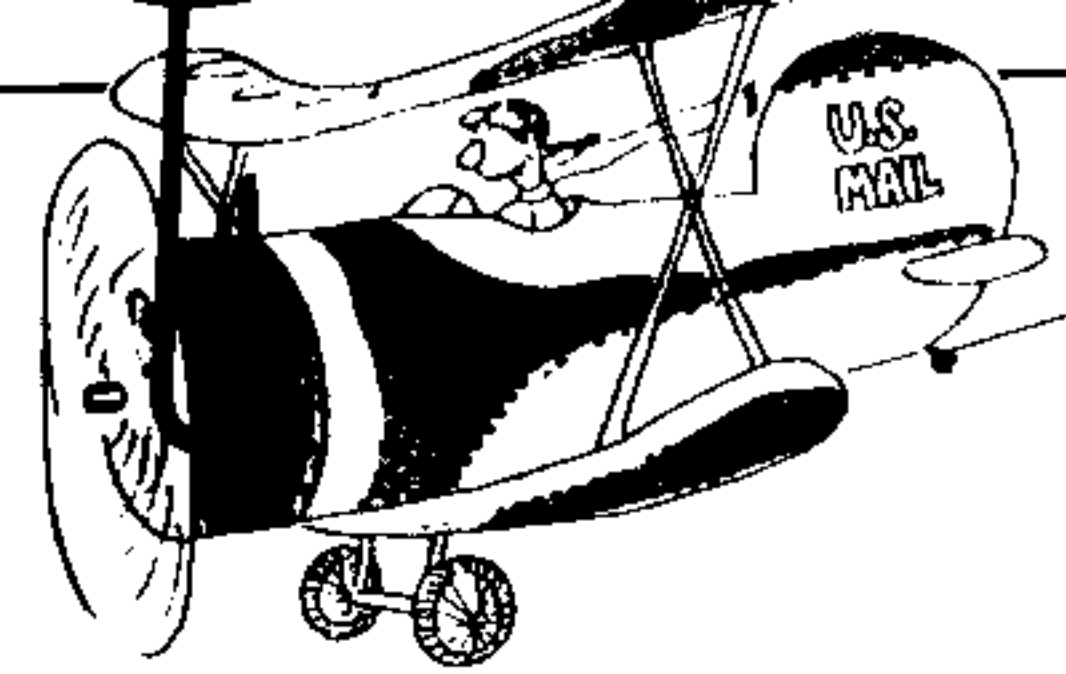
```
2380 IF K1 = 0 THEN 450
.
.
2620 IF K = 0 THEN 2790
```

to test for the zero return-variable (which wouldn't work on the TI-99/4A), you add a 1 to both sides and test for the integer,

```
2380 IF K1+1 = 1 THEN 450
.
.
2620 IF K+1 = 1 THEN 2790
```

the logical switch operation is the same, and the new coding will work on *both* the TI-99/4 and TI-99/4 A.

Since *99'er Magazine's* goal is to publish programs that will run "as is" on both models, we request that when readers send us listings for possible publication, they should implement their coding accordingly. We'd also appreciate it if anyone who "RUNs" across any other differences between the two machines inform us immediately so that the information may be disseminated.



Dear Sir:

Subject: TI-99/4 Software Print Restrictions

Most of the software as designed by TI has provisions for other optional printers to be utilized with the 99/4 hardware and their software. This software, whether in program form of cassette, disk, or command module all contains print instructions designed for the TI Thermal Printer.

Those of us 99/4 Owners with other printers, I'm sure, would find it advantageous to fully utilize the 80 to 132 character per line capability of the particular printer. At present, these TI software programs limit the printer to 32 characters per line.

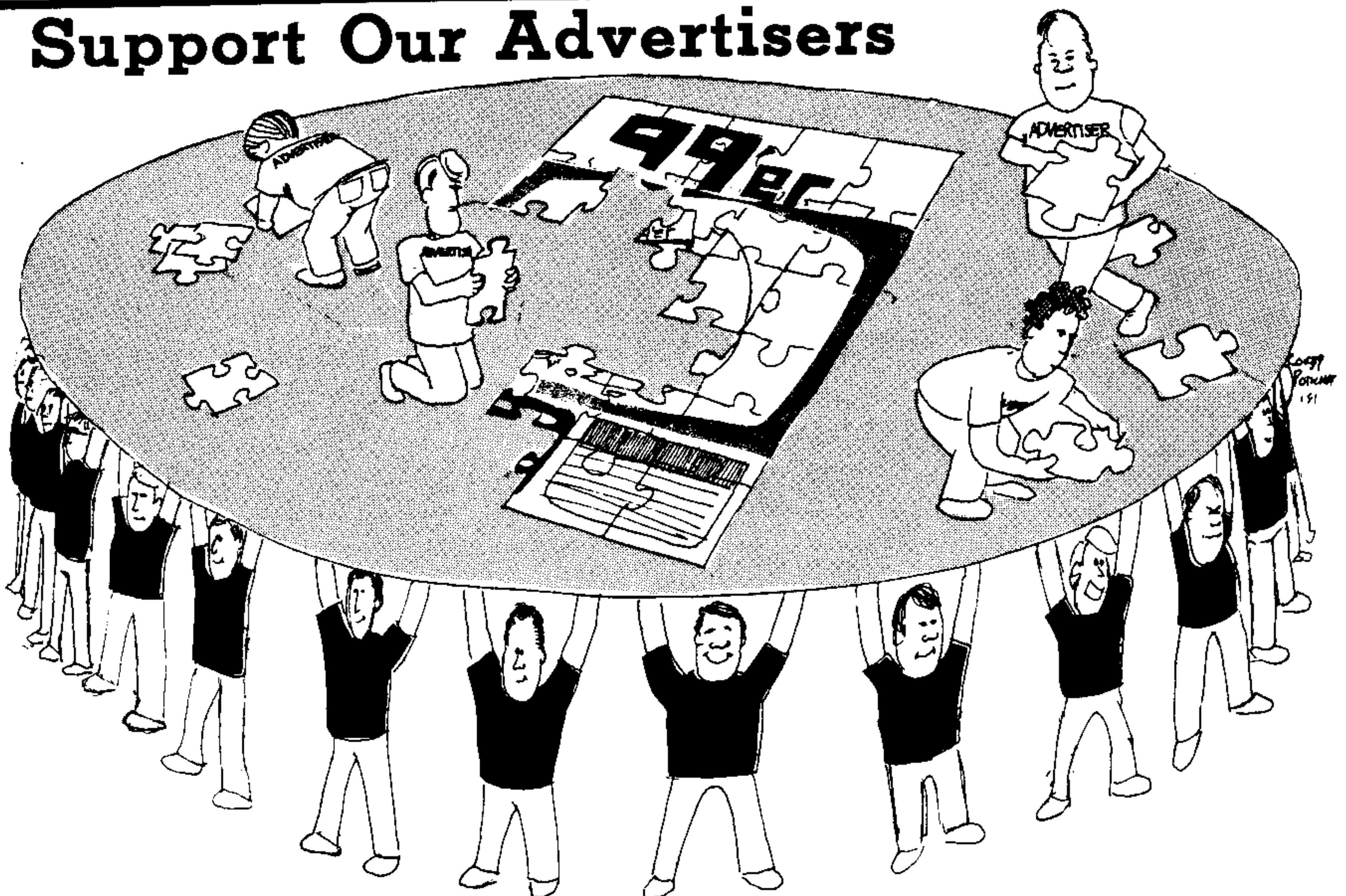
Personal Record Keeping and the Personal Checking Account programs are excellent examples. Instead of one complete line of data under various topic headings in the 80 or 132 character line, the result is perhaps two half-page lines of data to comply with the TI Thermal Printer's capacity restrictions.

It would be interesting to have a special program to resolve this situation, i.e., a subroutine that user's could blend into their TI cassette tape and disk software.

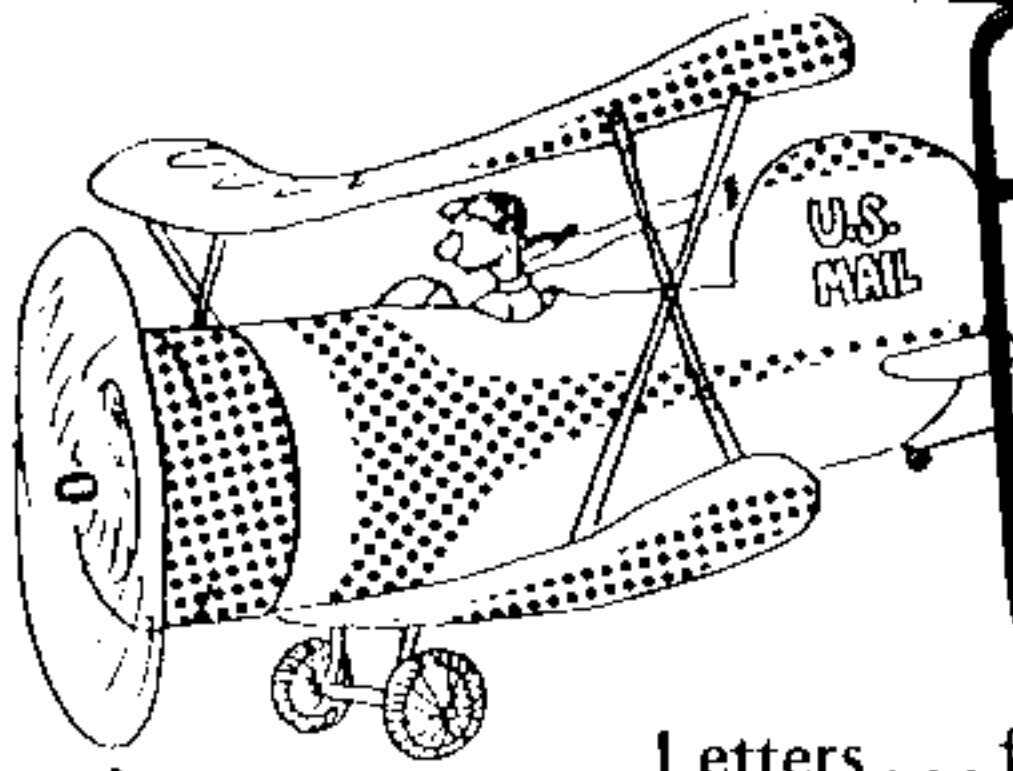
John O. K. Jacobsen
Eugene, OR

Continued on p. 94

Support Our Advertisers



They Make This Magazine Possible



LETTERS TO THE EDITOR

Letters . . . from p. 93

Dear Sir:

I am very impressed at the quality of the 99'er. It is everything a 99/4 owner could ask for. Actually, it makes all other personal computer magazines obsolete, since they only write about Apple and Radio Shack.

I've loaded in two of "Kelley's Korner" games. Besides being enjoyable they have an educational quality. I'm new to TI BASIC and looking at someone else's program is very helpful. James Dugan's article "Using Flow-Charts" was also very helpful.

I'm looking forward to the next issue. Thanks for publishing a magazine just for me and my TI 99/4 Home Computer.

Fred Werginz
College Park, GA

**PLEASE DON'T FORGET TO
RETURN THE QUESTIONNAIRE
ON THE FRONT BIND-IN CARD.**

Dear Sir:

I am very satisfied with interesting information, soft- and hardware news, descriptions . . . in 99'er Magazine. I can not do without any next issue 99'er Magazine get lost. So please give Magazine into enough strong envelope, because I am far from USA. I am received now first two issues of 99'er Magazine. Thank you!

I wish you much success in your exertions for a good 99'er Magazine! And Thank You! Excuse me for my broken English, please!

Likeb Stane
Ljubljana, Yugoslavia

A Micro "Extended" Editorial

Although both of the *Kelley's Korner* games in this issue are very similar in design, the choice of programming languages creates games that are vastly different in implementation and performance. This difference clearly illustrates the limits of TI BASIC, and additional facilities of Extended BASIC that a programmer can use to push far beyond these limits.

For example, when a missile is shot in *Space War*, all action stops until the missile either hits or passes off the screen. Since *Dogfight* utilizes the sprites of Extended BASIC, bullets can be shot simultaneously by each player *without* interfering with the independent motion of each plane. The speed of the moving sprites are also quite a bit faster than the HCHAR and VCHAR CALLs of regular BASIC. All this adds up to a more realistic, "arcade-like" video game for the Extended BASIC implementation.

The other major difference lies in the amount of coding required for each game (which translates into a significant time difference for readers who key in these programs): *Space War* requires over 600 lines of code, whereas *Dogfight* needs less than 200 lines. There are several reasons for this great disparity: (1) In Extended BASIC, sprites can be defined as four characters, and so *one* statement in this language would take *four* in BASIC; (2) Checking for hits also requires less statements in Extended BASIC (due to the coincidence subroutines); (3) Whereas the autowrapping characteristic of sprites are employed naturally in *Dogfight*, coding *Space War* in BASIC required the programmer to *force* the wrap and to check for it on each move; (4) And finally, additional coding is saved by the complex IF-THEN-ELSE statements that *Dogfight* is able to take advantage of.

All this is not to say that *Space War* is a bad program. On the contrary, it is relatively fast, requires quite a bit of strategy and skill, and is a lot of fun to play. The purpose of this brief analysis was to point out to many of our readers—even the non-programmers—who don't yet have the Extended BASIC Command Module, that this programming language can, in fact, really enhance the enjoyment received from the computer. And as more of our readers add this capability, *99'er Magazine* will be able to publish more original software in Extended BASIC—programs that take up significantly *less* space in the magazine, and so will allow us to include *more* programs for your enjoyment.

MINDSTORMS THE DEFINITIVE BOOK ON LOGO

Get Your Copy
TODAY!

See 99'er BOOKSTORE

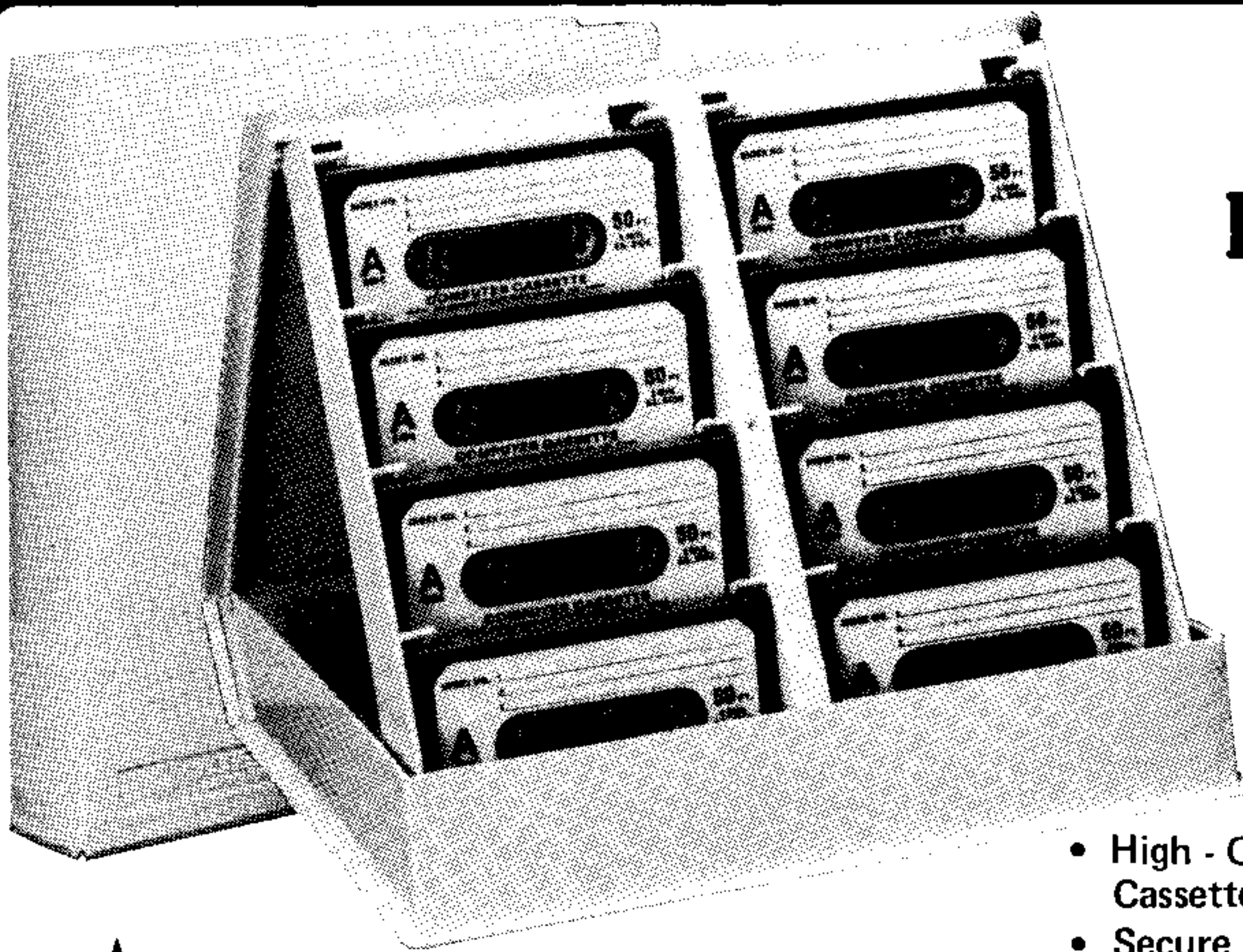
Index to Advertisers

A. ACE Computer	3	Letcher Offshore Design	55
American Software Design & Distribution Co.	39	Linear Aesthetic Systems	53
Anthistle Systems & Programming Ltd.	77	The Micro House	30
The Bach Company	17	Micromate Electronics	44
CBM Inc.	21	Charles Morreale	84
CompuTech	53	Norton Software	38
Computer Assistance	55	Pablo Diablo	84
Computer Mail Order	66	Paul Yates Computer Services	38
Computer Roomers	78	Pike Creek Computer Co., Inc.	30
C. S. C.	43	PRP Computer Graphics	34
Cumberland Technology	55	RCL Computers	11
Data Systems	39	RKS Enterprises, Inc.	82
Denali Data	52	Scott, Foresman & Company	68
East Bench Software Products	30	Small Business/Professional Computer Services	43
Electronic Specialists, Inc.	16	Software Exchange Group	31
Ehninger Associates, Inc.	16	Sunshine Software	53
ELEK-TEK, Inc.	63	TAM'S Inc.	2
Epson America, Inc.	7	Texas Instruments, Inc.	89, 96
Extended Software Co.	9	TI MORE Products	24
Fantasy Computing	62	UMI	16
FFF Software	85	Unisource Electronics, Inc.	71
Galactic Software	84	W. R. Wilson Co.	22
George Goode & Associates	45	Young Peoples LOGO Association	74
Innovative Concepts, Inc.	83	99'er Bookstore	86, 87
Kemp Software	22	99'er Magazine	92
Komputar Works	9	99'er Ware	90, 95

ALL 99'er-ware PRODUCTS PAGE MAY BE ORDERED USING THE BIND-IN CARDS NEAR THE FRONT & REAR OF THIS MAGAZINE

DIGITAL COMPUTER CASSETTES

with
LIBRARY ALBUM



★ Only \$10.95 each; or 3 for \$28.00

Add \$2.00 shipping and handling for the first album; 75¢ for each additional album. Use the bind-in card in the back of the magazine for your convenience in ordering. Telephone orders accepted if charged to credit cards.

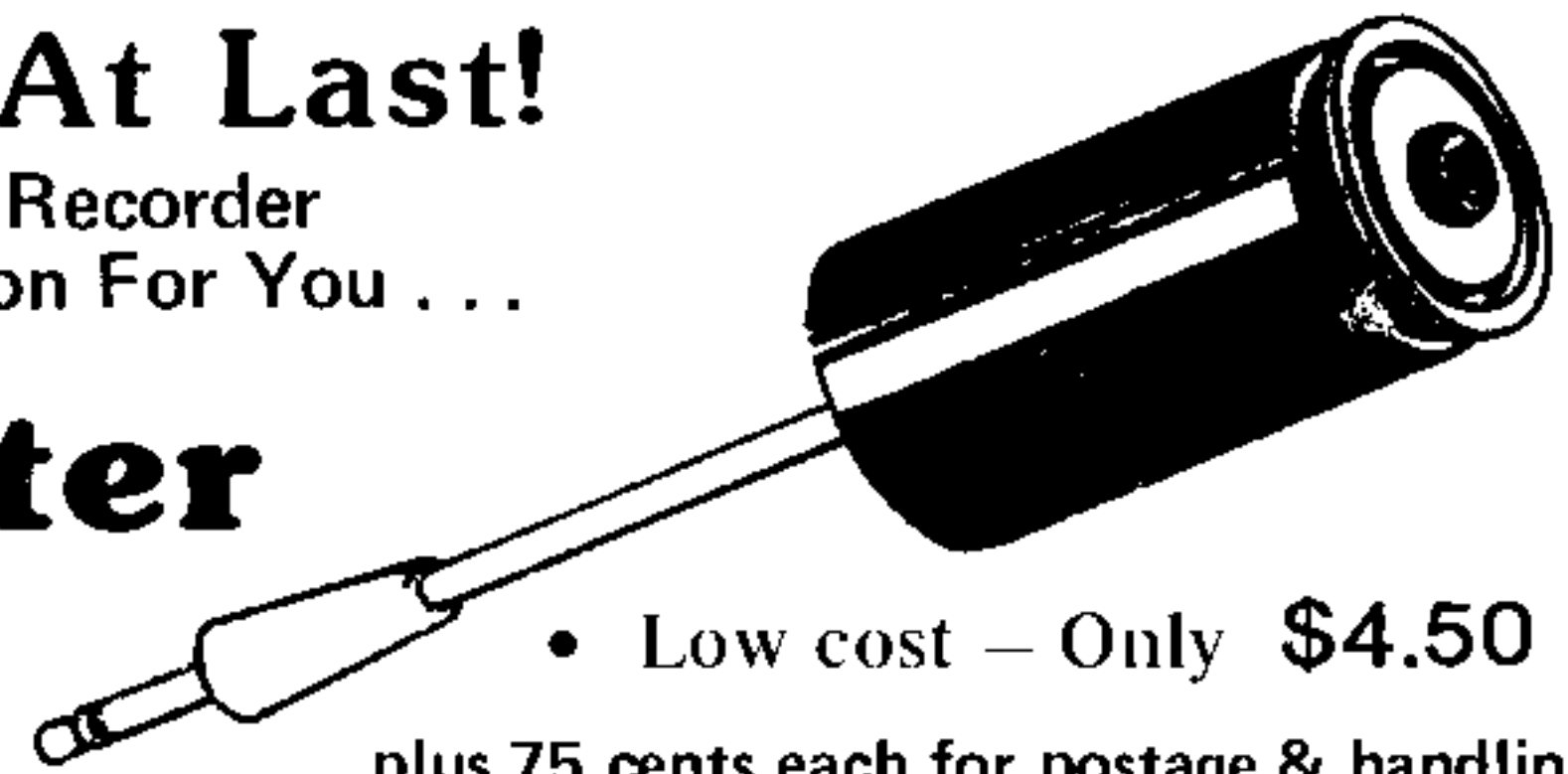
- High - Quality BASF Tape in a 5-Screw Cassette Housing for Data Integrity
- Secure Storage & Quick Convenient Retrieval with Protective File Album
- 8 Special 50-Foot Cassettes — Ideal for Building Up Your 99'er Program Library
- Economical As It Is Functional

Cassette Compatibility At Last!

If The TI-99/4 Will Not Control Your Cassette Recorder Through Its Remote Jack, We Have The Solution For You . . .

The TI-SETTE™ Adapter

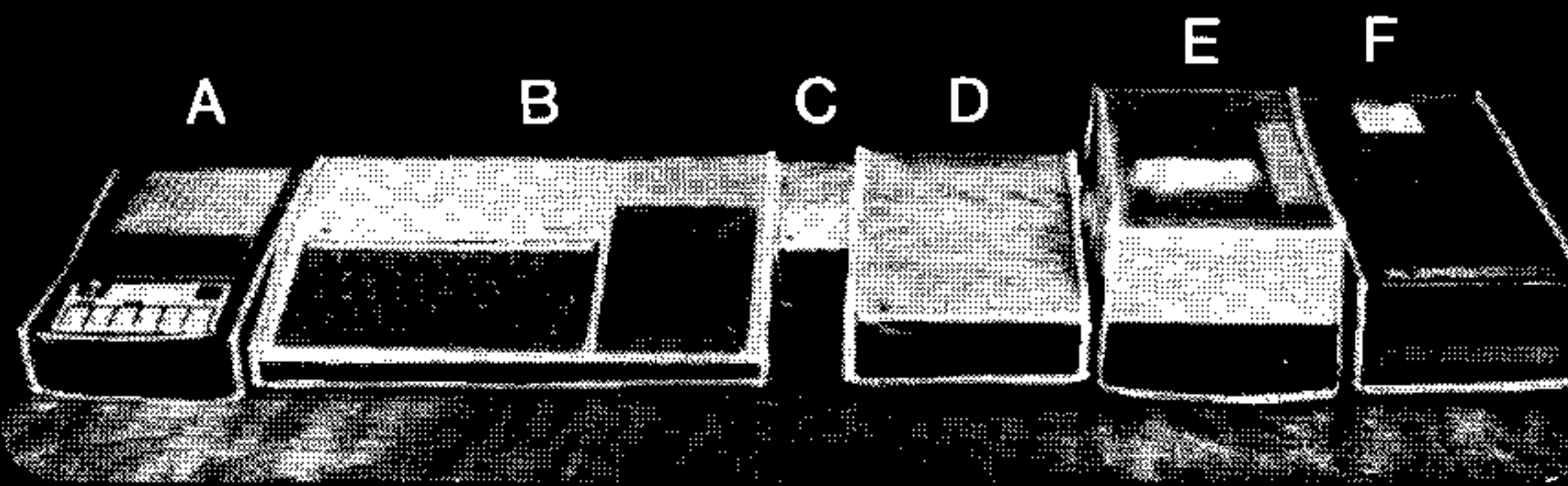
- Quickly connects between the black lead on the TI Dual Cassette Cable and your recorder's remote jack to establish compatible polarity.



• Low cost — Only \$4.50

plus 75 cents each for postage & handling.

DUST COVERS



Features:

- Equipment Protection
- Custom-Fit
- Handsome Appearance
- Antistatic Treated
- Quality Construction

A	Cassette Recorder Cover	\$4.95	E	Thermal Printer Cover	\$8.95
	(1 size fits up to 10½" x 6½")		F	Disk Drive Cover	\$5.95
B	TI-99/4(A) Console Cover	\$8.95	—Covers Not Shown in Photo:—		
C	Speech Synthesizer Cover	\$3.95		10" Color Monitor Cover	\$10.95
D	Peripheral Box Cover	\$5.95		13" Color Monitor Cover	\$11.95
	(Specify Exp. Ram, RS232, Disk Controller)			Epson MX-80 Cover	\$9.95

Add \$2.00 shipping/handling for the first dustcover; 50 cents for each additional cover.

DEALER
INQUIRIES
INVITED

99'er-ware



INNOVATIVE PRODUCTS
FOR TMS9900-BASED
PERSONAL COMPUTING

P.O. Box 5537
Eugene, Oregon 97405
Tel. (503) 485-8796



TI Logo: IT OPENED A DOOR THEY THOUGHT WAS LOCKED.

It opened a door to their minds.

The key: a Texas Instruments Learning Computer and TI LOGO, a programming language developed by TI and MIT.

In his inner-city, New York junior high classroom, teacher Steve Siegelbaum explains why it works so well. "When they use it, they think they're teaching the machine. In reality, it's teaching them how

to learn. It definitely improves their attitude toward their other courses. Written and verbal expression improve—they're eager to show you, to tell you, what they've done."

Another teacher, Pete Rentof, adds, "What it fights is fear of failure—a mistake becomes a starting point. The whole learning process turns into a positive experience. It works."

The TI Learning Computer, with TI LOGO and many other educational programs, is equipped to help open doors in any classroom. Including yours.

For information on this remarkable system, contact:

Texas Instruments Customer Relations, P.O. Box 53
Lubbock, Texas 79408.



TEXAS INSTRUMENTS
INCORPORATED