Vol. 1, No. 6 $3.50 in U.S.A.

For People Interested in the Enrichment of Personal Computing

99'er magazine™

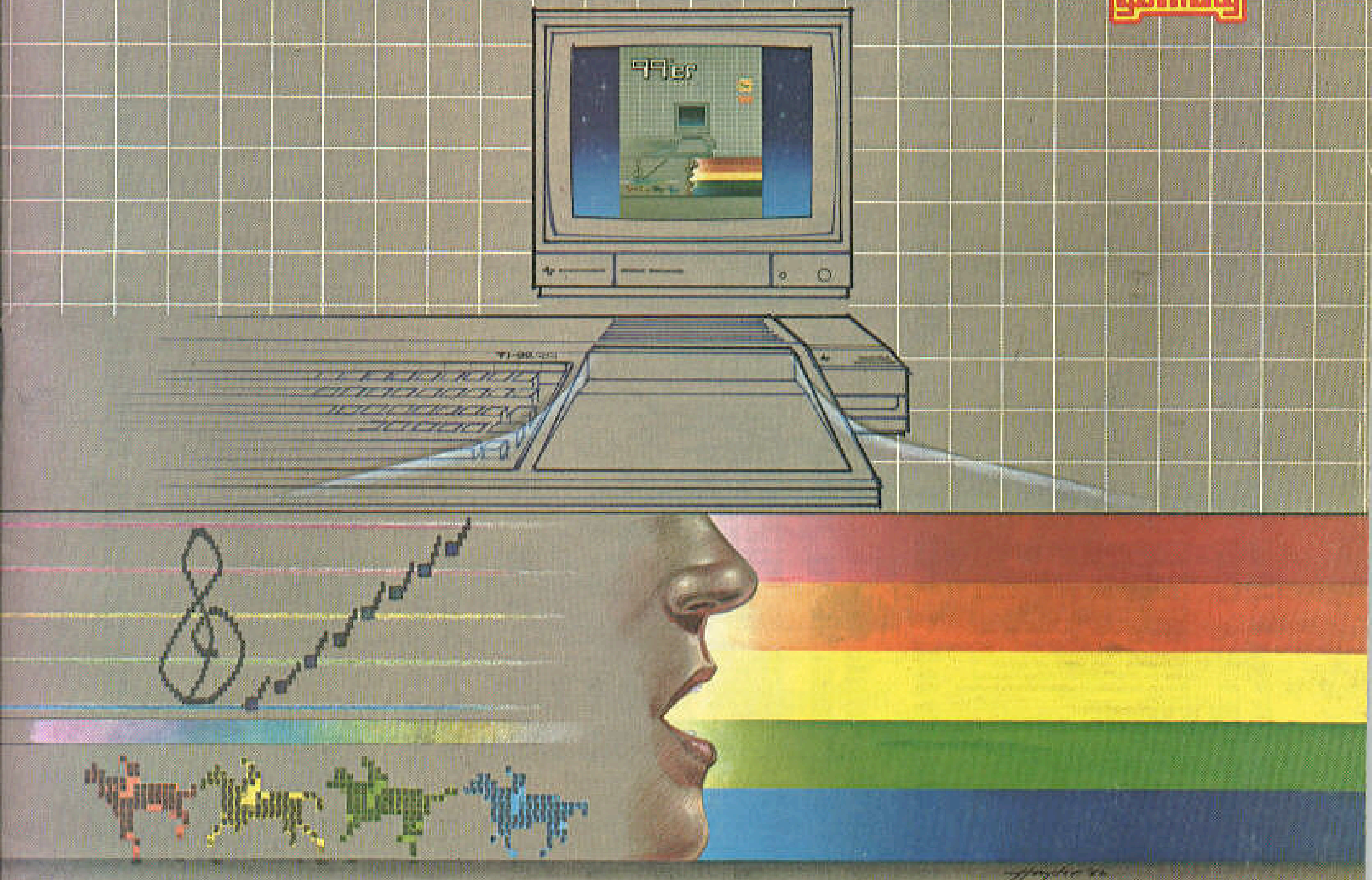
Covering the TI-99/4A and
other 16-Bit Texas Instruments
Personal Computer Systems

Also Featuring:

LOGO TIMES™
THE MAGAZINE OF THE LOGO LANGUAGE

PLUS

computer gaming™

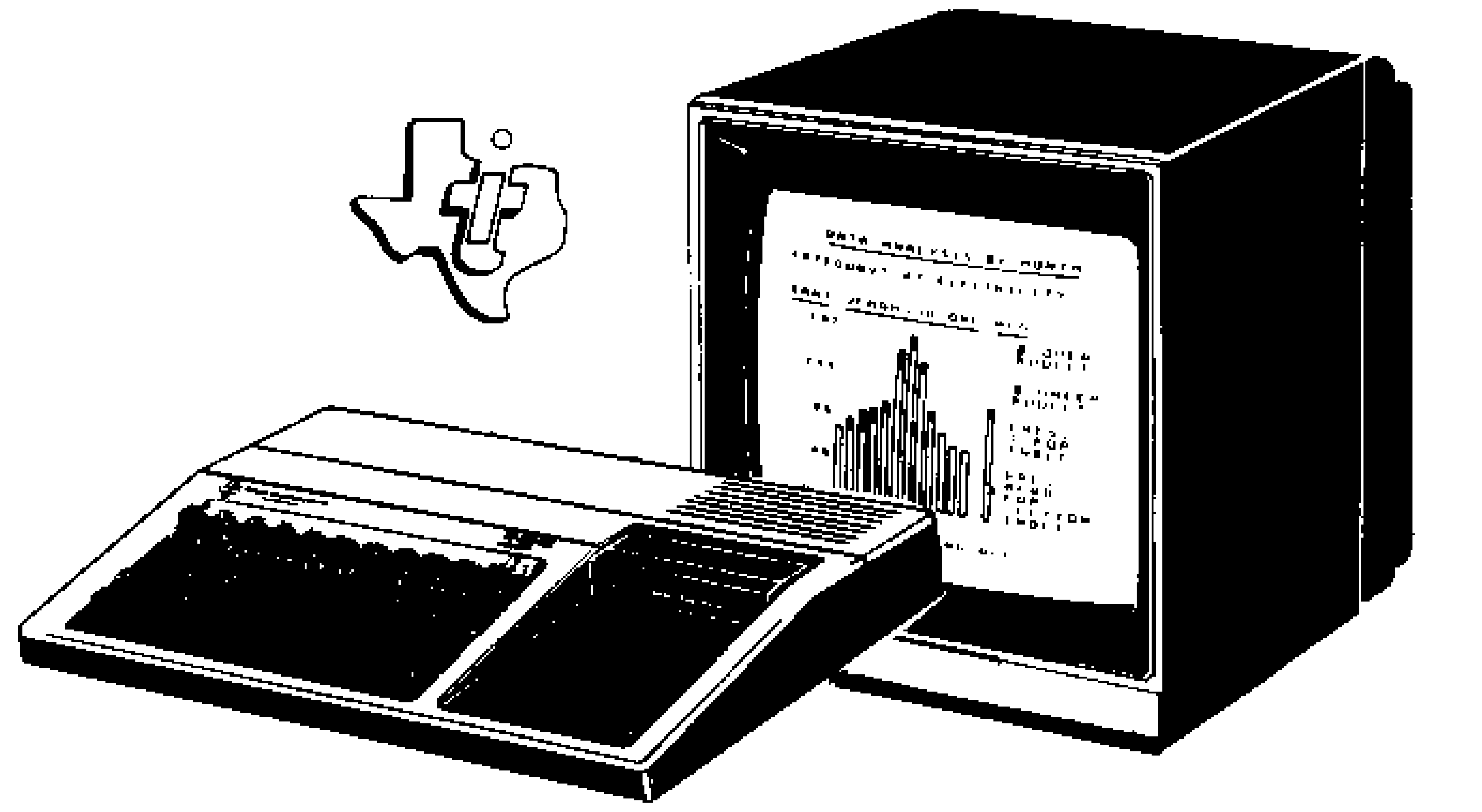


Essential Attributes Of A Home Computer:

A User-Friendly Design With Color, Music,
Animation, Speech, & Plug-In Software

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.

Texas Instruments

CONSOLE

PHC	004A	TI-99/4A with Free R. F. Modulator	**$299.95**

PERIPHERALS

PHP	1200	Peripheral Expansion Box**	**192.95**
PHP	1220	RS-232 Card**	**134.95**
PHP	1240	Disk Controller Card**	**192.95**
PHP	1250	Expansion Box Disk Drive Card**	**294.95**
PHP	1260	Memory Expansion Card** (RAM)*	**229.95**
PHP	1270	P-Code Card**	**192.95**
PHP	1500	Solid State Speech™ Synthesizer	**109.95**
PHP	1600	Telephone Coupler (Modem)	**164.95**
PHP	1700	RS-232 Accessories Interface**	**164.95**
PHP	1800	Disk Drive Controller[1] (One Disk Manager module packed with each Disk Controller)	**219.95**
PHP	1850	Disk Memory Drive	**367.95**
PHP	1900	Solid State Printer	**299.95**
PHP	2200	Memory Expansion[1] (32K RAM)*	**286.95**
PHP	2300	VCR Controller[2]	**539.95**
PHP	2400	P-Code Peripheral**	**297.95**
PHA	2100	R.F Modulator (TV Adapter)	**36.95**
PHA	4100	10" Color Monitor	**324.95**

OPTIONAL ACCESSORIES

PHP	1100	Wired Remote Controllers (Pair)	**26.95**
PHA	1950	Thermal Paper (2 Pack)	**9.95**
PHA	2000	Dual Cassette Cable	**12.95**
PHA	2010	Monitor Cable	**17.95**
PHA	2020	Audio Adapter (Headphone Jack)	**17.95**
PHA	2310	Panasonic VCR Controller Cable	**89.95**
PHA	2320	Sony VCR Controller Cable	**89.95**
PHA	2330	Pioneer VCR Controller Cable	**89.95**
PHA	2605	Blank Overlays (4 Pack)	**7.95**

APPLICATION PROGRAMS

Home Management/Personal Finance

		Command Module	
PHM	3006	Home Financial Decisions	**24.95**
PHM	3007	Household Budget Management (Data storage system is required)	**32.95**
PHM	3012	Securities Analysis	**45.95**
PHM	3013	Personal Record Keeping (Data storage system is recommended)	**39.95**
PHM	3016	Tax/Investment Record Keeping (Disk system is required)	**58.95**
PHM	3022	Personal Real Estate (Data storage system is recommended)	**58.95**
PHM	3044	Personal Report Generator (Data storage system is recommended)	**39.95**
		Diskette	
PHD	5001	Mailing List	**53.95**
PHD	5003	Personal Financial Aids	**16.95**
PHD	5021	Checkbook Manager	**16.95**
PHD	5022	Business Aids Library — Finance Management (Extended BASIC Command Module is required)	**33.95**
PHD	5024	Business Aids Library — Inventory Management (Personal Record Keeping or Statistics Command Module is required)	**58.95**
PHD	5027	Business Aids Library — Invoice Management (Personal Record Keeping or Statistics Command Module is required)	**58.95**
PHD	5029	Business Aids Library — Cash Management (Extended BASIC Command Module is required)	**33.95**
PHD	5038	Business Aids Library — Lease/Purchase Decisions	**58.95**
		Cassette	
PHT	6003	Personal Financial Aids	**12.95**
PHT	6038	Business Aids Library — Lease/Purchase Decisions	**54.95**

Education/Personal Enrichment

		Command Module	
PHM	3002	Early Learning Fun	**24.95**
PHM	3003	Beginning Grammar	**24.95**
PHM	3004	Number Magic	**16.95**
PHM	3005	Video Graphs	**16.95**
PHM	3008	Video Chess	**52.95**
PHM	3010	Physical Fitness	**24.95**
PHM	3015	Early Reading † (Solid State Speech™ Synthesizer is required)	**44.95**
PHM	3020	Music Maker (Data storage system is recommended)	**32.95**
PHM	3021	Weight Control and Nutrition (Data storage system is recommended)	**49.95**
PHM	3027	Addition and Subtraction I† (Solid State Speech™ Synthesizer is recommended)	**32.95**

APPLICATION PROGRAMS CONTINUED

Education/Personal Enrichment Continued

		Command Module	
PHM	3028	Addition and Subtraction II† (Solid State Speech™ Synthesizer is recommended)	**32.95**
PHM	3029	Multiplication I† (Solid State Speech™ Synthesizer is recommended)	**32.95**
PHM	3040	TI LOGO (Memory Expansion is required)	**74.95**
PHM	3043	Reading Fun†*	**45.95**
PHM	3059	Scholastic Spelling — Level III** (Solid State Speech™ Synthesizer is required)	**45.95**
PHM	3060	Scholastic Spelling — Level IV** (Solid State Speech™ Synthesizer is required)	**45.95**
PHM	3061	Scholastic Spelling — Level V** (Solid State Speech™ Synthesizer is required)	**45.95**
PHM	3062	Scholastic Spelling — Level VI** (Solid State Speech™ Synthesizer is required)	**45.95**
PHM	3064	Typing Tutor**	**32.95**
		Diskette	
PHD	5007	Teach Yourself BASIC	**27.95**
PHD	5009	Music Skills Trainer	**24.95**
PHD	5011	Computer Music Box	**16.95**
PHD	5018	Market Simulation	**16.95**
PHD	5019	Teach Yourself Extended BASIC (Extended BASIC Command Module is required)	**19.95**
PHD	5020	Music Maker Demonstration (Music Maker Command Module is required)	**12.95**
PHD	5023	Basketball Statistics (Extended BASIC Command Module is required)	**20.95**
PHD	5026	Bridge Bidding I	**24.95**
PHD	5030	Speak & Spell™ Program (Solid State Speech™ Synthesizer is required)	**24.95**
PHD	5031	Speak & Math™ Program (Solid State Speech™ Synthesizer and Terminal Emulator II are required)	**24.95**
PHD	5039	Bridge Bidding II	**24.95**
PHD	5041	Bridge Bidding III	**24.95**
PHD	5042	Spell Writer (Terminal Emulator II Command Module and Solid State Speech™ Synthesizer are required)	**24.95**
PHD	5067	Teach Yourself Beginning BASIC**	**24.95**
		Cassette	
PHT	6007	Teach Yourself BASIC	**24.95**
PHT	6009	Music Skills Trainer	**20.95**
PHT	6011	Computer Music Box	**12.95**
PHT	6018	Market Simulation	**12.95**
PHT	6019	Teach Yourself Extended BASIC (Extended BASIC Command Module is required)	**15.95**
PHT	6026	Bridge Bidding I	**19.95**
PHT	6031	Speak & Math™ Program (Solid State Speech™ Synthesizer and Terminal Emulator II are required)	**20.95**
PHT	6039	Bridge Bidding II	**19.95**
PHT	6041	Bridge Bidding III	**19.95**
PHT	6042	Spell Writer (Solid State Speech™ Synthesizer and Terminal Emulator II Command Module are required)	**20.95**
PHT	6067	Teach Yourself Beginning BASIC**	**20.95**

Entertainment

		Command Module	
PHM	3009	Football	**23.95**
PHM	3018	Video Games I	**23.95**
PHM	3023	Hunt the Wumpus	**19.95**
PHM	3024	Indoor Soccer	**23.95**
PHM	3025	Mind Challengers	**20.95**
PHM	3030	A-Maze-Ing	**19.95**
PHM	3031	The Attack††	**31.95**
PHM	3032	Blasto††	**19.95**
PHM	3033	Blackjack and Poker††	**19.95**
PHM	3034	Hustle††	**19.95**
PHM	3036	ZeroZap††	**15.95**
PHM	3037	Hangman††	**15.95**
PHM	3038	Connect Four††	**15.95**
PHM	3039	Yahtzee††	**19.95**
PHM	3041D	Adventure (Pirate Adventure Diskette Game Included)	**38.95**
PHM	3041T	Adventure (Pirate Adventure Cassette Game Included)	**38.95**
PHM	3052	Tombstone City, 21st Century	**31.95**
PHM	3053	TI Invaders	**31.95**
PHM	3054	Car Wars	**31.95**
PHM	3057	Munch Man**	**31.95**

APPLICATION PROGRAMS CONTINUED

Entertainment Continued

		Diskette	
PHD	5002	TI-Trek (with speech)	**12.95**
PHD	5010	Mystery Melody	**12.95**
PHD	5015	Oldies But Goodies — Games I	**16.95**
PHD	5017	Oldies But Goodies — Games II	**20.95**
PHD	5025	Saturday Night Bingo (Solid State Speech™ Synthesizer is required)	**24.95**
PHD	5037	Draw Poker	**20.95**
		Adventure Series (Adventure Module is required)	
PHD	5043	Pirate Adventure	**26.95**
PHD	5046	Adventureland	**23.95**
PHD	5047	Mission Impossible	**23.95**
PHD	5048	Voodoo Castle	**23.95**
PHD	5049	The Count	**23.95**
PHD	5050	Strange Odyssey	**23.95**
PHD	5051	Mystery Fun House	**23.95**
PHD	5052	Pyramid of Doom	**23.95**
PHD	5053	Ghost Town	**23.95**
PHD	5054	Savage Island I & II	**31.95**
PHD	5056	Golden Voyage	**23.95**
		Cassette	
PHT	6010	Mystery Melody	**9.95**
PHT	6015	Oldies But Goodies — Games I	**12.95**
PHT	6017	Oldies But Goodies — Games II	**16.95**
PHT	6025	Saturday Night Bingo (Solid State Speech™ Synthesizer is required)	**20.95**
PHT	6037	Draw Poker	**16.95**
		Adventure Series (Adventure Module is required)	
PHT	6043	Pirate Adventure	**23.95**
PHT	6046	Adventureland	**23.95**
PHT	6047	Mission Impossible	**23.95**
PHT	6048	Voodoo Castle	**23.95**
PHT	6049	The Count	**23.95**
PHT	6050	Strange Odyssey	**23.95**
PHT	6051	Mystery Fun House	**23.95**
PHT	6052	Pyramid of Doom	**23.95**
PHT	6053	Ghost Town	**23.95**
PHT	6054	Savage Island I & II	**31.95**
PHT	6056	Golden Voyage	**23.95**

OTHER APPLICATION PROGRAMS

		Command Module	
PHM	3000	Diagnostic	**24.95**
PHM	3001	Demonstration	**59.95**
PHM	3011	Speech Editor (Solid State Speech™ Synthesizer is required)	**34.95**
PHM	3014	Statistics (Data storage system is recommended)	**34.95**
PHM	3026	Extended BASIC	**74.95**
PHM	3035	Terminal Emulator II	**38.95**
PHM	3055	Editor Assembler	**79.95**
PHM	3058	Mini-Memory	**79.95**
		Diskette	
PHD	5004	Programming Aids I	**12.95**
PHD	5005	Programming Aids II	**19.95**
PHD	5006	Math Routine Library	**24.95**
PHD	5008	Electrical Engineering Library	**24.95**
PHD	5012	Programming Aids III	**15.95**
PHD	5013	Graphing Package	**16.95**
PHD	5016	Structural Engineering Library	**24.95**
PHD	5044	SMU Circuit Analysis I**	**124.95**
PHD	5063	UCSD-Pascal[3] Compiler**	**99.95**
PHD	5064	UCSD p-System[3] Assembler Linker**	**79.95**
PHD	5065	UCSD p-System[3] Editor Filer Utilities**	**64.95**
PHD	5066	TI PILOT**	**67.95**
		Cassette	
PHT	6004	Programming Aids I	**8.95**
PHT	6006	Math Routine Library	**20.95**
PHT	6008	Electrical Engineering Library	**20.95**
PHT	6013	Graphing Package	**12.95**
PHT	6016	Structural Engineering Library	**20.95**
PHT	6044	SMU Circuit Analysis I**	**124.95**

*Requires use of Extended BASIC or TI LOGO

**Available Second Quarter 1982.

† Developed by Scott Foresman

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

[1] Available only until replaced by peripheral card

[2] Pending FCC approval, available Second Quarter 1982.

[3] UCSD, UCSD Pascal and UCSD p System are all trademarks of the Regents of the University of California

tam's

INCORPORATED

14932 GARFIELD AVENUE
PARAMOUNT, CA 90723

VISA MasterCard

Order Toll-Free

800-421-5188

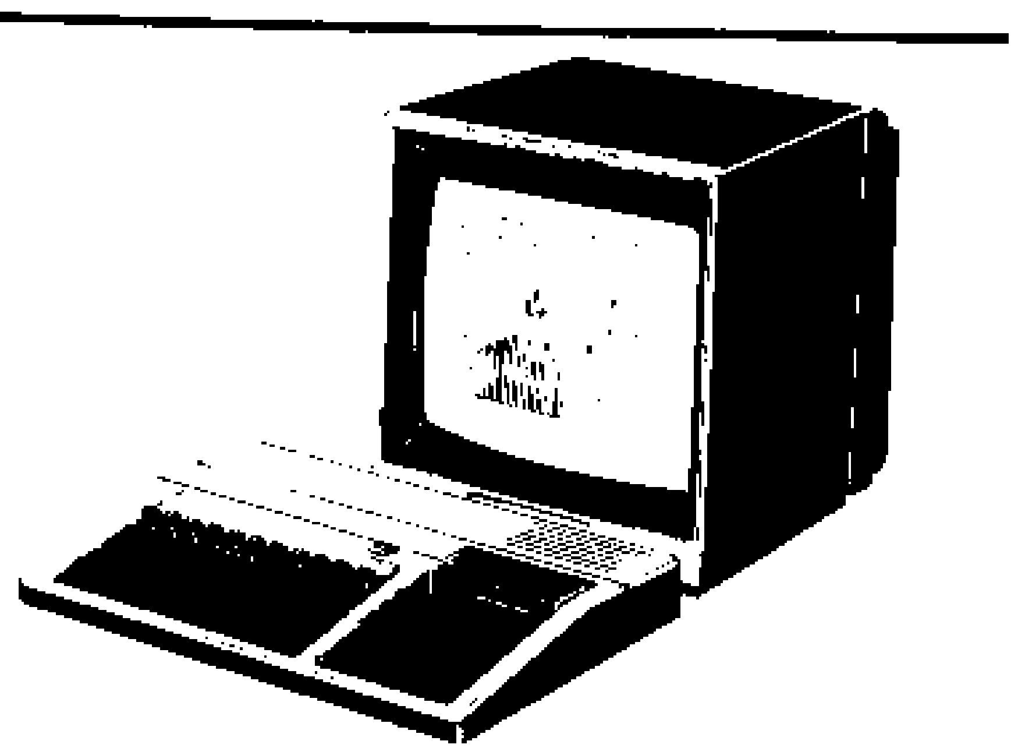
In California call (213) 633-3262

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 ($3.75 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 Home Computer



WE STOCK EVERYTHING MADE BY TEXAS INSTRUMENTS FOR THE 99/4(A)

And lots of 3rd party items also

COMPLETE UNIT FOR USE WITH YOUR TV (BW or color) PHC004A Computer includes the following

- Console 99/4A
- PHA 2100 Video Modulator (TV Adaptor)
- PHA 2600 Beginning Basic Manual
- PHA 2601 Users Reference Guide

$295⁹⁵

OUR MOST POPULAR PACKAGE

- Console 99/4A
- PHA 2100 Video Modulator (TV Adaptor)
- PHA 2600 Beginning Basic Manual
- PHA 2601 Users Reference Guide
- PHA 2000 Cable to connect 1 or 2 cassette recorders

$306 95

48K PERIPHERAL EXPANSION PACKAGE (Adds 32K RAM)

- PHP 1200 THE NEW PERIPHERAL EXPANSION BOX and
- PHP 1260 Memory Expansion Card

$379 95

- Above plus choice of any one of the following:
  PHM 3055 Assembler Editor
  PHM 3058 Mini Memory
  PHM 3040 Logo

$448 95

Add $2.98 shipping and handling any size order in Continental U.S. Iowa residents add 3% sales tax. No extra charge for VISA or Master Charge.

COUPON

FREE 1 Year Subscription to the COMPUTER BULLETIN

COUPON 6/99er

Name ____________________

Street ____________________

City ____________ State ____________ Zip ________

COUPON

A 20 page newspaper tabloid size, just for the 99/4[A] Computer

COUPON

DHEIN'S

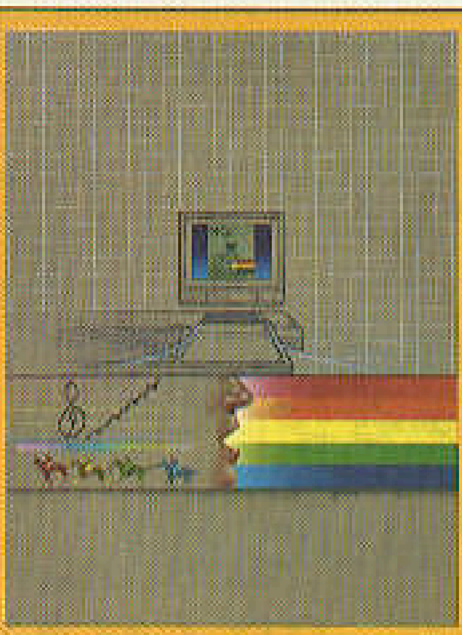
True Value HARDWARE STORE ®

(319) 236-3861

7 W. Airline Hwy.
Waterloo, IA 50701

99'er magazine™

PAGES FROM ON LOCAITION™
THE INTERNATIONAL JOURNAL OF COMPUTER ASSISTED INSTRUCTION



THIS ISSUE'S COVER

Hayder Amir's cover painting depicts the attributes that are now essential in a computer designed for home use. The artist's mechanical rendering of the machine is in stylistic harmony with the computer-generated color animation sequence and musical notation. This is contrasted with the visual realism in the illustration's portrayal of computer-synthesized human speech. The audio-visual attributes are all shown emanating from a plug-in, modular software port—a user-friendly feature designed as an integral part of a true Home Computer.

Publisher/Editor
Gary M. Kaplan

Technical Editors
William K. Balthrop
David G. Brader
John Clulow
Mike Kovacich
Patricia Swift

Program Editors
Cheryl Whitelaw
G. R. Michaels

Office Manager
Pat Kaplan

Circulation & Fullfillment
Benjamin Kaplan
Coleen Nelson

Advertising
Patana Ratanapreux
Tel. 503-485-8796

Contributing Editors
F. T. Berkey
Norma Clulow
Henry Gorman, Jr.
Roger Kirchner
Ira McComic
Samuel Pinćus
Regena
George Struble
Jerry Wolfe

Accounting
Tasanee Fry

Production Manager
Norman Winney, Jr.

Production & Design
June Gaber
Paul Olson
Corby Poticha
Alice Sundstrom

Programming Conventions

KEY-IN REFERENCE

```
@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_
 !"#$%&'()*+,-./0123456789:;<=>?
```

⊗ = Program as listed will completely f available memory of TI-99/4A ar cannot be RUN with disk controll (and possibly RS232 Interface) turn on. It must be SAVEd and RU from cassette.

● = End of Program or Article

computer gaming™

THE MAGAZINE OF LOGO TIMES THE LOGO LANGUAGE™

A Resource for People Interested in the Enrichment of Personal Computing

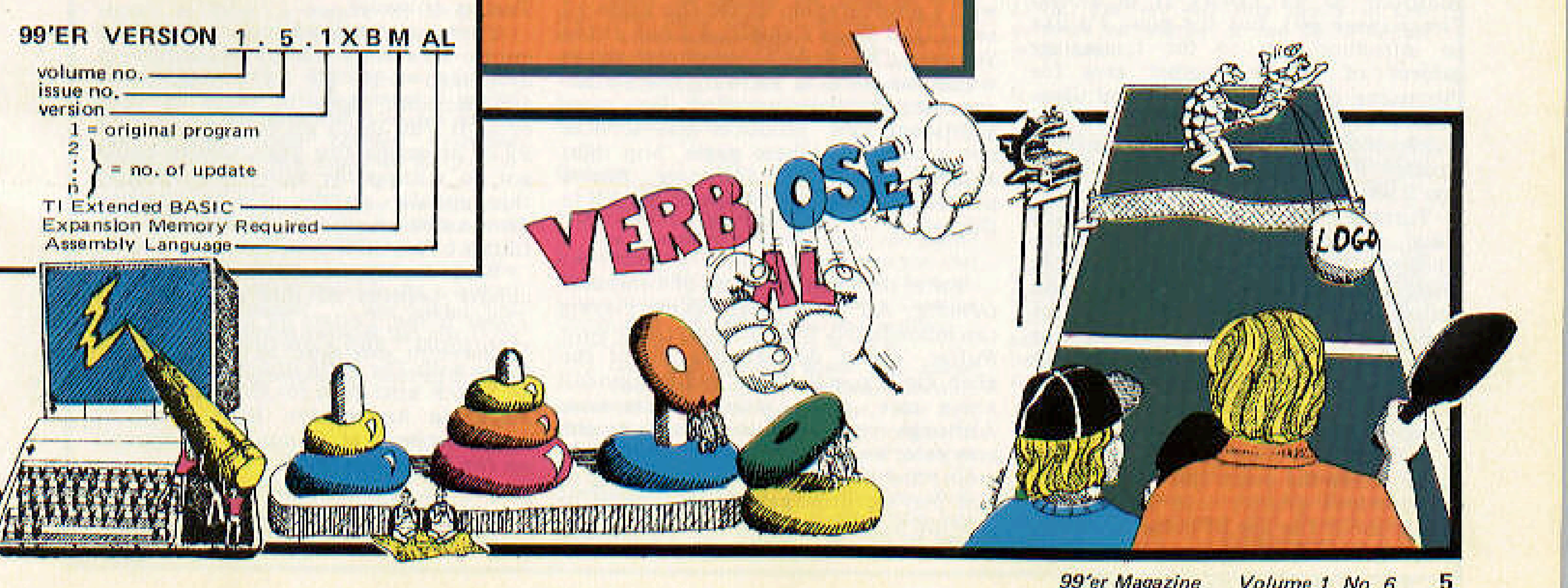
Volume 1 No. 6

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 $18 for one year, $32 for two years, and $45 for three years. In Canada and Mexico $22 for one year, $42 for two years, $60 for three years. Other foreign countries $28 for one year surface, $43 for one year air delivery. Single copy price in U.S. and its possessions is $3.50, and $4.00 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.

99'ER VERSION 1 . 5 . 1 X B M AL

volume no.
issue no.
version
1 = original program
2 . . n } = no. of update
TI Extended BASIC
Expansion Memory Required
Assembly Language



ON SCREEN

By Gary M. Kaplan
Editor & Publisher

With what do you follow a first year of themes that included such topics as telecommunications, information management, gaming, and decision making? All these themes are, of course, applications for a home computer, so it would have been only natural to pick another broad application—education, perhaps . . . But as readers of this magazine well know by now, each of the past five issues has been chock full of educational applications. So something else was definitely called for in this end-of-first-year issue.

As you can now obviously see from the cover art, we've decided to focus on the important capabilities of the TI Home Computer system that sets it apart from all the other contenders—most notably its superior color graphics, animation, music, sound effects, and synthesized speech. It was hard work—but undeniable fun—putting together an issue with a theme as broad and encompassing as this one; I hope *you* enjoy and learn from it as much as *we* enjoyed and learned from putting it together . . .

Our leadoff article, *Livening Up Your Call SOUNDs*, is an interactive tutorial. By calling it "interactive," I'm suggesting that everyone with a computer should type in the few short lines for each sound effect, and then experiment by modifying them. You'll have a lot of fun, and will benefit from the sound advice presented in the article.

The mere mention of *sound* causes us to recognize the great musical capabilites of the TI-99/4A system—with its score of TI and *99'er* programs already in use worldwide. You'll read more about the new, impressive musical additions to TI LOGO II in *LOGO Times* (page 69), but for now, I'd like to introduce you to the fascinating subject of speech—another area for "sounding off" on the advanced capabilities of the Home Computer. The article and associated utility program is entitled *Verbose* for good reason . . . as you'll soon discover.

Turning from the audio realm to the visual, we're now ready to do some *Dynamic Manipulation of Screen Character Graphics*. Don't let this imposing title scare you off. There are really some innovative, time-saving programming ideas contained in this article—not to mention the impressive multi-color screen displays accompanying the text.

Getting your color graphic characters up on screen the quick and easy way (of the previous article) lead us to wondering about the use of color to convey meaning to hard-to-digest numerical data. See the statistically-significant result for yourself in *Color Mapping*.

So far, all the feature articles have been centered around the TI BASIC language—the built-in, high-level (English-like) standard language with which most of us first start communicating with our computers. But lest we totally ignore low-level languages, we've included, for the more experienced users, an Assembly Language tutorial on accessing Multicolor Mode—complete with the source coding for a fun-to-use *Magic Crayon*.

Color, music, sound effects, and speech capabilities are so easy to access on the TI-99/4A, that even beginners can soon be producing impressive results. One area, however, that hasn't previously been well explained is program and file storage on cassette. *The Beginner's Guide to Cassette Operation with the Home Computer* will fill this information void.

After beginners learn the ropes (or is it *tapes?*) with cassette recorder operation, they're usually ready to start learning a lot more about computer operation and programming. One excellent way of doing this is through association with others of similar interests. If you'd like some help with finding others in your area to participate, be sure to read, *How To Start a User's Group*.

There really are many extremely talented people who are active in user's groups—including quite a few of the authors published in *99'er Magazine*. This fact brings to mind the frequently-asked question, "How do these articles and programs make it into *99'er*?" For the answer to this, read *An Interview With A 99'er Program Editor*. You'll also pick up valuable tips on how to type in the programs *accurately*, without those frustrating, hard-to-find typos . . .

Before entering our very own microworlds of magazines-within-a-magazine, we round out our standard *99'er* fare with a photographic spread (on pages 12 and 13) of *TI on Exhibit: A Look at the Summer CES & NCC*—computer shows where the latest in software and peripherals are publicly unveiled. For some additional new products and services not covered on these pages, hop right aboard the *99'er Shopping Bus*—a new feature that I'm happy to launch in this issue.

Enter the exciting world of *Computer Gaming*. All you hungry game players can immediately byte into our *Gameware Buffet*, where two encounters of the alien kind, as well as an "old-fashioned" horse race, await your participation. Although you'd get to experience the computer's repertoire of sound effects, color, and animation just by typing in and playing these games, we wanted to provide you with a solid understanding of how the color and animation really work (sound will be explained in an upcoming issue). So if you're still a little hungry after the Buffet, munch on a sprite sandwich in *3-D Animation With the TMS9918A Video Chip*.

Then, when you're ready to learn how to write your own games, pick up our *Chuck-A-Luck* series and follow along. Part 3 of this series is published in this issue. But if programming and debugging aren't of interest right now, you can still type in the full TI BASIC program listing, and have fun playing the dice game. The final installment in next issue explains the graphics and provides the Extended BASIC "exploded dice" version of the program using sprites.

To round out your *Computer Gaming* fun, and to learn quite a lot in the process, be sure to read through our regular features, *Designer's Spotlight*, *Arcade Arbiter's Review*, and *Adventure Registry*. Then take a look at the two new departments making their debut in this issue—*Strategy Corner* and the *99'er Hall of Fame*.

Next, we'll go *OnLoCAItion* to visit Colorado Springs for a review of the TI *Course Designer Authoring System*, as implemented at the Air Force Academy. If you're interested in CAI video-based courseware development, don't miss reading all about it here.

We haven't forgotten the preschoolers either—or, for that matter, any programmers interested in developing special character sets. Find out what these two groups have in common when you read *Pre-School Block Letters*.

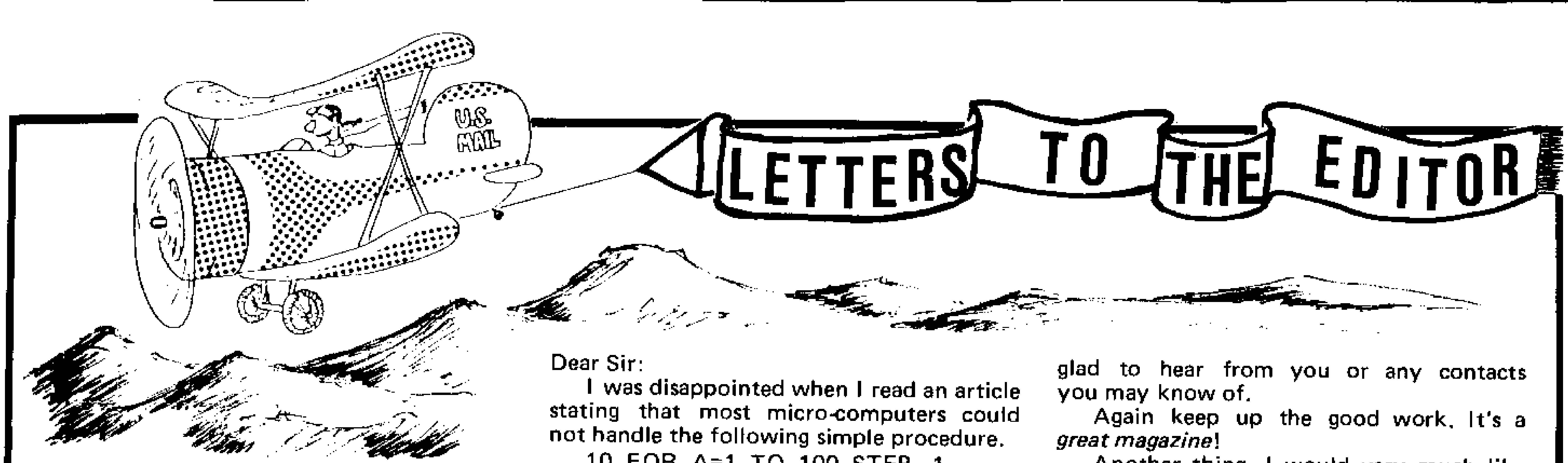
As our journey of learning and discovery takes us from *OnLoCAItion* into *LOGO Times*, we are immediately submerged into the "inner workings" of a game that, in fact, launched the video revolution. Find out why there's more to PONG than meets the paddle . . . in our leadoff LOGO article, *Who Is LOGO For? It's Not Just For Turtles Anymore . . .*

Then, as promised in last issue, the entire *Tower of Hanoi* procedure is developed—with all its sophistication and elegance—right in front of your eyes. If you don't get to see any other *99'er* programs this year, you've really got to witness the auto-solve mode of this popular puzzle. It's an exciting demonstration of the power and versatility of TI LOGO in action.

And finally, rounding out the recursive content of this issue's *LOGO Times* is our *Letters on LOGO* "reader reportorial," and a preview of things to come with *The Birth of a New LOGO*.

Watch next time for the special *99'er Magazine* Anniversary Issue, with an in-depth look at *Computer Languages on the Home Computer*.

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



LETTERS TO THE EDITOR

Dear Sir:

I know you've received at least dozens of letters commending you and your staff on an AWESOME magazine. Well here's another. Congratulations!

I have also written to offer some notes that other readers might be interested in. These deal with the *Home Secretary* program (Vol. 1 No. 2).

I found that I needed a "pause" feature when dialing. For example: accessing an outside line (dial "9" and wait for the tone) and when using a long-distance service such as MCI.

I needed to have the computer dial the number, wait out the necessary pause, and then continue dialing the number I am trying to reach. All I needed to do is add 3 lines as follows:

```
3495 IF ASC(TMP$)=80 THEN 3605
3605 CALL SOUND(1500,44000,30)
3606 GOTO 3600
```

This will generate a 1-½ second pause whenever a "P" is encountered. Thus when you enter the phone # in the program, type in a "P" wherever you need a pause. For example: 9 P 827-5309.

I soon ran into a problem though. If your # is more than 10 characters in length, the program put a "1" in front to dial long distance. This problem is then solved by deleting lines 3430, 3440, and 3450. The only inconvenience is that you will need to include the "1" when you enter the number into your file. I don't really see it as a problem since the string will hold up to 112 characters. I include my entire MCI # in mine. (651-1451 P XXXXX 714 8275309). Once again, congrats on a fine magazine.

Robert MacKay
Dallas, TX

Dear Sir:

In John Clulow's description of TI's *Programming Aids III* in the September/October issue, he implies that line re-numbering is not possible without the Editor in *PA III*. Actually, while not specifically stated as being a feature of Extended BASIC, the use of the REDO (shift R) key to change a line number is shown in the program entry example on page 32 of the Extended BASIC manual. I would have missed this point also if it were not for the program *Teach Yourself Extended BASIC* where it was explained in Chapter One.

I was at first perturbed by the fact that the only way I could get off-the-shelf delivery of Extended BASIC was to get it bundled with *Teach Yourself...* in the "Super Programmer" package. As it turned out, *Teach Yourself...* was useful not only for its intended purpose but also from the standpoint of being able to examine the programming techniques used to present the material.

Al Kanda
Scottsdale, AZ

Thanks for your discovery, Al. In an upcoming issue we will be having a review of both "Teach Yourself..." BASIC courses.

Dear Sir:

I was disappointed when I read an article stating that most micro-computers could not handle the following simple procedure.

```
10 FOR A=1 TO 100 STEP .1
20 PRINT A
30 NEXT A
```

I tried the procedure on the following micro-computers: Apple, Atari, Ohio Scientific, Pet, Radio Shack Model 3, Radio Shack Color and TI-99/4. Only Atari and TI printed the correct list of numbers. I then discovered another simple procedure which ran properly *only* on the TI. Here it is:

```
10 PRINT "BEGIN"
20 FOR A=1 TO 0
30 PRINT A
40 NEXT A
50 PRINT "END"
```

Every micro-computer *except* the TI, printed a 1 between the BEGIN and END when the FOR-NEXT loop was not true.

Walt Dollard
Pittsburgh, PA

Dear Sir:

I have owned my TI-99/4 about 1½ years, and even with the vast amount of information provided by your magazine from V1, No. 1, it is still quite unclear to me many of the associations of hardware and software which must be assembled to produce a given capability, for example:

a) Speech
b) Printing with the thermal printer or with an accessory printer
c) LOGO
d) Assembly Language programming and operation

It seems to me that many of these mysteries would be resolved if you could invent a master "family tree," flow chart, or bubble chart which would depict all of the interrelationships of all of the materials offered—hopefully by TI and by third parties alike. Each time a new product is described, it then becomes possible for you to show how the device (or whatever) fits into the overall scheme of things, and especially if it obsoletes something.

Ralph Patterson
San Diego, CA

Thanks for taking the time to write that long (only part of which is printed here), constructive letter, Ralph. The idea you describe above is particulary intriguing —so intriguing, in fact, that we've already begun work on its implementation. Watch for it in next issue.

Dear Sir:

I am a programmer/anaylst and I just subscribed to your magazine.

The question is where have you been? I was looking for just the magazine you and your staff publish. I must say it is the *best*. I have subscribed to Byte, Softside, Creative Computing and Kilabaud, and nothing—I mean *nothing*—can compare to 99'er.

I must feel this way about your magazine because I am a TI-99/4A console owner and am very happy that I did purchase this great computer. I've looked at them all, but I found this the best for what it has. With your magazine and the TI computer, I'd like to start my own small business here in Jersey. If you have any suggestions I can follow, I would be very glad to hear from you or any contacts you may know of.

Again keep up the good work. It's a *great magazine*!

Another thing. I would very much like to push your great magazine. I was thinking of starting a user's club for TI owners. Got any suggestions?

Sal Melillo
Scotch Plains, NJ

Thanks for the kind words, Sal. Since, you are looking for business opportunities, we'll be sending you information on becoming a dealer/distributor of 99'er, as well as a media kit with advertising on the chance that you have written some software that you have considered selling to fellow readers. As for the user's club, see the related article on how to start one in this issue.

Dear Sir:

You can put my name among the list of those who avidly await each issue of *99'er*. I can't wait for it to be a monthly publication! Your music programs have been a dream come true for me and my 50 piano students. *Let's Learn Notes* by the mysterious Regena is a hit with beginners and the programs that Norma and John Clulow have written are super. And I am working with these programs and the TI speech synthesizer so that my two blind students can use them! (They are already using the *Music Skills Trainer.*)

Please keep up the music-learning entries. I am curious about the "Rhythm" program mentioned in the Clulow article. (Vol. 1, No. 4). Music teachers could use programs teaching scales, key signatures, intervals (visual), etc. If I can get any written successfully, I'll certainly send them in. Meanwhile I hope that you or your advertisers can come up with additional music-learning programs. They are a great boon to making the teaching and learning of music fun and exciting.

P.S. Will this type of program be available for viewing at the TI-Fest in San Francisco?

Lolita W. Gilkes
Plano, TX

Glad you like the music programs, Lolita. Yes, we will have plenty of CAI software on display at the 99'er TI-Fest in October.

Dear Sir:

In *Now What?* by Regena [Vol. 1, No. 3 pg. 36] "Sorting" in column 2 should be renamed "Shuffling." Numbers are shuffled —not sorted. To verify, change as follows:

```
340 BREAK
350 GOTO 190
```

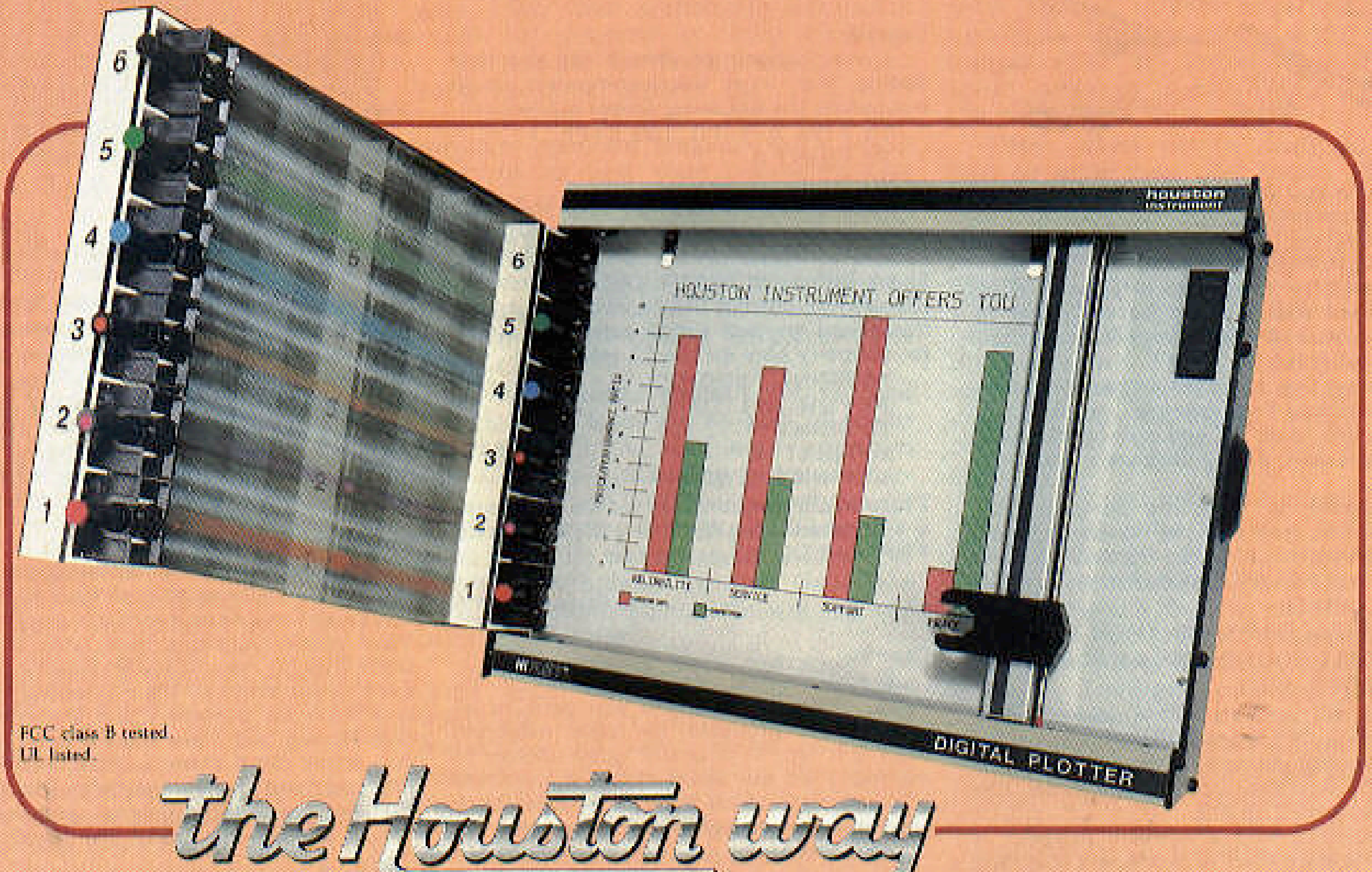
After each printout and BREAK, the order of numbers can be read. Re-run using CON (ENTER) and read the list of numbers again. The order in sequence is better with each RUN. Sorting of nine digits in sequential order is accomplished in approximately five reRUNs using CON on the 99/4A.

W. C. Dale
Albany, OR

I believe the difference between "shuffling" and "sorting" is that shuffling means mixing up. It appears that Regena meant

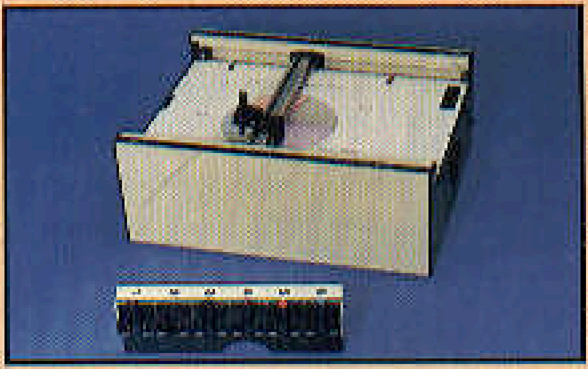
Continued on p. 15

New from HIPLØT™ multi-pen plotting for as little as $1480*.



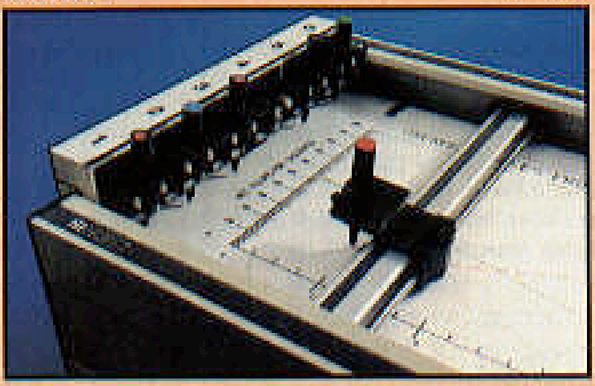
FCC class B tested.
UL listed.

The new HIPLØT DMP Series 6-pen option makes high performance multi-pen plotting affordable. It's available on the DMP 2, 3, and 4 models in the HIPLØT family so you can enjoy the advantages of multi-colored plots on 8½" × 11" (DIN A4) surfaces. Of course, you also get the standard HIPLØT range of capabilities such as intelligence, controls, interfaces and resolutions. There's a model for virtually every plotting application.



Big Performance in a Small Plotter

Since it's introduction, the HIPLØT DMP Series has been recognized as the innovative plotter line which made low-cost, high performance digital plotting a reality.



Now, with our new 6-pen option, there's an exciting new dimension in the DMP Series' versatility. Imagine two standard models with RS-232-C and parallel interfaces, four intelligent models with RS-232-C or Centronics®compatible interfaces, a choice of controls, resolutions, and pen speeds. Add to this the ability to plot with 6-pens on paper, vellum or mylar (ideal for overhead projectors) and you have the ultimate plotter price/performance combination — the perfect choice for the user or OEM.

8-Pen Models Also Available.

If you need a little more capability, take a look at our new 8-pen option. It's available on the DMP 5, 6, and 7 so you can have 8-pen multi-colored plots on 11" × 17" (DIN-A3) surfaces.

Why wait? Let us send you complete information on this breakthrough in **affordable, multi-pen plotting.** *Contact Houston Instrument, P.O. Box 15720, Austin, Texas 78761. (512) 835-0900. For rush literature requests, outside Texas, call* **toll free 1-800-531-5205.** *For technical information ask for operator #5. In Europe contact Houston Instrument, Rochesterlaan 6, 8240 Gistel, Belgium. Phone 059/27-74-45.*

INSTRUMENTS & SYSTEMS DIVISION
Together...we'll create tomorrow.

BAUSCH & LOMB

TM Trademark of Houston Instrument.
* U.S. Suggested retail prices.
Centronics® registered trademark of Centronics Data Computer Corp.

DON'T MISS A SINGLE ISSUE!

Please Use This Card To Renew Your Subscription

□ YES – Please Renew My Subscription to 99'er magazine™

PLEASE ATTACH OR PRINT ALL INFORMATION FROM MAILING LABEL

Term	U.S.A.	Canada & Mexico	Foreign Surface	Foreign Air
1-yr (6 issues)	□ $18	□ $22	□ $28	□ $43
2-yr (12 issues)	□ $32	□ $42		
3-yr (18 issues)	□ $45	□ $60		

ENCLOSED IS MY PAYMENT OR CREDIT CARD BILLING INFORMATION

□ Check enclosed MUST BE IN U.S. FUNDS DRAWN ON A U.S. BANK

Bill my:

□ VISA

□ Master Charge

Expiration Date ____________

Signature ____________

Account No □□□□□□□□□□□□□□□□

MC Only – List 4 digits above your name □□□□

My Subscriber Code Number Is: ____________

(line above your name from mailing label)

NAME ____________

ADDRESS ____________

____

____

PLEASE INDICATE BELOW ANY CHANGE IN NAME OR ADDRESS

NAME ____________

ADDRESS ____________

____

____

PLEASE USE OTHER ORDER CARD FOR ORDERING NEW SUBSCRIPTIONS, BACK ISSUES, AND MERCHANDISE.

____

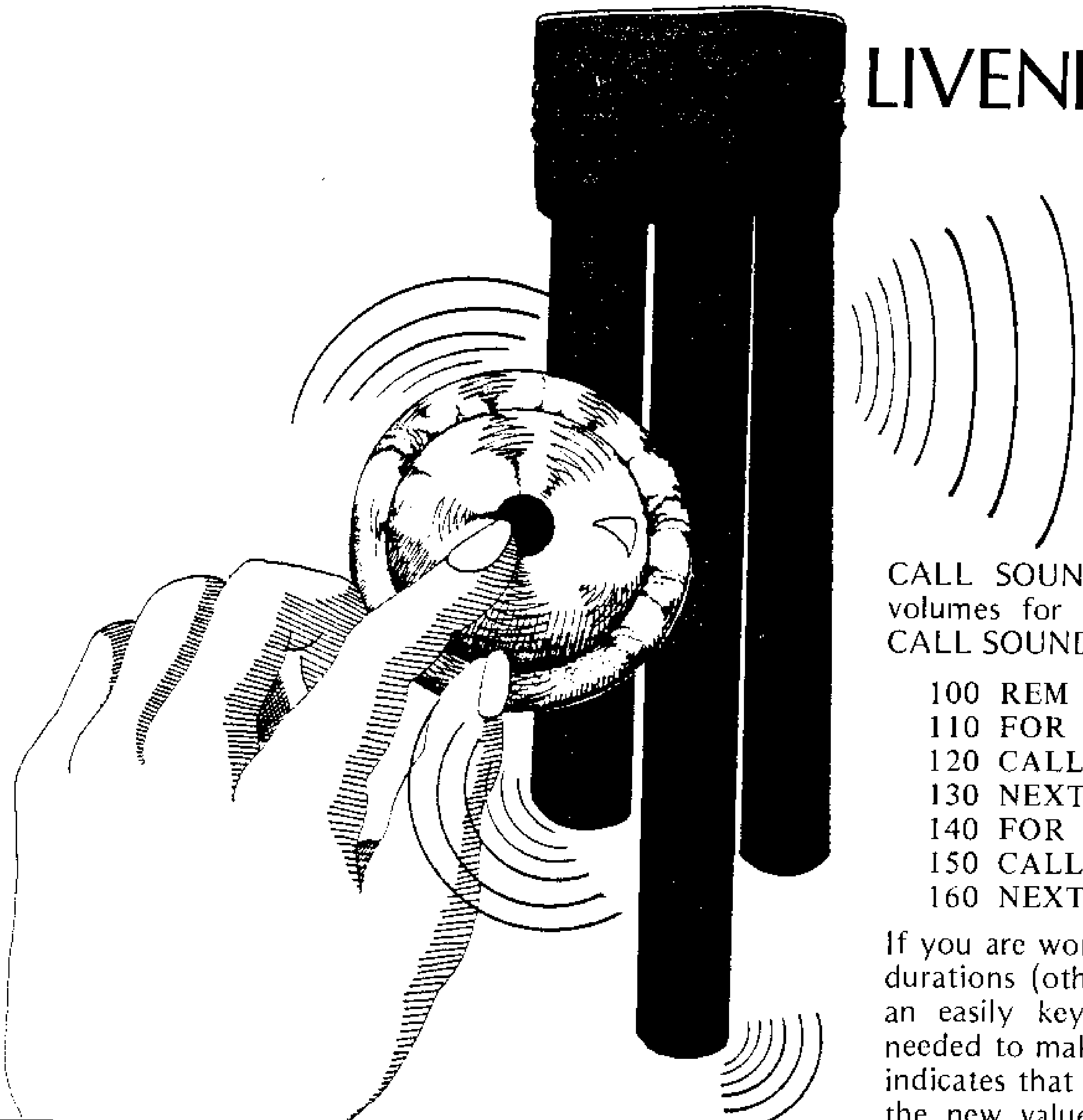
____

____

PLACE STAMP HERE

99'er magazine™

P.O. Box 5537
Eugene, OR 97405



LIVENING UP YOUR CALL SOUNDs

By: Al Kanda
Box 3494
Scottsdale, AZ 85257

The CALL SOUND subprogram in TI BASIC commands an amazing integrated circuit in your TI-99/4A, called the SN76489 Sound Generation Controller. On a single chip, TI has squeezed in three programmable frequency dividers, a programmable noise generator, four programmable attenuators (volume controls), and eight registers to hold the data that control the tones, noise, and their volume levels. In effect, the tones and noise are synthesized to your specifications from a frequency of 3.58 megahertz; this is also the frequency that carries the color information from your computer to your color monitor or video modulator.

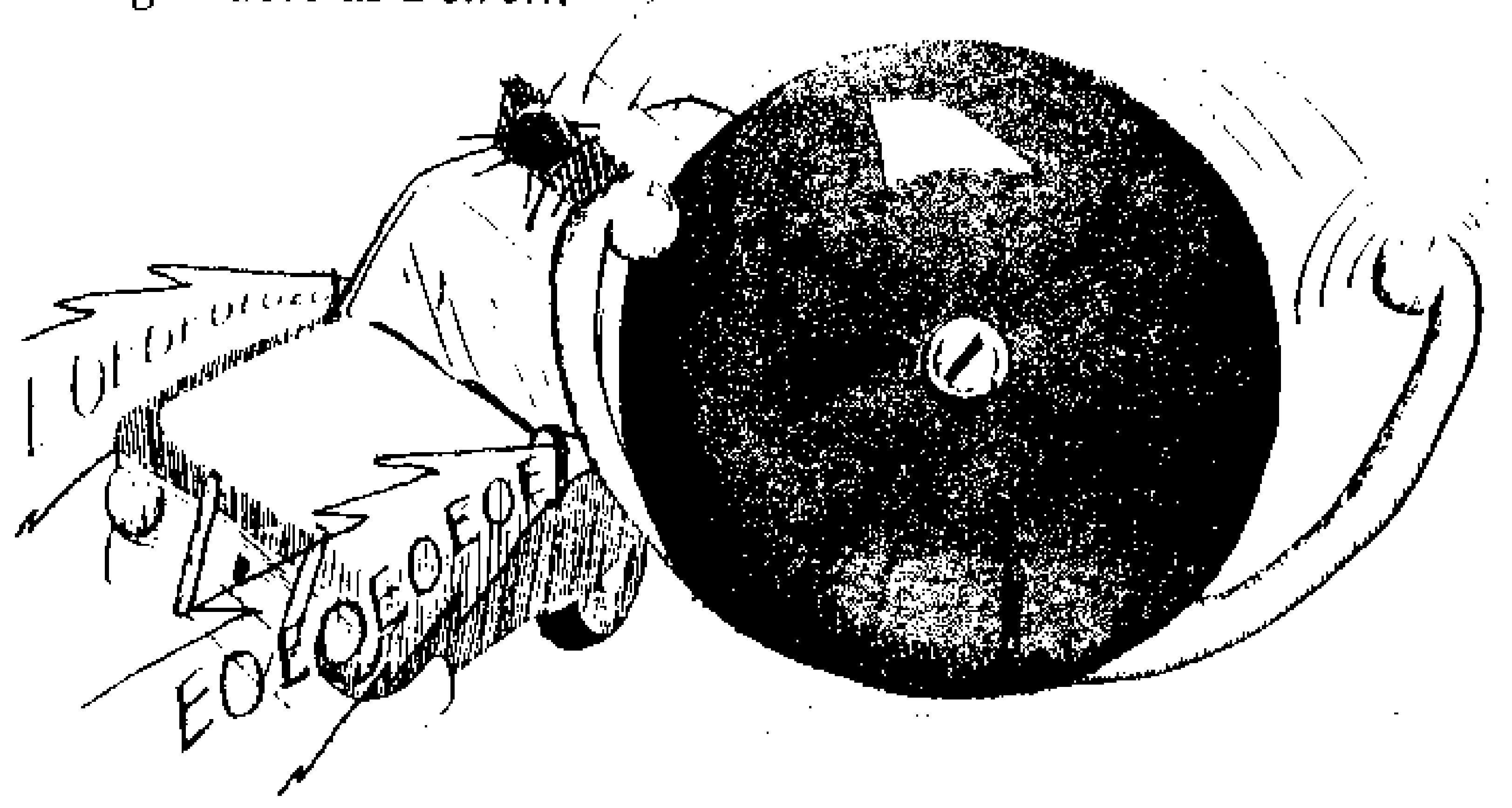
If the only use you have made of CALL SOUND has been to produce miscellaneous beeps, noise, and music, read on. I'm going to give you some "mini programs" that demonstrate the variety of other sounds your 99/4A is capable of producing.

For the first example, let us try to recreate the sound of a door bell of the type associated with the once popular "Avon Calling" commercial. This is an example of an object that is struck with a sharp blow and allowed to vibrate at its resonant frequencies. The following characteristics are needed to recreate this sound: 1) the fundamental frequencies of the two tones, 2) the overtone frequencies, and 3) a gradually decaying volume. Those of you with a sense of absolute pitch would immediately recognize the two fundamental frequencies, but in my case, I actually measured the dimensions of the sounding bars, their points of support, and determined with a magnet that the bars were probably steel. From a textbook, *Acoustical Engineering* by Harry F. Olson, I obtained the formula and values of the constants needed to calculate the resonant frequencies of the bars. The calculated frequencies came out to be very close to 698 and 554 cycles per second (F and C# above high C). The book also told me that the two closest overtones were 2.756 and 5.404 times the fundamental frequency. The bars were supported on rubber mounts close to the theoretical nodes (points of minimum vibration) for the fundamental and the first overtones, but were located near points of maximum vibration for the second overtone. I therefore assumed that the second overtone would be dampened out, so I omitted it from the CALL SOUND specification for each tone. The decaying volumes for the tones were obtained by including each CALL SOUND in a FOR–NEXT loop as follows:

```
100 REM DOOR CHIMES
110 FOR A=0 TO 30 STEP 5
120 CALL SOUND(–99,698,A,1924,A)
130 NEXT A
140 FOR A=0 TO 30 STEP 5
150 CALL SOUND(–99,554,A,1527,A)
160 NEXT A
```

If you are wondering about the significance of the 99 for the durations (other than this being a 99'er article), it is simply an easily keyed number larger than the 50 milliseconds needed to make the steps sound continuous. The minus sign indicates that the sound generator will be updated as soon as the new value for A is determined; the duration specified need only be long enough to cover the time between updates.

Next, let us try a sound in which the frequency varies with time. A siren is an example which can be characterized by a slowly rising and falling frequency. Apparently, this is a sufficient clue to the brain for us to recognize it as a siren. Try varying the frequency range and step in the following program, and see how far they can be varied and still have it recognizable as a siren.



```
170 REM SIREN
180 N=1
190 FOR F=700 TO 900 STEP 5
200 CALL SOUND(–99,F,0)
210 NEXT F
220 FOR F=900 TO 700 STEP –8
230 CALL SOUND(–99,F,0)
240 NEXT F
250 N=N+1
260 IF N=4 THEN 270 ELSE 190
270 REM END
```

N=4 on line 260 limits the siren to 3 up-down frequency sweeps.

In the next example, let us vary both the frequency and the volume as a function of time. Imagine a large "killer" bee buzzing around you, with the frequency of the buzz proportional to the rate of the beating wings, and the volume proportional to the closeness of the bee.

Continued on p. 16

Birth of a legend.

Epson.

A whole new generation of Epson MX printers has just arrived. And while they share the family traits that made Epson famous — like unequalled reliability and ultra-fine printing — they've got a lot more of what it takes to be a legend.

For instance, they've got a few extra type styles. Sixty-six, to be exact, including italics, a handy subscript and superscript for scientific notation, and enough international symbols to print most Western languages.

What's more, on the new-generation MX-80, MX-80 F/T and MX-100, you get GRAFTRAX-Plus dot addressable graphics. Standard. So now you can have precision to rival plotters in a reliable Epson printer. Not to mention true backspace, software printer reset, and programmable form length, horizontal tab and right margin.

All in all, they've got the features that make them destined for stardom. But the best part is that beneath this software bonanza beats the

Uh...three legends.

heart of an Epson. So you still get a bidirectional, logical seeking, disposable print head, crisp, clean, correspondence quality printing, and the kind of reliability that has made Epson the best-selling printers in the world.

All of which should come as no surprise, especially when you look at the family tree. After all, Epson *invented* digital printers almost seventeen years ago for the 1964 Tokyo Olympics. We were the first to make printers as reliable as the family stereo. And we introduced the computer world to correspondence quality printing and disposable print heads. And now we've given birth to the finest printers for small computers on the market.

What's next? Wait and see. We're already expecting.

EPSON
EPSON AMERICA, INC.
3415 Kashiwa Street • Torrance, CA 90505 • (213) 539-9140

FEATURE	ORIGINAL MX-80	GRAFTRAX-80*	ORIGINAL MX-100	MX-80	MX-80 F/T	MX-100
				with GRAFTRAX-Plus		
Bidirectional printing	X	X	X	X	X	X
Logical seeking function	X	X	X	X	X	X
Disposable print head	X	X	X	X	X	X
Speed: 80 CPS	X	X	X	X	X	X
Matrix: 9 x 9	X	X	X	X	X	X
Selectable paper feed			X		X	X
PAPER HANDLING FUNCTIONS						
Line spacing to n/216		X		X	X	X
Programmable form length	X	X	X	X	X	X
Programmable horizontal tabs	X	X	X	X	X	X
Skip over perforation			X	X	X	X
PRINT MODES AND CHARACTER FONTS						
96 ASCII characters	X	X	X	X	X	X
Italics character font		X		X	X	X
Special international symbols				X	X	X
Normal, Emphasized, Double-Strike and Double/Emphasized print modes	X	X	X	X	X	X
Subscript/Superscript print mode				X	X	X
Underline mode				X	X	X
10 CPI	X	X	X	X	X	X
5 CPI	X	X	X	X	X	X
17.16 CPI	X	X	X	X	X	X
8.58 CPI	X	X	X	X	X	X
DOT GRAPHICS MODE						
Line drawing graphics				X	X	X
Bit image 60 D.P.I.		X	X	X	X	X
Bit image 120 D.P.I.		X	X	X	X	X
CONTROL FUNCTIONS						
Software printer reset		X		X	X	X
Adjustable right margin			X	X	X	X
True back space		X		X	X	X
INTERFACES						
Standard — Centronics-style 8-bit parallel	X	X	X	X	X	X
Optional — RS-232C current loop w/2K buffer	X	X	X	X	X	X
RS-232C x-on/x-off w/2K buffer	X	X	X	X	X	X
IEEE-488	X	X	X	X	X	X

*Tandy TRS-80 block graphics only available with GRAFTRAX 80.

ABCDEFGHIJKLMN abcdefghijklmn *ABCDEFGHIJKLMN abcdefghijklmn* Ø1234

ABCDEFGHIJKLMN abcdefghijklmn *ABCDEFGHIJKLMN abcdefghijklmn* Ø1234

ABCDEFGHIJKLMN abcdefghijklmn *ABCDEFGHIJKLMN abcdefghijklmn* Ø1234

ABCDEF abcdef *ABCDEF abcdef* Ø123456

ABCDEFGHIJKLMNOPQRSTUVWX abcdefghijklmnopqrstuvwx *ABCDEFGHIJKLMNOPQRSTUVWX abcdefghijklmnopqrstuvwx* Ø1234567

ABCDEF abcdef *ABCDEF abcdef* Ø123456

ABCDEFGHIJKLMN abcdefghijklmn *ABCDEFGHIJKLMN abcdefghijklmn* Ø1234

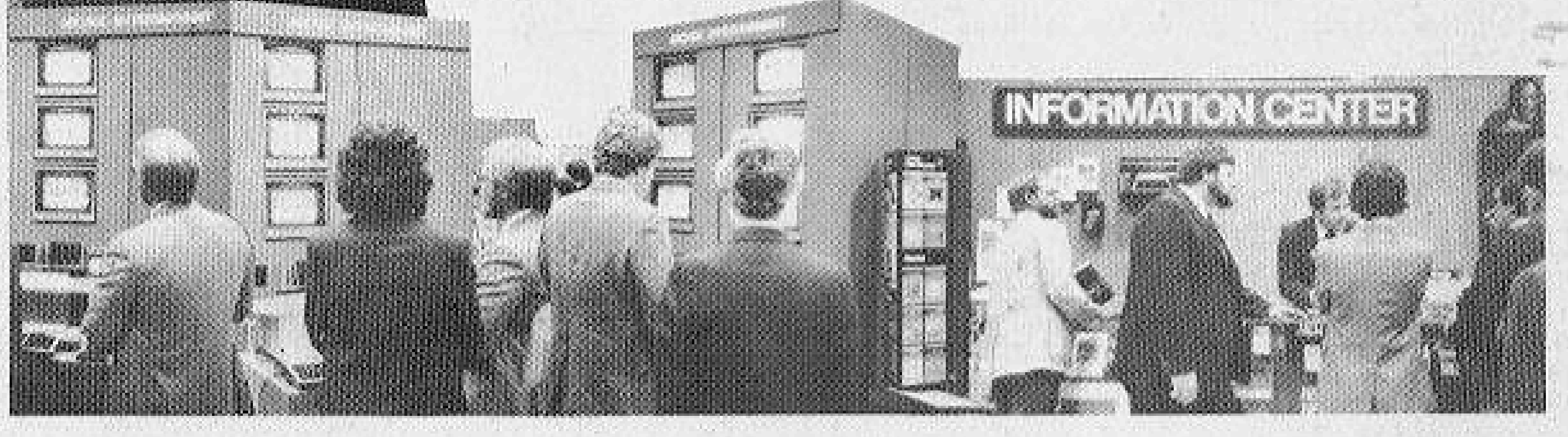
ABCDEFGHIJKLMNOPQRSTUVWX abcdefghijklmnopqrstuvwx *ABCDEFGHIJKLMNOPQRSTUVWX abcdefghijklmnopqrstuvwx* Ø1234567

On Exhibit:

A Look at the Summer CES & NCC

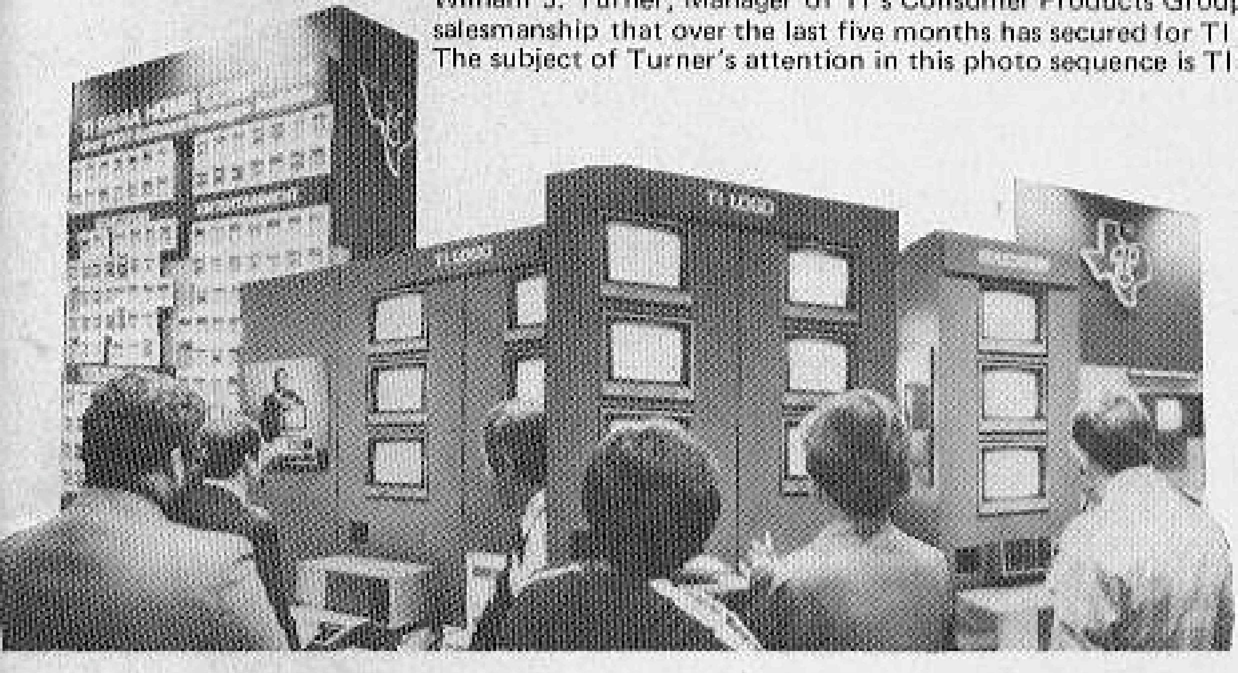
As Photographed By Gary M. Kaplan

Nowhere was the video games software explosion more evident than at the Summer Consumer Electronics Show (CES) in Chicago. Texas Instruments was there in force, demonstrating that its Home Computer—with a larger, more varied solid-state software offering than all its computer competitors combined—could also hold its own in the entertainment area against the popular games-only machines. Here, Jim Dramus, a master TI programmer, is shown demonstrating his latest creation, *Parsec*—a fast action, high-resolution space game with synthesized speech (a female Oriental voice with an Oxford-English accent) that goes beyond where the popular arcade games, *Defender* and *Scramble*, leave off. Other forthcoming game releases previewed at the show included *Tunnels of Doom* (a fantasy adventure game with graphics), and *Chisholm Trail* (an innovative brand of Western maze excitement). *Munch Man*, *TI Invaders*, and *Tombstone City: 21st Century* were other arcade-style games that drew huge crowds of interested dealers, distributors, and chain-store buyers. The tremendous interest in TI's entertainment offerings was reflected in the record amount of product literature and price sheets distributed at the information center.



The TI-99/4A also enjoyed exposure at the National Computer Conference (NCC) in Houston. Although confined to part of the top level of the two-story TI display area, the crowds of showgoers found their way up there to see demonstrations of games, educational software, and a prototype IEEE-488 interface card (installed in the Peripheral Expansion System) that was shown controlling a stack of sophisticated electronic test equipment. Edutronics/McGraw-Hill was also at CES demonstrating its 99/4A-based interactive video CAI delivery system with present offerings including *Advanced Project Management*, *VSAM*, *Structured COBOL*, and *Pascal*.

William J. Turner, Manager of TI's Consumer Products Group, is seen here at the CES demonstrating the type of aggressive salesmanship that over the last five months has secured for TI the largest U.S. retail distribution of any consumer computer. The subject of Turner's attention in this photo sequence is TI's new, attractive point-of-purchase display.



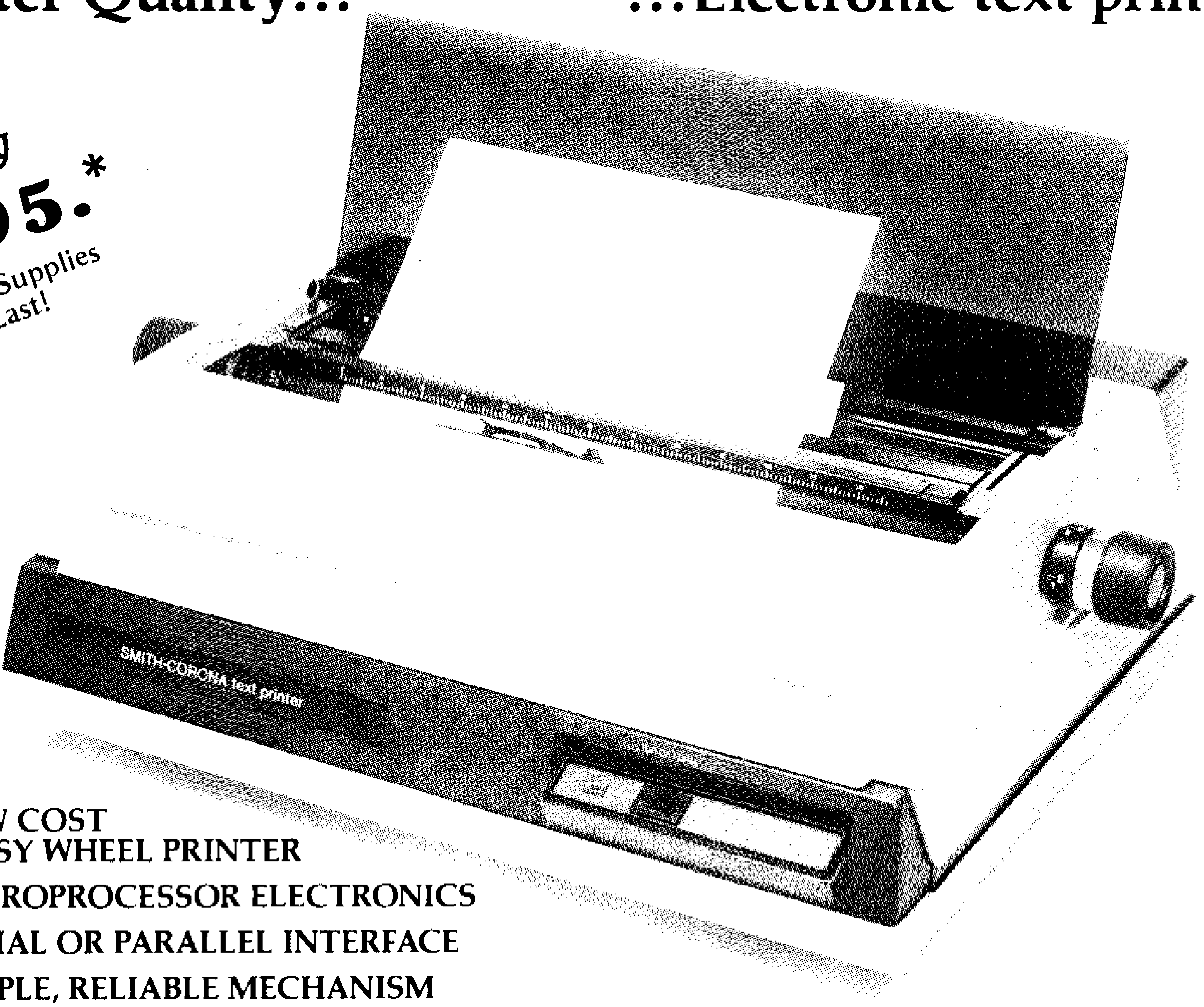
TI's strong focus on educational software at CES included the introduction of TI LOGO II: the new Scott Foresman *Division 1* and *Reading* Command Modules; plus the Addison-Wesley *Computer Math Games*, *Milliken Math Series*, and *Scholastic Spelling Series*; as well as the disk-based Minnesota Educational Computing Consortium (MECC) economics, math and science programs. By demonstrating that a *single* Home Computer could hold its own with any video game machine, while at the same time being clearly superior to any game machine or consumer computer in its educational capabilities, TI delivered a classic 1-2 punch that was squarely aimed at knocking out any contenders for the consumer electronics entertainment dollar.

The Home Computer wasn't the only TI product line receiving the close scrutiny of the electronics-sophisticated crowd attending CES. The long-awaited keystroke-programmable computer, TI-88, was finally introduced. Positioned as the first TI product that is part of a new thrust in portable computing, the user-prompting TI-88 system already includes a 3-lines-per-second thermal printer, audio cassette interface, and the ability to use snap-in Constant Memory™ Modules (CMOS RAM) for program and data storage. Although TI's other stunning CES introduction, the Magic Wand Speaking Reader, at first seems worlds apart from the TI-88 and 99/4A system, the "computer connection" in adaptable technology should soon be forthcoming. In much the same way that TI's advanced speech synthesis technology found its way into a low-cost, revolutionary Speak & Spell learning aid—founding a new industry in the process— the extremely rugged, low-cost, and tiny infrared optical scanning technology found in Magic Wand has the ability to decipher computer-readable information (into human speech, data, or programs) and replace the time-consuming process of committing material to silicon chips.

SMITH-CORONA® TP-I

Letter Quality... ...Electronic text printer

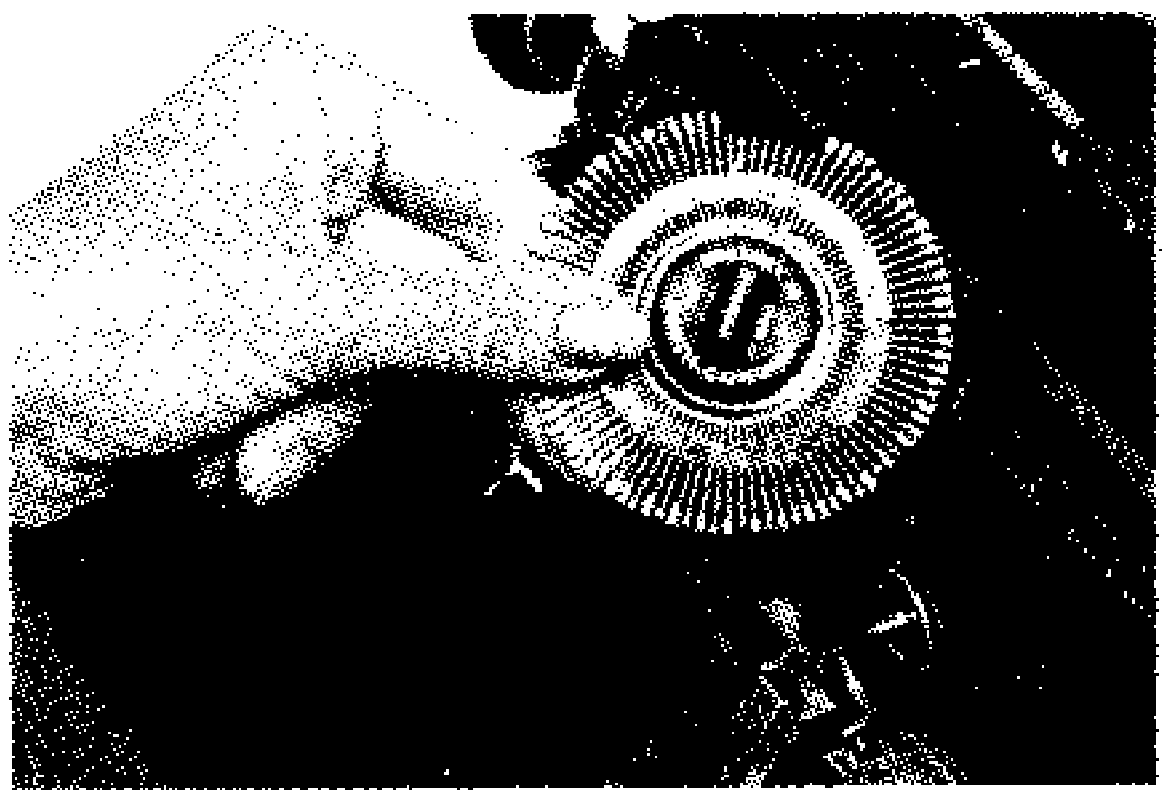
Only $695.* While Supplies Last!

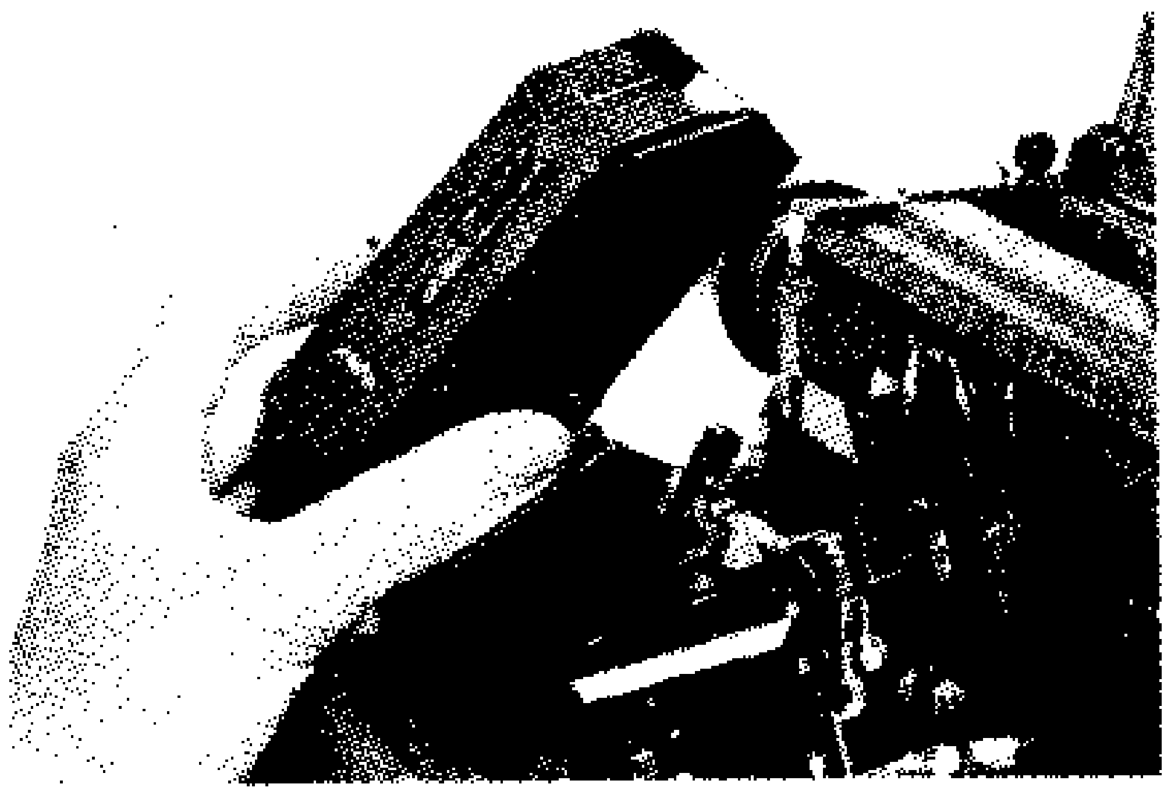


- LOW COST DAISY WHEEL PRINTER
- MICROPROCESSOR ELECTRONICS
- SERIAL OR PARALLEL INTERFACE
- SIMPLE, RELIABLE MECHANISM

The Smith-Corona TP-I text printer is a microprocessor controlled daisy wheel printer which delivers fully formed executive quality printout at a speed of 144 words per minute. The printer is a simple, low cost, and reliable unit which can be utilized with word processing systems, microcomputers, personal computers, small business systems, or in any environment which requires high quality printing. Its compact size and attractive packaging will allow it to blend into any environment.

The TP-I text printer is easy to use and incorporates a limited number of operator controls. Simply turn power on, load paper as you would with any standard office typewriter, and the printer is ready to go. Drop-in ribbon cassettes and quick change snap-on daisy print wheels are standard features which help make the printer simple to use.





The printer is available with either a parallel or a serial data interface. It prints an 88 character ASCII set in either a 10 character per inch or 12 character per inch version. The 10 CPI model prints a 105 character line while the 12 CPI version expands the line length to 126 characters. Various fonts are available for each pitch and the printer will handle single sheets or forms.

The TP-I text printer is a product of Smith-Corona's advanced technology, capacity for design simplification, parts count reduction, modular construction, mass production techniques, and vast experience in the printing products field. It offers the benefits of a high quality daisy wheel printer at an extremely low cost.

Now, letters, documents, forms, reports, price lists, data sheets, — all can be printed with the quality that your company's image deserves, and at a readily affordable price.

* FOB Springfield, MO.

COMPUTECH DISTRIBUTING

209 E. WALNUT SPRINGFIELD, MO 65806
417/869-1684

Letters . . . from p. 7

sorting to be arranging in order, in this case smallest to largest. And it sounds like you are printing each process or "pass" of the sort routine and looking how the sort is at that stage. The program sorts completely before printing anything. I wonder if you inadvertently left out line 300.

300 IF SW=1 THEN 210

This means if at least one switch has been made, then reset the limit and pass through the numbers again.

I just typed in the program again as published, and it is correct, it does indeed sort.

By the way, you indicate sorting of nine digits is accomplished in approximately five reRUNs. The number of "passes" or times through the FOR-NEXT loop depends on what order the numbers are in originally. If they are already in order, one pass will be necessary. If they are in exactly opposite order, eight passes are needed.I guess the average for a random order could be five.

Regena has developed another sort routine that is more efficient for large amounts of mixed-up numbers. Watch for it in a future issue.

Dear Sir:

Hallelujah! Finally a magazine for TI people. This is the best thing since soap! I just received all the issues from start to No. 4 and I love them all. I have not been able to put them down and find myself constantly picking them back up again and again . . . I have referred your magazine to a local dealer and he feels that he *should* be carrying your magazine so that's Step one.

Now for some suggestions:

1. On the programs you list in the magazines from now on, I think it would be helpful to include some word about if the program may use "CS1" for output as well as "DSK1."—so those of us poor people (who don't have disks) may use our cassettes.
2. Are BASF 5¼ inch floppies compatible with the TI controller? In your article on floppies, this drive wasn't mentioned.
3. In the book "Programming BASIC with the TI Home Computer" there are many programs which do not run properly. Would you please review the book for an article and fix the errors; you'll find them easily—they are the biggest problems and programs at the end of each chapter.
4. I would like to see more graphics programs that are pretty and exciting to watch.
5. What about real-time clock for the TI? Well for now, that's all my questions. Again your magazine is "boss" and Byte has nothing on you!

Greg O'Hara
Fairbon, OH

As to the cassette option, Greg, we have offered it consistantly whenever possible. See, for example, Electronic Home Secretary, Interactive Forms Generator, Music Text Editor, *and* Spriter. *By their very nature, some utility programs are impractical with cassette file operation (e.g., GRAPHTRAX), so we can only publish them with disk I/O.*

The BASF disks can probably be made to work if the ones you get have a track-to-track access time of less than 20 ms. As you may have noticed, the 99'er Bookstore stopped carrying the McGraw-Hill book you mentioned (by H. D. Peckham) because of the many errors uncovered. Instead, we now only carry Introduction to TI BASIC. *Watch an upcoming issue for corrections to the Peckham book.*

Now that the Mini Memory is available, watch for those exciting graphics exhibitions a-la "TI LINES" demo. As for the real-time clock, I heartily agree that it's about time (excuse the pun!) we had one. Perhaps TI will let us publish the assembler code for the simulated analog clock residing in Mini RAM that was demonstrated recently at the CES. . .

Dear Sir:

As only a semi-proficient programmer, I have really enjoyed your magazine. By entering some of the games and other programs you offer, I have learned much about programming convention and am now able to create my own efficient programs.

One thing I would like to see is much more on the LOGO language. I am working with my 3 and 5 year-olds on LOGO and could use some hints as how to direct them without pushing—especially with the 3 year-old, who I am having to teach how to read and type.

Walter A. Elsaesser Jr.
Spring, TX

I'm glad you're getting so much benefit from the magazine, Walter. Yes, you can expect to see much more of this kind of LOGO material in our magazine-within-a magazine, LOGO Times. *You might like to try out this issues's "Tower of Hanoi" puzzle/game with the children, and let us know how they respond.*

Call Sounds . . .
from p. 9

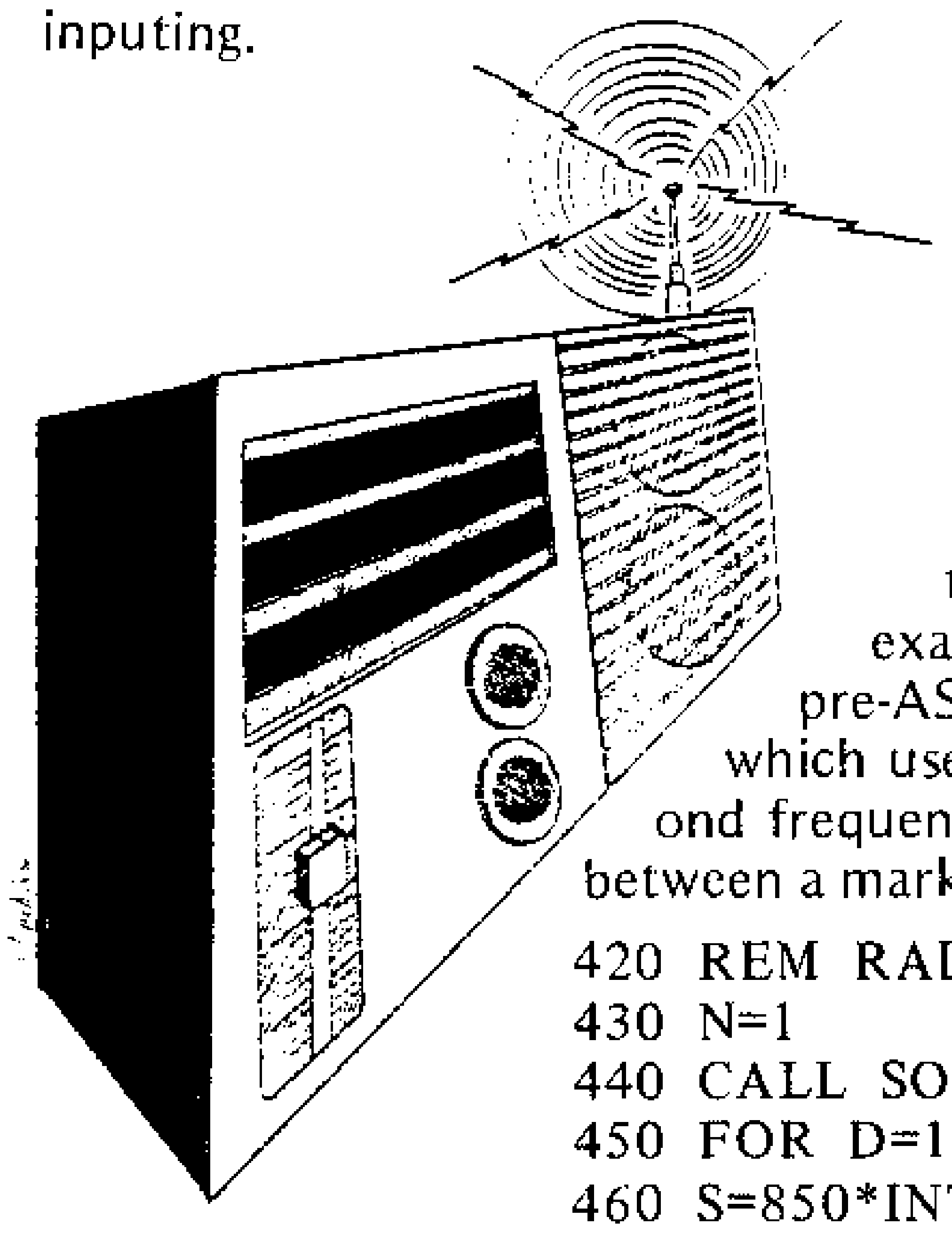
```
280 REM BEE
290 N=1
300 CALL SOUND(-99,RNDx8+110,RNDx10)
310 N=N+1
320 IF N=75 THEN 330 ELSE 300
330 REM END
```

Unlike the previous examples, where the variations in frequency and volume were obtained by using a FOR–NEXT loop, the variations in this case were obtained by using the RND statement. It is interesting to note that this routine will not sound the same in TI Extended BASIC—with the bee sounding very sluggish. This is one case in which TI BASIC runs faster than the Extended version.

For the next sound, imagine that you are tuning a shortwave radio receiver. The background static is simulated with noise type –8 and the random signal is simulated with frequency #3. The random volume on frequency #3 simulates varying signal levels with the noise volume formulated to be high when the signal level is low and vice versa.

```
340 REM SHORTWAVE RECEIVER
350 N=1
360 F=RND*15000+110
370 A=RND*30
380 CALL SOUND(-99,111,30,111,30,F,A,-8,30-A)
390 N=N+1
400 IF N=100 THEN 410 ELSE 360
410 REM END
```

Frequencies #1 and #2 are "do nothing frequencies," since their volumes are set to the minimum, and are inserted so the program will recognize frequency #3 from which noise type –8 is derived. The 111's therefore were picked for ease of inputing.



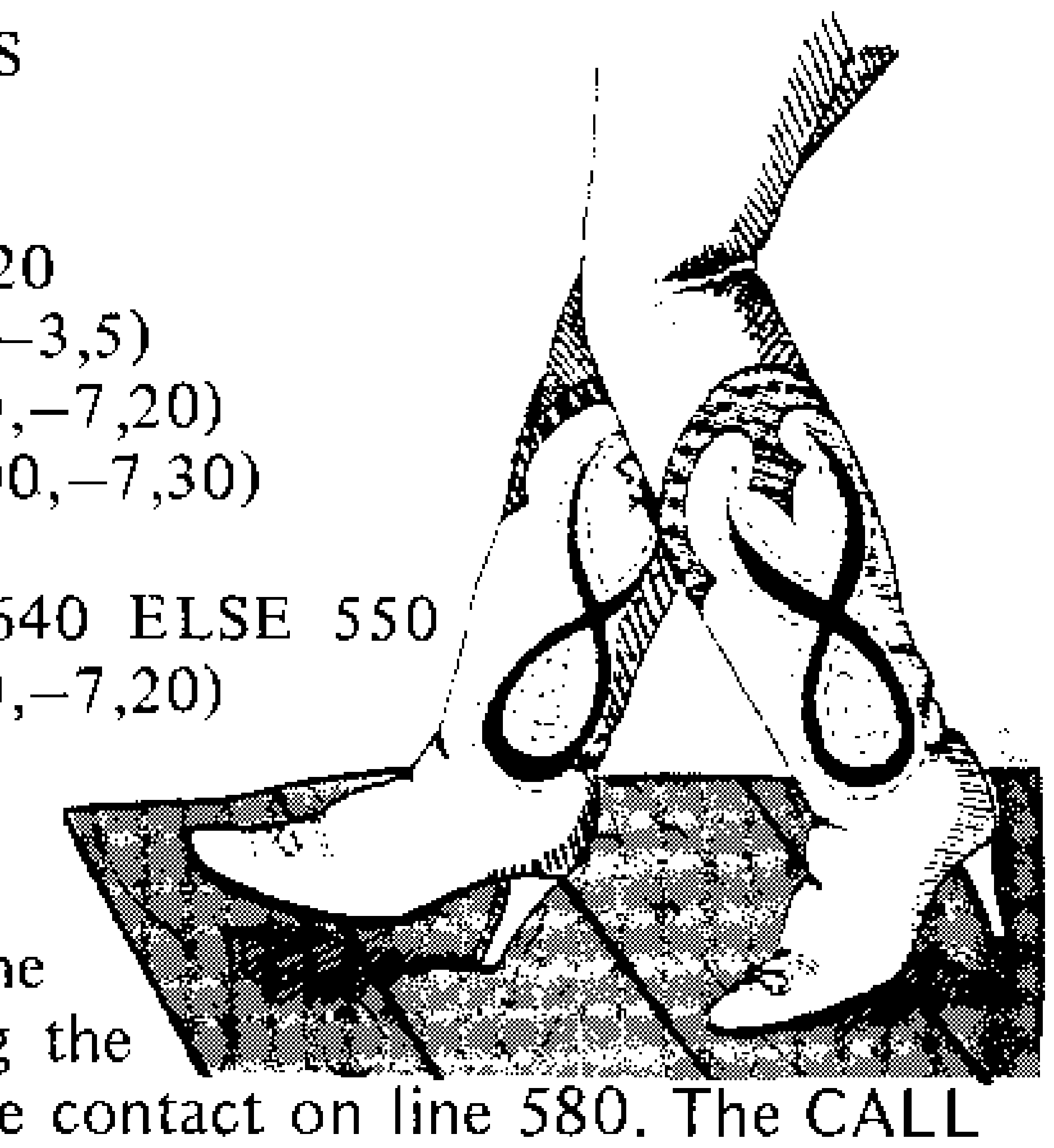
Next, imagine that the radio of the previous example is now tuned to a pre-ASCII teleprinter signal which uses an 850 cycle-per-second frequency shift to differentiate between a mark and space.

```
420 REM RADIO TELEPRINTER
430 N=1
440 CALL SOUND(22,2975,0)
450 FOR D=1 TO 5
460 S=850*INT(RND*2)
470 CALL SOUND(22,2125+S,0)
480 NEXT D
490 CALL SOUND(31,2125,0)
500 N=N+1
510 IF N=30 THEN 520 ELSE 440
520 REM END
```

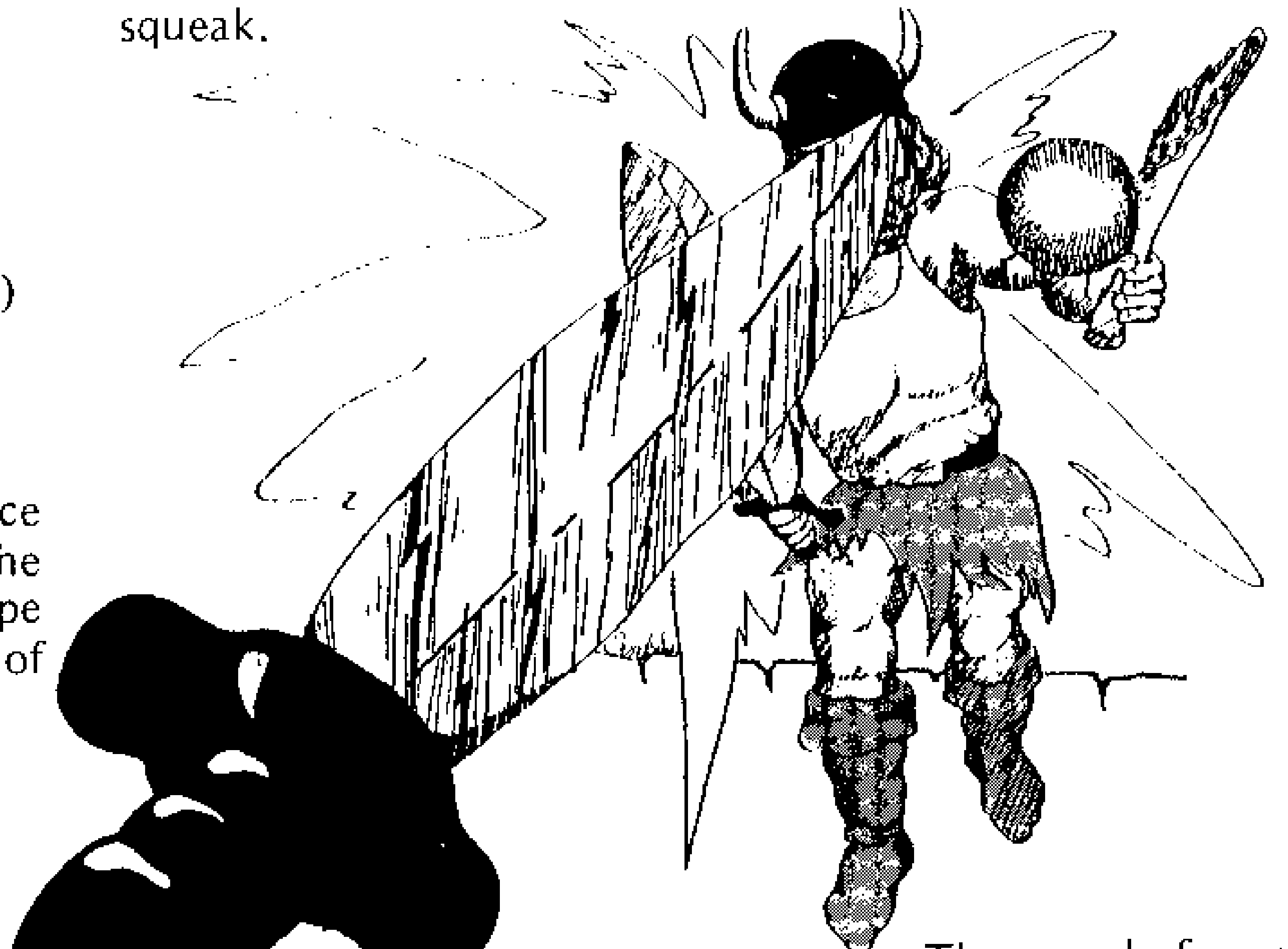
One character consists of a 22 millisecond (ms) start pulse, followed by a five bit code for the character, with each bit 22 ms long, and a 31 ms stop pulse. Line 440 generates the start pulse, which is always a space. The FOR--NEXT loop in lines 450-480 randomly generates a mark or space pulse for the five data bits, and line 490 generates the stop pulse, which is always a mark. Line 510 limits the number of characters generated to 29. Like the "bee" sound, this will not come out well in Extended BASIC. In general, data communications signals are easy to imitate because they are well defined by standards.

For a change of pace, try the following sound:

```
530 REM FOOTSTEPS
540 N=1
550 X=INT(RND*5)
560 IF X=2 THEN 620
570 CALL SOUND(5,-3,5)
580 CALL SOUND(30,-7,20)
590 CALL SOUND(500,-7,30)
600 N=N+1
610 IF N=30 THEN 640 ELSE 550
620 CALL SOUND(60,-7,20)
630 GOTO 590
640 REM END
```



The CALL SOUND on line 570 is the heel contacting the floor, followed by the sole contact on line 580. The CALL SOUND on line 590 is the delay between steps. Lines 550, 560, and 620 add a shuffle about once in every 4 steps to make the footsteps sound a little more natural. Changing the noise type on line 580 from –7 to –5 will make the shoes squeak.



The sound of a sword fight can be recreated by recognizing that the sword blade is a resonator like the door chimes, except that instead of being essentially free, it is clamped at the handle—thus creating overtones at different ratios than for the chime bars. Also, the amplitude decays faster, since the collision of the two blades would have a dampening effect.

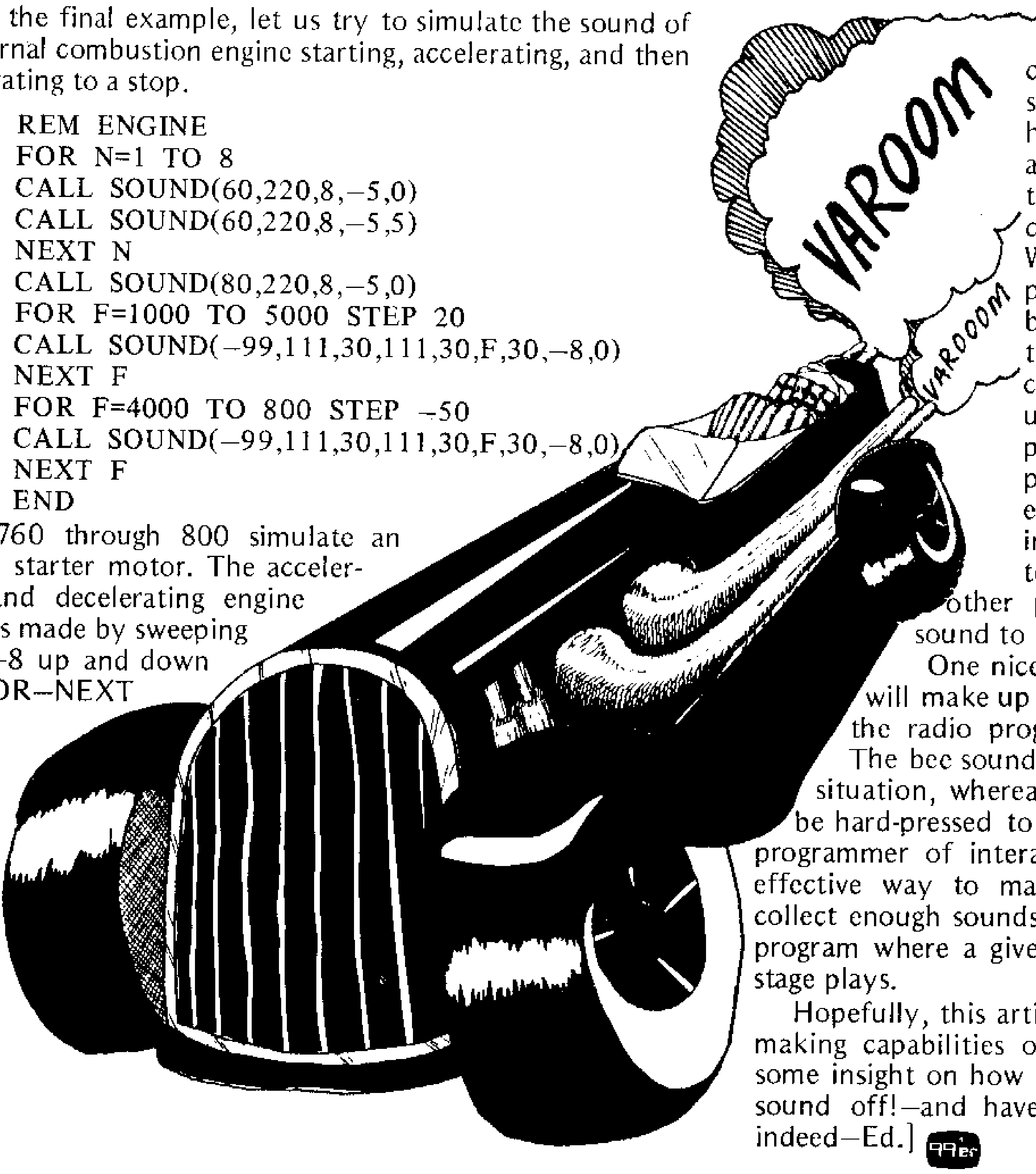
```
650 REM SWORD FIGHT
660 N=1
670 FOR A=0 TO 30 STEP 15
680 CALL SOUND(-99,1000,A,3250,A,6750,A)
690 NEXT A
700 FOR D=1 TO RNDx200
710 NEXT D
720 N=N+1
730 IF N=30 THEN 740 ELSE 670
740 REM END
```

Lines 700 and 710 add a random delay between sword clashes.

For the final example, let us try to simulate the sound of an internal combustion engine starting, accelerating, and then decelerating to a stop.

```
750 REM ENGINE
760 FOR N=1 TO 8
770 CALL SOUND(60,220,8,-5,0)
780 CALL SOUND(60,220,8,-5,5)
790 NEXT N
800 CALL SOUND(80,220,8,-5,0)
810 FOR F=1000 TO 5000 STEP 20
820 CALL SOUND(-99,111,30,111,30,F,30,-8,0)
830 NEXT F
840 FOR F=4000 TO 800 STEP -50
850 CALL SOUND(-99,111,30,111,30,F,30,-8,0)
860 NEXT F
870 END
```

Lines 760 through 800 simulate an electric starter motor. The accelerating and decelerating engine sound is made by sweeping noise −8 up and down in FOR–NEXT loops.

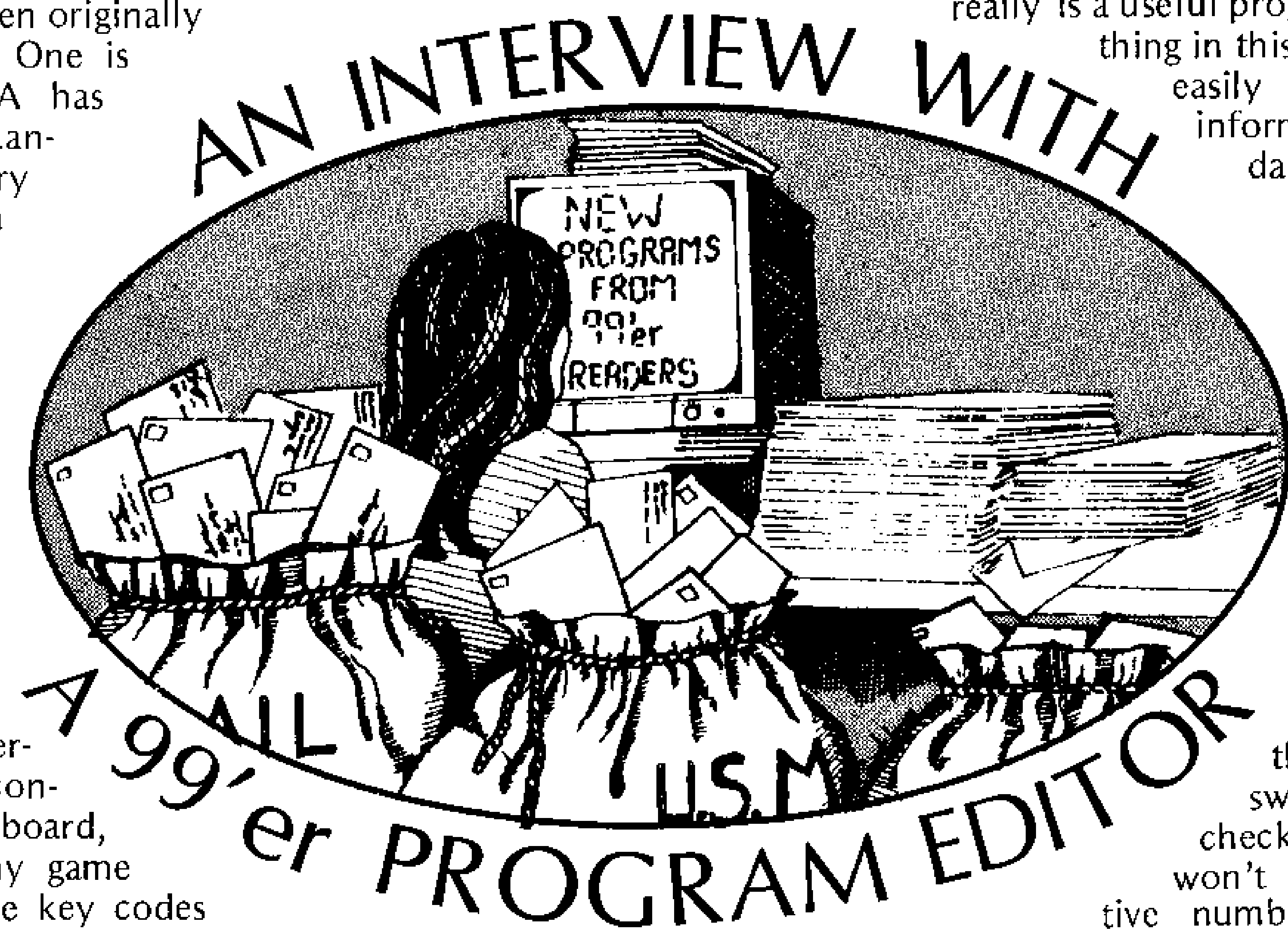


Now that you're convinced that your computer can produce a wide variety of sounds, you are probably wondering how one uses these sounds. If you are an adventure game programmer, suppose that the player is confronted with a door with a knocker and a bell button. Wouldn't it be more interesting if the player *heard* the bell upon pressing the bell button—before getting the usual textual message? Or if you are dynamically simulating a race car, you could use line 820 in the engine sound example in a CALL KEY loop where the F parameter would depend on the accelerator pedal setting. The duration in the CALL SOUND would have to be increased if you are updating other parameters in the loop—i.e., for the sound to be continuous.

One nice thing about sounds is that the listener will make up the visual image that fits, which is why the radio programs of years past were so effective. The bee sound, for instance, immediately conveys the situation, whereas a screenful of color graphics would be hard-pressed to evoke the same feeling. Thus, for the programmer of interactive fiction, sound should be a very effective way to make a story come alive. If you could collect enough sounds, you could even write a sound effects program where a given sound could be accessed on cue for stage plays.

Hopefully, this article has opened your ears to the sound-making capabilities of your TI-99/4A, and has given you some insight on how to create and use your own sounds. So sound off!—and have fun doing it. [That's sound advice, indeed—Ed.]

AN INTERVIEW WITH A 99'er PROGRAM EDITOR



GARY KAPLAN: *What does the Program Editor do?*

CHERYL WHITELAW: I review computer programs for the *99'er Magazine* to make sure they are in "publishable" form. As an editor, I add the *99'er* standard heading, check the spelling, check for user-friendliness, and make sure the program can be run on both the TI-99/4 and TI-99/4A.

GK: *What do you mean by "user-friendly?"*

CW: We want all of our *99'er* readers, not just the author of the program, to be able to use the programs we publish. I check the instructions to see if they are clear, both easy to understand and easy to read. Some color combinations are harder to read on the 99/4A than on the 99/4. Also, printing on the 99/4A should be double-spaced if there is enough room on the screen.

Any information the user needs to INPUT can be a source of error, so the program needs to be specific in its prompts. If data needs to be entered, perhaps an example should be given. If the user needs to press only one key from a multiple choice, a CALL KEY method is better than INPUT. The program should not crash on user input, so limits need to be checked.

GK: *How do you make sure the program can run on both the 99/4 and 99/4A?*

CW: There are two main things I check for if a program has been originally written for the 99/4. One is memory; the TI-99/4A has 256 fewer bytes of Random Access Memory (RAM). I go through the program several times to make sure that the user won't get a MEMORY FULL error. Sometimes I can go through a program completely with no problem, thenchoose an option to return to one section and get MEMORYFULL.

The other main difference between the two consoles is the split keyboard, which is used in many game programs. Some of the key codes returned for keys pressed in the CALL KEY statements are different. One example is that the ENTER key cannot be used with the split keyboard approach. Several of the keys toward the middle of the keyboard have different key codes than the 99/4 console.

GK: *Are there problems with programs written for the 99/4A that are run on the 99/4?*

CW: Yes—the 99/4 does not have the lower-case letters defined; this involves characters numbered from 96 through 122. Another problem I watch for is the use of the Q key. Q should *not* be used as the key to press for firing in a game, nor should it be used as a variable name in a program because it is too easy to accidentally press SHIFT Q and lose the program on the TI-99/4.

GK: *What do you look for or test in game programs?*

CW: A game program probably has the potential of the greatest number of users, so it has to be user friendly. (There's that phrase again—see how important it is!) The instructions need to be understandable. If the game is a standard parlor game, it should follow the standard rules. You wouldn't want to be playing a checkers game, for example, and suddenly have the computer make an illegal move. If the game involves randomness, I'll play several times to make sure a pattern doesn't develop. Sometimes a RANDOMIZE statement at the beginning of the program does not provide sufficient randomness, and additional RANDOMIZE statements should be used just before each statement containing RND.

If the game involves skill and coordination, then speed is also a factor. In Extended BASIC I spend a lot of time testing the CALL COINC statement (sometimes testing different tolerances) to make sure hits are recorded realistically. If the program detects the fire button, but ten statements later checks to see if the object was hit, the delay could be too frustrating for a person playing the game.

If the program requires using the arrow keys, I test to make sure the object moves correctly and also to see what happens if an incorrect key is accidentally pressed (is there a default value, is the key ignored, or does the program give errors?). I also check the graphics if a moving object approaches the edge of the screen—the object may wrap or it may stay at the edge. I have to make sure you don't get a BAD VALUE IN . . . . error. If the game displays a score, I also make sure it is calculated correctly.

GK: *What about utility or data handling and file processing programs?*

CW: I read the author's manuscript and make sure the program does what the author says it is supposed to do and really is a useful program. Probably the main thing in this type of program is how easily the user can enter his information without making data entry errors.

GK: *What do you check in technical programs?*

CW: I get out my calculator and make sure the formulas are correct and consistent with the author's manuscript. Quite often the technical program will ask the user to INPUT numbers and then will return an answer. The program should check the limits so there won't be problems with negative numbers, large numbers, or division by zero, for example. A technical program differs from a tutorial or game program because you can assume the person using the program is knowledgeable in the subject.

GK: *You mentioned tutorials. Is there something in particular you check in educational programs?*

CW: The first thing I do is pretend I do not know about the subject, then go through the program to see if it teaches in a logical, step-by-step process. An important consideration in educational programs is what happens if an incorrect answer is entered. Another thing to consider is that educational programs should be very friendly and non-intimidating. The programs should utilize the capability of this computer to combine text with graphics and sound to enhance the learning process.

GK: *How much can you change a program submitted for publication?*

CW: *99'er Magazine* reserves the right to edit any program or article that is submitted for publication, and the terms are explained in the contract sent to the author; however, I try to change as little as possible. There are usually

Continued on p. 20

TEX-COMP™

99/4-USERS SUPPLY DIVISION OF CALVERT ENGINEERING, INC.

TI-99/4 PRODUCTS AT PROFESSIONAL PRICES

ATTENTION TI-99/4 USERS   TEX COMP HAS BEEN SUPPORTING THE ENTIRE 99/4 PRODUCT LINE TO PROFESSIONAL AND INDUSTRIAL ACCOUNTS, SINCE THE INTRODUCTION OF THE TI-99/4. NOW YOU CAN PURCHASE ALL YOUR TI-99/4 REQUIREMENTS FROM ONE SOURCE AT LOW "PROFESSIONALLY DISCOUNTED" PRICES ON THE **ENTIRE** 99/4 PRODUCT LINE. WE ARE A DIRECT TI NATIONAL ACCOUNT AND DEAL DIRECTLY WITH TI, ASSURING YOU THE BEST SERVICE, EARLIEST SHIPMENT AND LOWEST POSSIBLE PRICES.

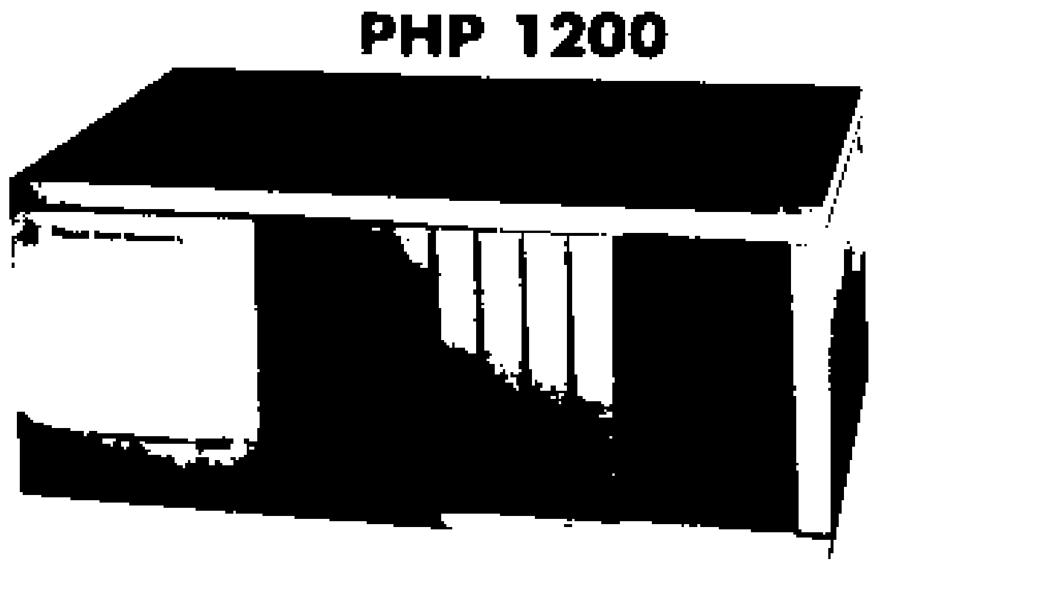
SEND FOR FREE ORDER KIT AND COMPLETE PRICE LIST
CONTAINS MONEY SAVING SPECIAL OFFERS AND NEW PRODUCT INFORMATION

TI-99/4 PRODUCT LINE

			Your Cost
CONSOLE			
PHC	004A	TI-99 4A Home Computer	279.95
PERIPHERALS			
PHP	1200	Peripheral Expansion Box**	178.95
PHP	1220	RS-232 Card**	125.95
PHP	1240	Disk Controller Card**	178.95
PHP	1250	Expansion Box Disk Drive Card**	286.95
PHP	1260	Memory Expansion Card** (RAM)	214.95
PHP	1270	P-Code Card**	178.95
PHP	1500	Solid State Speech™ Synthesizer	107.95
PHP	1600	Telephone Coupler (Modem)	160.95
PHP	1700	RS-232 Accessories Interface	160.95
PHP	1800	Disk Drive Controller (One Disk Manager module packed with each Disk Controller)	214.95
PHP	1850	Disk Memory Drive	357.95
PHP	1900	Solid State Printer	286.95
PHP	2200	Memory Expansion (32K RAM)	286.95
PHP	2300	VCR Controller**	499.95
PHP	2400	P-Code Peripheral*	286.95
PHA	2100	R F Modulator (TV Adapter)	35 95
PHA	4100	10 Color Monitor	299 95
OPTIONAL ACCESSORIES			
PHP	1100	Wired Remote Controllers (Pair)	25 95
PHA	1950	Thermal Paper (2 Pack)	7 50
PHA	2000	Dual Cassette Cable	10 95
PHA	2010	Monitor Cable	14.95
PHA	2020	Audio Adapter (Headphone Jack)	14.95
PHA	2310	Panasonic VCR Controller Cable	82.95
PHA	2320	Sony VCR Controller Cable	82.95
PHA	2330	Pioneer VCR Controller Cable	82.95
PHA	2605	Blank Overlays (4 Pack)	5.95

SPECIAL Peripheral EXPANSION BOX

PHP 1200



Reg. $178.95

NOW $169.95

With any two cards at our regular low price.

TEX-COMP PRICE SPECIAL

$279.95

Console only
PHC 004A

Reg. $49.95 modulator (PHA2100) only $14.95 when Purchased with console.

Education/Personal Enrichment Continued

			Your Cost
		Diskette	
PHD	5011	Computer Music Box	15.95
PHD	5018	Market Simulation	15.95
PHD	5019	Teach Yourself Extended BASIC (Extended BASIC Command Module is required)	19 95
PHD	5020	Music Maker Demonstration (Music Maker Command Module is required)	11.95
PHD	5023	Basketball Statistics (Extended BASIC Command Module is required)	19.95
PHD	5026	Bridge Bidding I	19.95
PHD	5030	Speak & Spell™ Program (Solid State Speech™ Synthesizer is required)	22 95
PHD	5031	Speak & Math™ Program (Solid State Speech™ Synthesizer and Terminal Emulator II are required)	22.95
PHD	5039	Bridge Bidding II	22.95
PHD	5041	Bridge Bidding III	22.95
PHD	5042	Spell Writer (Terminal Emulator II Command Module and Solid State Speech™ Synthesizer are required)	22.95
PHD	5067	Teach Yourself Beginning BASIC**	22.95
		Cassette	
PHT	6007	Teach Yourself BASIC	22.95
PHT	6009	Music Skills Trainer	19 95
PHT	6011	Computer Music Box	11 95
PHT	6018	Market Simulation	11.95
PHT	6019	Teach Yourself Extended BASIC (Extended BASIC Command Module is required)	15.95
PHT	6026	Bridge Bidding I	9.95
PHT	6031	Speak & Math™ Program (Solid State Speech™ Synthesizer and Terminal Emulator II are required)	19.95
PHT	6039	Bridge Bidding II	19 95
PHT	6041	Bridge Bidding III	19 95
PHT	6042	Spell Writer (Solid State Speech™ Synthesizer and Terminal Emulator II Command Module are required)	19.95
PHT	6067	Teach Yourself Beginning BASIC**	19 95

Education Personal Enrichment

		Command Module	
PHM	3002	Early Learning Fun	22.95
PHM	3003	Beginning Grammar	22.95
PHM	3004	Number Magic	15 95
PHM	3005	Video Graphs	15 95
PHM	3008	Video Chess	53 95
PHM	3010	Physical Fitness	22 95
PHM	3015	Early Reading † (Solid State Speech™ Synthesizer is required)	41 95
PHM	3020	Music Maker (Data storage system is recommended)	30 95
PHM	3021	Weight Control and Nutrition (Data storage system is recommended)	46 95
PHM	3027	Addition and Subtraction I† (Solid State Speech™ Synthesizer is recommended)	30 95
PHM	3028	Addition and Subtraction II† (Solid State Speech™ Synthesizer is recommended)	30 95
PHM	3029	Multiplication I† (Solid State Speech™ Synthesizer is recommended)	30 95
PHM	3040	TI LOGO (Memory Expansion is required)	69 95
PHM	3043	Reading Fun†*	41 95
PHM	3064	Typing Tutor**	30 95
		Diskette	
PHD	5007	Teach Yourself BASIC	25 95
PHD	5009	Music Skills Trainer	25 95

Entertainment

		Command Module	
PHM	3009	Football	22 95
PHM	3018	Video Games I	22.95
PHM	3023	Hunt the Wumpus	22 95
PHM	3024	Indoor Soccer	22.95
PHM	3025	Mind Challengers	19 95
PHM	3030	A-Maze-Ing	19.95
PHM	3031	The Attack†*	30.95
PHM	3032	Blasto††	19 95
PHM	3033	Blackjack and Poker††	19 95
PHM	3034	Hustle††	19.95
PHM	3036	ZeroZap††	15.95
PHM	3037	Hangman††	15 95
PHM	3038	Connect Four††	15.95
PHM	3039	Yahtzee††	19 95
PHM	3041D	Adventure (Pirate Adventure Diskette Game included)	37 95
PHM	3041T	Adventure (Pirate Adventure Cassette Game included)	37 95
PHM	3052	Tombstone City: 21st Century	30 95
PHM	3053	TI Invaders	30 95
PHM	3054	Car Wars	30.95
PHM	3057	Munch Man**	30.95
		Diskette	
PHD	5002	TI Trek (with speech)	10.95
PHD	5010	Mystery Melody	10 95
PHD	5015	Oldies But Goodies - Games I	15.95
PHD	5017	Oldies But Goodies — Games II	19 95
PHD	5025	Saturday Night Bingo (Solid State Speech™ Synthesizer is required)	22 95
PHD	5037	Draw Poker	19.95
		Adventure Series	
PHD	5043	Pirate Adventure	22.95
PHD	5046	Adventureland	22.95
PHD	5047	Mission Impossible	22.95
PHD	5048	Voodoo Castle	22 95
PHD	5049	The Count	22 95
PHD	5050	Strange Odyssey	22 95
PHD	5051	Mystery Fun House	22 95
PHD	5052	Pyramid of Doom	22.95
PHD	5053	Ghost Town	22.95
PHD	5054	Savage Island I & II	30.95
PHD	5056	Golden Voyage	22 95
		Cassette	
PHT	6010	Mystery Melody	7.95
PHT	6015	Oldies But Goodies - Games I	10 95
PHT	6017	Oldies But Goodies — Games II	15 95
PHT	6025	Saturday Night Bingo (Solid State Speech™ Synthesizer is required)	19 95
PHT	6037	Draw Poker	14 95
		Adventure Series	
PHT	6043	Pirate Adventure	22.95
PHT	6046	Adventureland	22.95
PHT	6047	Mission Impossible	22 95
PHT	6048	Voodoo Castle	22.95
PHT	6049	The Count	22 95
PHT	6050	Strange Odyssey	22 95
PHT	6051	Mystery Fun House	22 95
PHT	6052	Pyramid of Doom	22 95
PHT	6053	Ghost Town	22 95
PHT	6054	Savage Island I & II	30.95
PHT	6056	Golden Voyage	22 95

			Your Cost
OTHER APPLICATION PROGRAMS			
		Command Modules	
PHM	3000	Diagnostic	22.95
PHM	3001	Demonstration	53 95
PHM	3011	Speech Editor (Solid State Speech™ Synthesizer is required)	35.95
PHM	3014	Statistics (Data storage system is recommended)	35 95
PHM	3026	Extended BASIC	74 95
PHM	3035	Terminal Emulator II	38 95
PHM	3055	Editor Assembler*	74 95
PHM	3058	Mini Memory*	74.95
		Diskette	
PHD	5004	Programming Aids I	11 95
PHD	5005	Programming Aids II	19 95
PHD	5006	Math Routine Library	22.95
PHD	5008	Electrical Engineering Library	22 95
PHD	5012	Programming Aids III	15 95
PHD	5013	Graphing Package	15.95
PHD	5016	Structural Engineering Library	22 95
PHD	5044	SMU Circuit Analysis I**	112.95
PHD	5063	UCSD Pascal* Compiler**	93 95
PHD	5064	UCSD p-System* Assembler Linker**	74.95
PHD	5065	UCSD p-System* Editor Filer Utilities**	57 95
PHD	5066	TI PILOT**	61 95
		Cassette	
PHT	6004	Programming Aids I	7 95
PHT	6006	Math Routine Library	19 95
PHT	6008	Electrical Engineering Library	19.95
PHT	6013	Graphing Package	11.95
PHT	6016	Structural Engineering Library	19.95
PHT	6044	SMU Circuit Analysis I**	12 95

SPECIAL—10

PREMIUM GRADE HUB-REINFORCED DISKETTES & FREE PLASTIC LIBRARY CASE

$24.95
POSTPAID

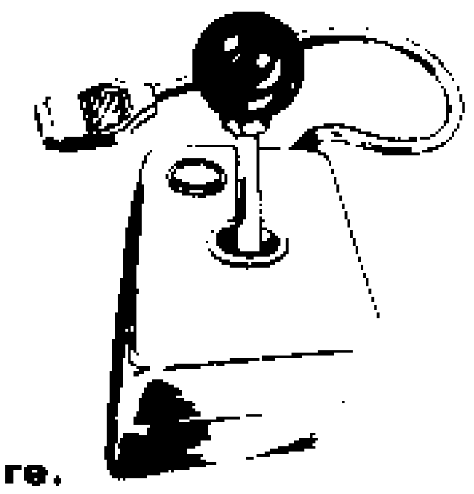
TI 99/4 COMPATIBLE

SUPER JOY STICK!

A commercial arcade joy stick adapted for use with your 99/4.

Available exclusively from TEX-COMP.

34.95 each
(specify 99/4 or Atari plug connection).



Increase your TI Invaders score. More reponsive action.

TERMS: All prices F.O.B. Los Angeles. For fastest service use cashiers check or money ordern. Personal checks take at least ten days to clear. Add 3% shipping and handling ($3.00 minimum). East of Mississippi 4½%. (Free shipping on all software orders over $100.00). All products are sold with the original manufacturer's guarantee (sent on request). Prices and availability subject to change without notice. We reserve the right to limit quantities. Sorry no credit card purchases or collect calls, these costs increase your prices.

TEX-COMP USERS SUPPLY DIVISION
P.O.BOX 33084 — — GRANADA HILLS, CA. 91344

(213) 873-4750
SOURCE TI-0166

TI99/4 Compatible

SPECIAL - 8

DATA QUALITY C-11 Cassettes (BASF) in FREE Plastic storage case

$9.95 Postpaid

Interview . . . from p. 18

several ways of programming to get the same result, and I try to preserve the author's style. Usually editing involves only a few lines. However, if I completely revamp a program or do major changes, you will see "Revised by C. Whitelaw" in the header remarks of the listing. Major changes are discussed with the author before the program is published.

GK: *What are examples of programs that require major changes?*

CW: One example is if there is a full-memory problem. Usually I have to sacrifice program documentation by deleting REMarks to save memory. I may have to change some of the logic to conserve memory. Another example is a program executing rather slowly because of inefficient or unnecessary loops, or because of using complicated logic when a simpler method will do. Sometimes I look at a program that seems to have a super idea but it simply won't run. I'll take a look at it anyway to try to find out how to make it run. Again, I try to preserve as much of the original program as possible, but the program isn't complete unless it can be run successfully.

GK: *If the 99'er readers wish to submit or sell programs, what are some guidelines?*

CW: Keep in mind that one of the *99'er Magazine's* major objectives is to help its readers get the most out of their computers–to become computer literate, to improve programming skills, and to be able to use their computers in a variety of ways. If the program has an interesting programming technique or unique graphics, it would be a good program to publish. If there is a use for the computer that would benefit the general public and save hours of tedious work for someone, it could be published. If the program can be easily adapted for a multitude of uses or if it contains a routine that could be incorporated into other programs, it would be worthwhile.

If you like to program games, remember there are many other people that like to program games, and the competition is stiffer. Therefore, your game must use unique graphics, be speedy and exciting, or captivate your user with a new idea. With the greater capabilities of Extended BASIC, more games will be written in Extended BASIC, but we still especially need good games written in TI BASIC–the language resident in the console.

Please include a comprehensive write-up that describes the program, what it does, how to use it, and any special features. I also like you to include a line number explanation of the program so I won't have to spend hours to figure it out. The line explanation is important to our readers who are learning new programming techniques with each program.

GK: *Have you got some hints on how to "bug-proof" a program?*

CW: One of the best things to do is to have somebody who did *not* help write the program *use* the program. Ask a friend or spouse or neighbor to run your program. Watch the reactions and typical responses. Remember that if your program is published, you can't actually be with the people who are running the program to explain each step–all the information must be contained in the program or the accompanying article. Be sure to test the limits of all parameters and what happens if something is input incorrectly.

GK: *Okay. Now suppose the program has been published. What are some suggestions you can give the readers who are keying in programs?*

CW: First read the article. You have heard the phrase, "When all else fails, read the instructions." The article will tell what the program is supposed to do and what the rules are for playing the game or how to enter information. The article will also tell you what keys to press to cause a certain result. I also recommend using an index card under the lines as you type, so you won't get mixed up. Sometimes the coding looks similar and you could skip a few lines. About every ten lines look at the screen to compare your work with the magazine listing. Keep in mind that it's a good idea to SAVE your work every so often so you don't lose the whole program on an accidental power failure, SHIFT Q, or FUNCTION +. When you are SAVEing progressively larger chunks, turn the tape over, rewind, and record over the lines that were previously there.

GK: *What if the program is keyed-in, but will not RUN without errors?*

CW: My job is to make sure the program works before it is published. An editor frequently has bad dreams that he or she will inadvertently publish the *wrong* version of a program; however, that is an unlikely, rare occurence. PLEASE do not call the editor or the author without first scrutinizing *your* version—and then, it's always better to write. Editors cannot be expected to remember what each line does in hundreds of programs and be able to tell you instantly in a hurried phone call. Some basic ideas of what to look for if you run into errors, include things like:

DATA statements. Check the numbers *and* the commas. A mis-typed number is the most common type of mistake made in keying-in a program. Since DATA statements have lots of numbers, check them carefully.

Check line numbers in ON GOTO and ON GOSUB statements. Do not try to second-guess the author; sometimes some of the numbers in the series will be the same. That means more than one choice can initially go to the same routine.

Compare line numbers on similar coding patterns. Your eyes may have slipped down a few lines if two lines or groups of lines look similar. Do not leave out REM statements. I understand some people leave them out when typing in a program, but the author may GOTO that statement number. (By the way, authors should never GOTO a REMark.)

Check FOR-NEXT loops. Each FOR statement needs a matching NEXT statement with the same loop counter. Loops may be nested.

Read the program Explanation. If you know which piece of logic is causing the problem, you can pinpoint which lines are involved.

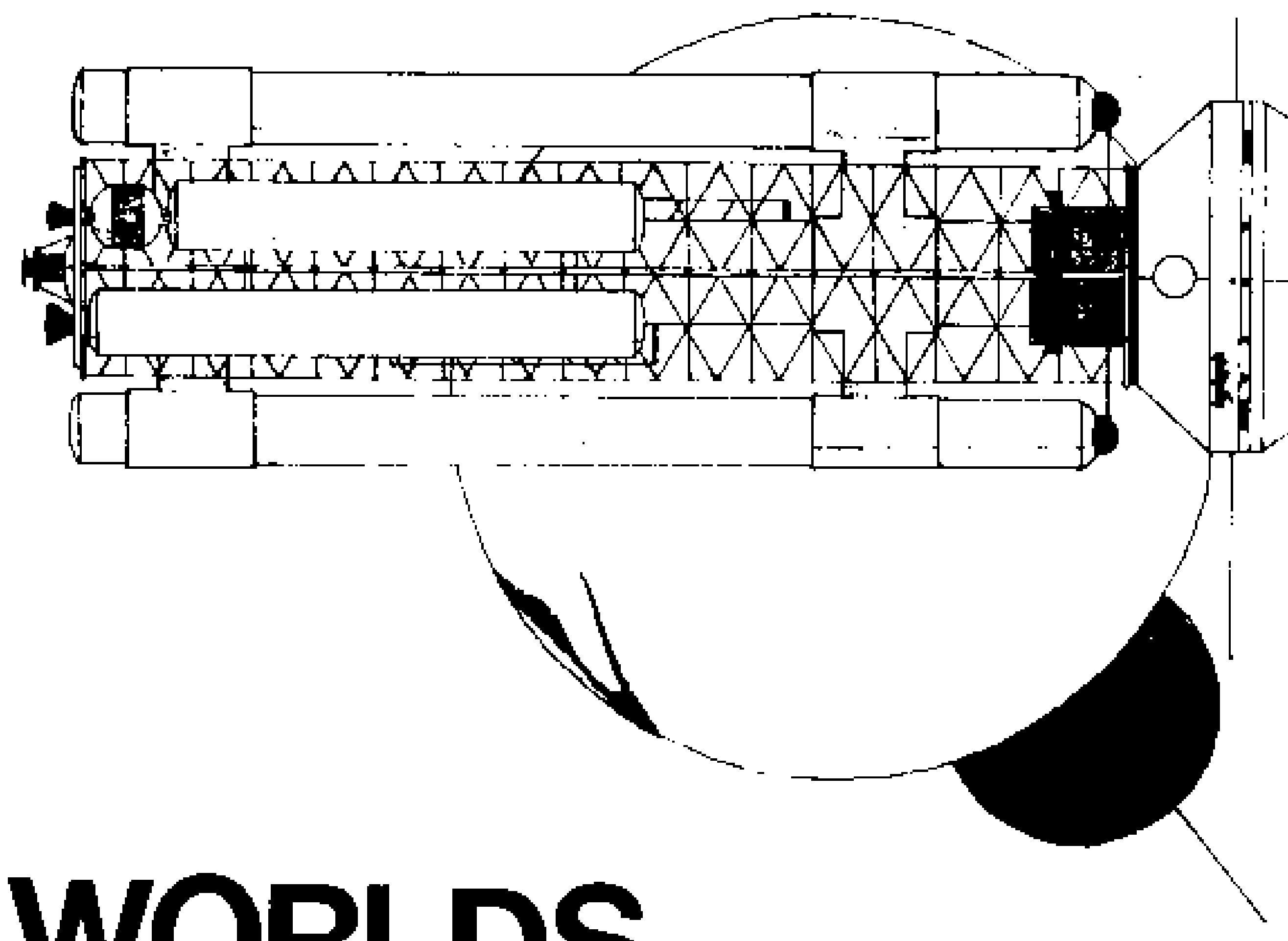
Try "brute force." RUN the program and just check each line that causes the program to stop. LIST the line to see if it contains a syntax error. If the statement looks okay, PRINT any variables that may affect a function or parameter in that line, then you can trace the logic leading up to that line to try to find the error.

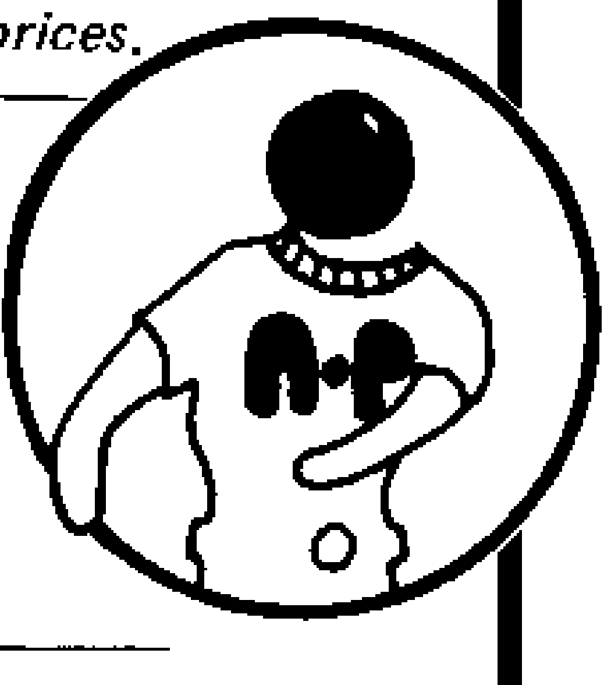
Don't give up. One advantage of keying in your own programs is that you learn programming in the process. You can get many ideas from other people's programs to incorporate in your own programs.

GK: *Any final comments?*

CW: Yes. The quality of the programs available for the Texas Instruments Home Computer is definitely improving. First, there are more computers out and therefore more people programming; and second, those who have had their computers for some time are learning new techniques and refining other techniques so their programs are even more efficient. People are realizing how powerful TI BASIC and Extended BASIC are, and are experimenting in many areas. *99'er Magazine* would like to continue to publish good programs that all users can enjoy and benefit from, and offer programming techniques for both beginning programmers as well as the more experienced ones. We want our readers to be happy with their computers, enjoy using them in a variety of applications, and take full advantage of the TI capabilities.

Support Our Advertisers.
They Too Make This Magazine Possible.

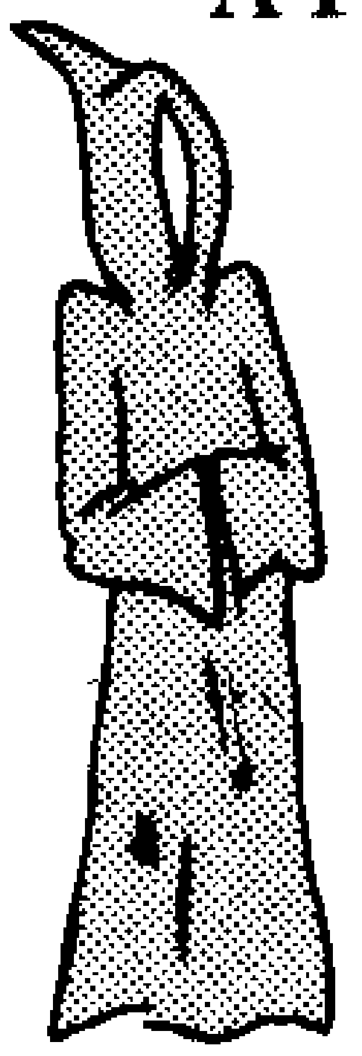




AMERICAN SOFTWARE

99/4 Presents 99/4A

WIZARD'S DOMINION
A Fantasy Adventure



The Wizard is dead. The Voritka Ogres are on the prowl. Only the Wizard's Apprentice, the Evil Wizard, the Hero and the Evil Prince have enough courage to defeat them and rule Wavoria. Inside the caverns of The Wizard's Dominion lay gold, adventure and magical powers. Spend your gold wisely and prepare to meet a Voritka Ogre!

- 3-D Perspective
- Superb Graphics
- Ages 12 to Adult
- Become Master Wizard
- Magical Powers Galore
- Complex Battles
- Many Levels to Conquer
- Hundreds of caves per level

EXTENDED BASIC LANGUAGE

ALL THIS FOR ONLY!! $19.95 cassette
$21.95 disk

3-D MAZE - Enter the world of 3-D MAZE. Look down the hallways, explore them, find the dead ends and the loops. Try to remember in what direction you are going and then find the exit! There are hundreds of mazes, some easy, some very difficult. Come, try to conquer the world of 3-D Maze. Ages 12 to Adult. X Basic $14.95 -cassette, $16.95 - disk

BOMB SQUAD - Time is ticking away. Only 16 seconds left. To stop the clock the timer circuit must be cut. But which one is it? One more test and you should know. The question is, can you figure it out in time?

For high intensity excitement this it the best game we've seen. As the time relentlessly ticks away you must make tests on the bombs circuits and use logic to deduce which wires to cut. Over 20 levels of difficulty make this game a challenge for the beginner or expert. Ages 12 to Adult. X Basic $14.95 - cassette, $16.95 - disk

		Cassette	Disk
• **METEOR SHOWER**	X Basic	$ 9.95	$11.95
• **SKI**	Basic	$14.95	$16.95
• **LAND ON MARS***	Basic	$12.95	$14.95
• **LASER SHIELD**	Basic	$14.95	$16.95
• **SPACE BATTLE 2056**	Basic	$13.95	$15.95
• **MR. FROG**	X Basic	$14.95	$16.95
• **BARNYARD FUN**	Basic	$14.95	$16.95

***Dragon option included!**

Complete instructions included. To save C.O.D. charges, send check or money order plus $1.50 shipping/handling. Mn. residents add 5% sales tax.

ASD&D

AMERICAN SOFTWARE
DESIGN & DISTRIBUTION CO.
P.O. BOX 46
COTTAGE GROVE, MN 55016-0046

NOW AVAILABLE...
from MECA, Inc.
A Music Program for the 99/4
that's **FUN TO USE** and
Really Works
to improve Music Skills

Created by a University Music Professor
For Beginning, Intermediate & Advanced Musicians

NOTE WHIZ	$29.95
Extended Basic	Cassette or Disk (specify)

- Learn to Read Notes Quickly and Accurately
- Four Clefs Included (Treble, Bass, Alto and Tenor)
- Three Levels—Beginner to Advanced
- Plays Like a Game! (Fast-paced/Scoring/Music/Rewards)
- Outstanding Color Graphics
- Easy to Use
- Built-in Rewards for Exceptional Scores
- Record Keeping with Optional Thermal Printer
- FREE INSTRUCTION BOOK (Copy protected to conceal special rewards.)

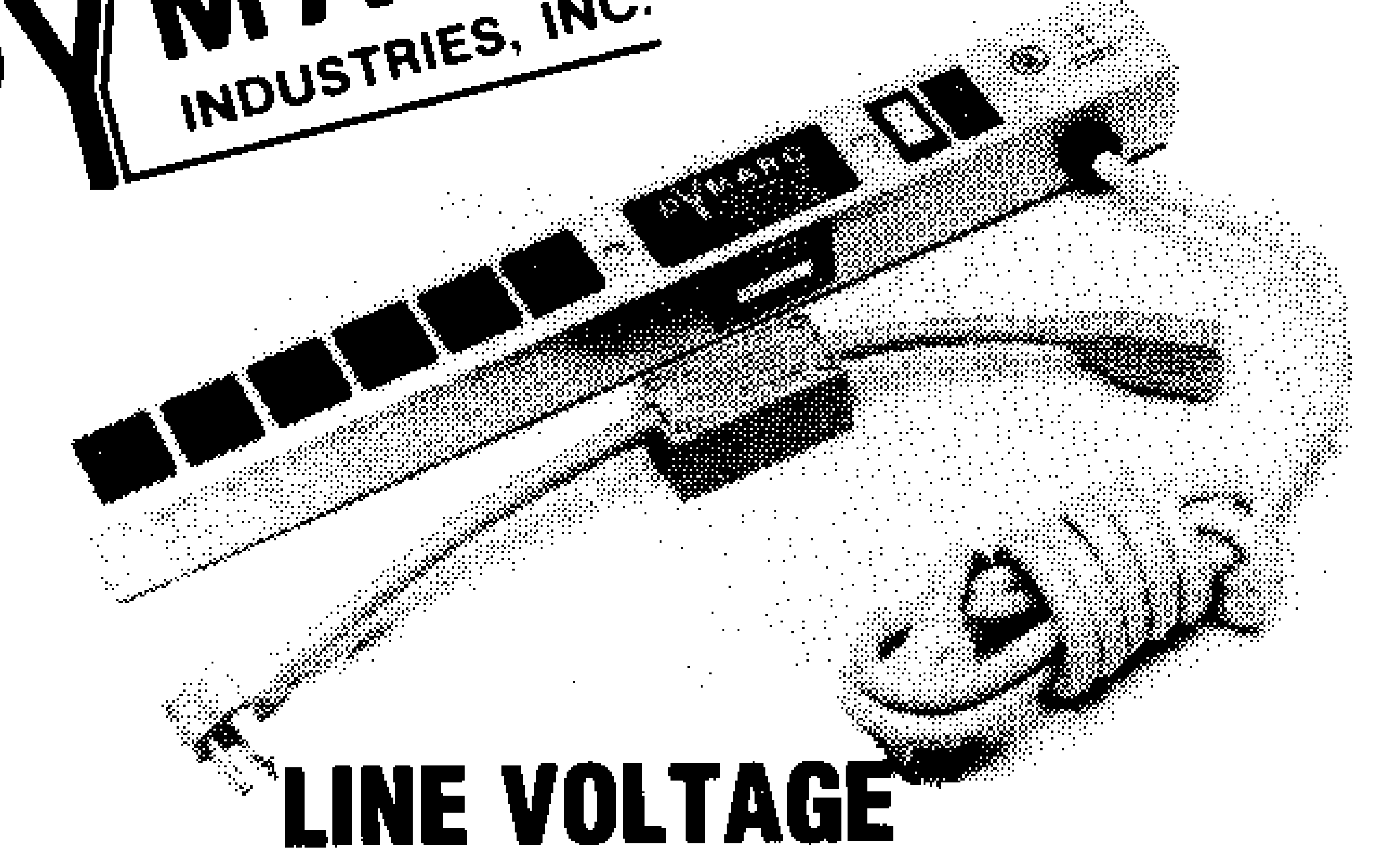
MECA
Inc.

Add $2.00 shipping and handling. Virginia residents, add 4% sales tax.

Send check or money order to:
MECA, Inc.
P.O. Box 5425
Richmond, VA 23220

Other exciting Music products coming soon! Write for details!

DYMARC INDUSTRIES, INC.



LINE VOLTAGE TRANSIENT SUPPRESSORS

Features Parallel Operation 5000 Hits/Second

Clipstrip™ AND *Clipper*™

PROTECTS:
- Computers
- Micro-Computer Systems
- Word Processors
- Cash Registers
- Power Supplies

PROTECTS AGAINST:
- High Energy Voltage Transients
- On-Off Switching
- Lightning Induced Transients
- Inrush of On/Off Power

DYMARC INDUSTRIES, INC.
DEALER INQUIRIES INVITED

Clipstrip 977E UL LISTED
Clipper 678F UL LISTED

WE HONOR MasterCard VISA

7133 Rutherford Rd.
Baltimore, Md. 21207

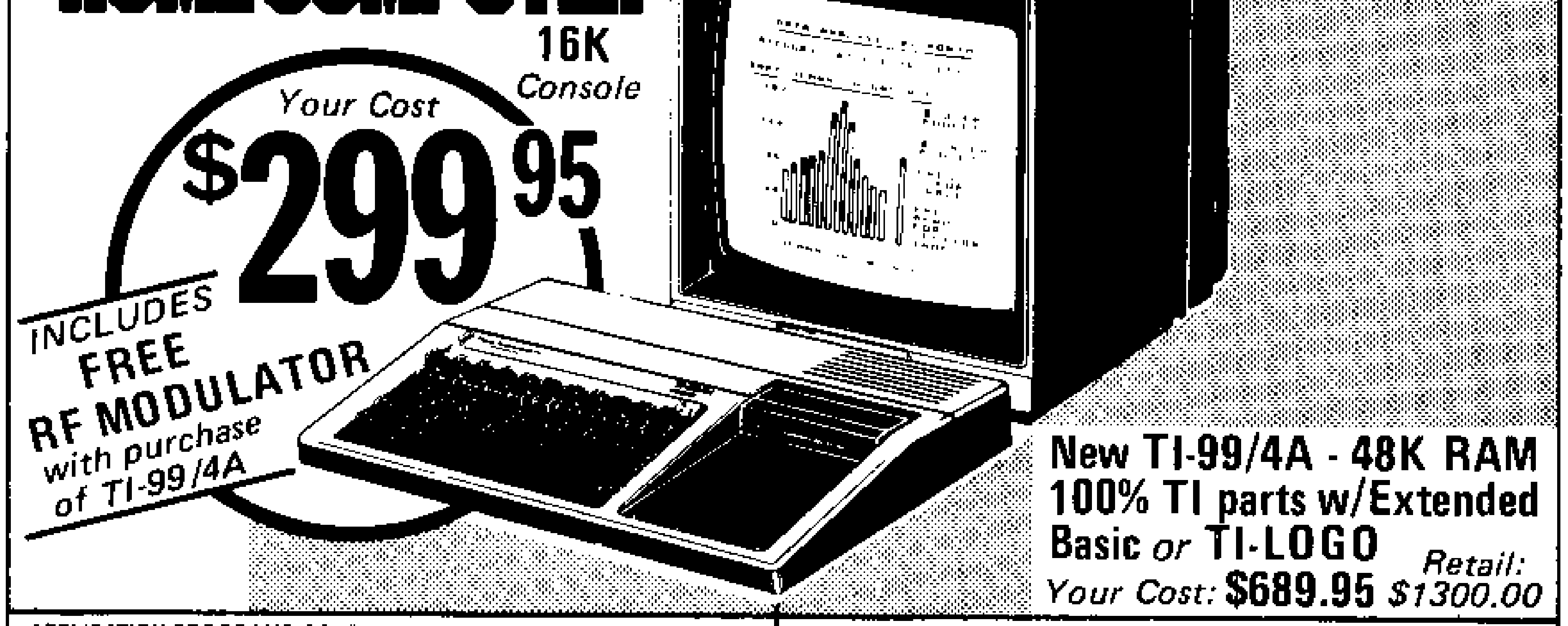
800-638-9098
(301) 298-3130

WE WILL BEAT ANY COMPETITOR'S* ADVERTISED PRICES**

**in this magazine **goods subject to availability*

TI-99/4A HOME COMPUTER

16K *Console*



New TI-99/4A - 48K RAM 100% TI parts w/Extended Basic *or* TI-LOGO
Your Cost: **$689.95** *Retail: $1300.00*

CONSOLE			*Your Cost*
PHC	004A	TI-99 4A Home Computer *inc. FREE RF Modulator*	299.95

PERIPHERALS

PHP	1200	Peripheral Expansion Box**	194.00
PHP	1220	RS-232 Card**	136.00
PHP	1240	Disk Controller Card**	194.00
PHP	1250	Expansion Box Disk Drive Card**	299.00
PHP	1260	Memory Expansion Card** (RAM)	234.00
PHP	1270	P-Code Card**	194.00
PHP	1500	Solid State Speech™ Synthesizer	114.50
PHP	1600	Telephone Coupler (Modem)	169.50
PHP	1700	RS-232 Accessories Interface	169.50
PHP	1800	Disk Drive Controller (One Disk Manager module packed with each Disk Controller)	229.50
PHP	1850	Disk Memory Drive	368.50
PHP	1900	Solid State Printer	298.95
PHP	2200	Memory Expansion (32K RAM)	288.95
PHP	2300	VCR Controller **	569.95
PHP	2400	P-Code Peripheral	314.00
PHA	2100	R F Modulator (TV Adapter)	38.95
PHA	4100	10" Color Monitor	329.00

OPTIONAL ACCESSORIES

PHP	1100	Wired Remote Controllers (Pair)	28.50
PHA	1950	Thermal Paper (2 Pack)	9.50
PHA	2000	Dual Cassette Cable	12.50
PHA	2010	Monitor Cable	17.50
PHA	2020	Audio Adapter (Headphone Jack)	17.50
PHA	2310	Panasonic VCR Controller Cable	89.95
PHA	2320	Sony VCR Controller Cable	89.95
PHA	2330	Pioneer VCR Controller Cable	89.95
PHA	2605	Blank Overlays (4 Pack)	7.50

DOCUMENTATION

PHA	2600	Beginning BASIC Manual (TI-99 4 Only)	8.95
PHA	2601	Users Reference Guide (TI-99 4 Only)	8.95
PHA	2602	Beginning BASIC Manual (TI-99 4A Only)	8.95
PHA	2603	User's Reference Guide (TI-99 4A Only)	8.95
PHA	2606	Creative Programming Computer Competency Series Volume I	8.95
PHA	2607	Creative Programming Computer Competency Series — Volume II	8.95
PHA	2608	Creative Programming Computer Competency Series — Volume III	8.95
PHA	2609	Creative Programming Computer Competency Series — Allstar Projects	8.95
PHA	2610	Creative Programming Computer Competency Series (6 volumes)	79.95

APPLICATION PROGRAMS

Home Management Personal Finance

		Command Modules	
PHM	3006	Home Financial Decisions	24.95
PHM	3007	Household Budget Management (Data storage system is required)	33.50
PHM	3012	Securities Analysis	45.95
PHM	3013	Personal Record Keeping (Data storage system is recommended)	39.50
PHM	3016	Tax Investment Record Keeping (Disk system is required)	58.95
PHM	3022	Personal Real Estate (Data storage system is recommended)	58.95
PHM	3044	Personal Report Generator (Data storage system is recommended)	39.50
		Diskette	
PHD	5001	Mailing List	54.50
PHD	5003	Personal Financial Aids	16.50
PHD	5021	Checkbook Manager	16.50
PHD	5022	Business Aids Library — Finance Management (Extended BASIC Command Module is required)	33.50
PHD	5024	Business Aids Library — Inventory Management (Personal Record Keeping or Statistics Command Module is required)	58.95
PHD	5027	Business Aids Library — Invoice Management (Personal Record Keeping or Statistics Command Module is required)	59.95
PHD	5029	Business Aids Library — Cash Management (Extended BASIC Command Module is required)	33.95
PHD	5036	Business Aids Library · Lease Purchase Decisions	59.95
		Cassette	
PHT	6003	Personal Financial Aids	12.50
PHT	6038	Business Aids Library - Lease Purchase Decisions	54.95

Education Personal Enrichment

		Command Module	
PHM	3040	TI-LOGO (Memory Expansion is required)	$99.95
PHM	3002	Early Learning Fun	24.95
PHM	3003	Beginning Grammar	24.95
PHM	3004	Number Magic	16.50
PHM	3005	Video Graphs	16.50
PHM	3008	Video Chess	54.50
PHM	3010	Physical Fitness	24.95
PHM	3015	Early Reading † (Solid State Speech™ Synthesizer is required)	45.95
PHM	3020	Music Maker (Data storage system is recommended)	33.50
PHM	3021	Weight Control and Nutrition (Data storage system is recommended)	49.95
PHM	3027	Addition and Subtraction I† (Solid State Speech™ Synthesizer is recommended)	33.50
PHM	3028	Addition and Subtraction II† (Solid State Speech™ Synthesizer is recommended)	33.50
PHM	3029	Multiplication I† (Solid State Speech™ Synthesizer is recommended)	33.50
PHM	3043	Reading Fun†*	45.95
PHM	3059	Scholastic Spelling — Level III** (Solid State Speech™ Synthesizer is required)	45.95
PHM	3060	Scholastic Spelling — Level IV** (Solid State Speech™ Synthesizer is required)	45.95
PHM	3061	Scholastic Spelling — Level V** (Solid State Speech™ Synthesizer is required)	45.95
PHM	3062	Scholastic Spelling — Level VI** (Solid State Speech™ Synthesizer is required)	45.95
PHM	3064	Typing Tutor**	33.50
		Diskette	
PHD	5007	Teach Yourself BASIC	28.95
PHD	5009	Music Skills Trainer	24.95

APPLICATION PROGRAMS CONTINUED

Education Personal Enrichment Continued

			Your Cost
		Diskette	
PHD	5011	Computer Music Box	16.50
PHD	5018	Market Simulation	16.50
PHD	5019	Teach Yourself Extended BASIC (Extended BASIC Command Module is required)	20.95
PHD	5020	Music Maker Demonstration (Music Maker Command Module is required)	12.50
PHD	5023	Basketball Statistics (Extended BASIC Command Module is required)	20.95
PHD	5026	Bridge Bidding I	24.95
PHD	5030	Speak & Spell™ Program (Solid State Speech™ Synthesizer is required)	24.95
PHD	5031	Speak & Math™ Program (Solid State Speech™ Synthesizer and Terminal Emulator II are required)	24.95
PHD	5039	Bridge Bidding II	24.95
PHD	5041	Bridge Bidding III	24.95
PHD	5042	Spell Writer (Terminal Emulator II Command Module and Solid State Speech™ Synthesizer are required)	24.95
PHD	5067	Teach Yourself Beginning BASIC**	24.95
		Cassette	
PHT	6007	Teach Yourself BASIC	24.95
PHT	6009	Music Skills Trainer	20.95
PHT	6011	Computer Music Box	12.50
PHT	6018	Market Simulation	12.50
PHT	6019	Teach Yourself Extended BASIC (Extended BASIC Command Module is required)	16.50
PHT	6026	Bridge Bidding I	20.95
PHT	6031	Speak & Math™ Program (Solid State Speech™ Synthesizer and Terminal Emulator II are required)	20.95
PHT	6039	Bridge Bidding II	20.95
PHT	6041	Bridge Bidding III	20.95
PHT	6042	Spell Writer (Solid State Speech™ Synthesizer and Terminal Emulator II Command Module are required)	20.95
PHT	6067	Teach Yourself Beginning BASIC**	20.95

Entertainment

		Command Module	
PHM	3009	Football	24.50
PHM	3018	Video Games I	24.50
PHM	3023	Hunt the Wumpus	20.95
PHM	3024	Indoor Soccer	24.50
PHM	3025	Mind Challengers	20.95
PHM	3030	A-Maze-Ing	20.95
PHM	3031	The Attack††	32.95
PHM	3032	Blasto††	20.95
PHM	3033	Blackjack and Poker††	20.95
PHM	3034	Hustle††	20.95
PHM	3036	ZeroZap††	16.50
PHM	3037	Hangman††	16.50
PHM	3038	Connect Four††	16.50
PHM	3039	Yahtzee††	20.95
PHM	3041D	Adventure (Pirate Adventure Diskette Game Included)	39.95
PHM	3041T	Adventure (Pirate Adventure Cassette Game Included)	39.95
PHM	3052	Tombstone City 21st Century	32.95
PHM	3053	TI Invaders	32.95
PHM	3054	Car Wars	32.95
PHM	3057	Munch Man**	32.95
		Diskette	
PHD	5002	TI-Trek (with speech)	12.50
PHD	5010	Mystery Melody	12.50
PHD	5015	Oldies But Goodies Games I	16.50
PHD	5017	Oldies But Goodies — Games II	20.95
PHD	5025	Saturday Night Bingo (Solid State Speech™ Synthesizer is required)	24.95
PHD	5037	Draw Poker	20.95
		Adventure Series	
PHD	5043	Pirate Adventure	24.95
PHD	5046	Adventureland	24.95
PHD	5047	Mission Impossible	24.95
PHD	5048	Voodoo Castle	24.95
PHD	5049	The Count	24.95
PHD	5050	Strange Odyssey	24.95
PHD	5051	Mystery Fun House	24.95
PHD	5052	Pyramid of Doom	24.95
PHD	5053	Ghost Town	24.95
PHD	5054	Savage Island I & II	32.95
PHD	5056	Golden Voyage	24.95

APPLICATION PROGRAMS CONTINUED

Entertainment Continued

			Your Cost
		Cassette	
PHT	6010	Mystery Melody	7.95
PHT	6015	Oldies But Goodies — Games I	12.50
PHT	6017	Oldies But Goodies Games II	16.50
PHT	6025	Saturday Night Bingo (Solid State Speech™ Synthesizer is required)	20.95
PHT	6037	Draw Poker	16.50
		Adventure Series	
PHT	6043	Pirate Adventure	24.95
PHT	6046	Adventureland	24.95
PHT	6047	Mission Impossible	24.95
PHT	6048	Voodoo Castle	24.95
PHT	6049	The Count	24.95
PHT	6050	Strange Odyssey	24.95
PHT	6051	Mystery Fun House	24.95
PHT	6052	Pyramid of Doom	24.95
PHT	6053	Ghost Town	24.95
PHT	6054	Savage Island I & II	32.95
PHT	6056	Golden Voyage	24.95

OTHER APPLICATION PROGRAMS

		Command Modules	
PHM	3000	Diagnostic	24.95
PHM	3001	Demonstration	58.95
PHM	3011	Speech Editor (Solid State Speech™ Synthesizer is required)	35.95
PHM	3014	Statistics (Data storage system is recommended)	35.95
PHM	3026	Extended BASIC	75.95
PHM	3035	Terminal Emulator II	39.50
PHM	3055	Editor Assembler*	89.95
PHM	3058	Mini-Memory*	89.95
		Diskette	
PHD	5004	Programming Aids I	12.50
PHD	5005	Programming Aids II	20.95
PHD	5006	Math Routine Library	24.95
PHD	5008	Electrical Engineering Library	24.95
PHD	5012	Programming Aids III	16.50
PHD	5013	Graphing Package	16.50
PHD	5016	Structural Engineering Library	24.95
PHD	5044	SMU Circuit Analysis I**	124.95
PHD	5063	UCSD-Pascal Compiler**	109.95
PHD	5064	UCSD p-System Assembler Linker**	89.95
PHD	5065	UCSD p-System Editor Filer Utilities**	67.95
PHD	5066	TI PILOT**	69.95
		Cassette	
PHT	6004	Programming Aids I	8.95
PHT	6006	Math Routine Library	20.95
PHT	6008	Electrical Engineering Library	20.95
PHT	6013	Graphing Package	12.50
PHT	6016	Structural Engineering Library	20.95
PHT	6044	SMU Circuit Analysis I**	124.95

SOFTWARE LIBRARIES

PHL	7001	The Home Financial Manager	119.95
PHL	7002	The Family Entertainer	74.95
PHL	7003	The Elementary Educator	89.95
PHL	7004	The Music Educator	56.95
PHL	7005	The Super Programmer	99.95
PHL	7006	The Speaking Math Teacher	99.95
PHL	7007	The Speaking Reading Teacher	92.95
PHL	7008	The Speaking Scholastic Spelling Teacher**	181.95
PHL	7009	The TI Arcade Game Series	99.95
PHL	7010	The Milton Bradley Game Series	94.95
PHL	7011	The Computer Introductory Package	99.95

†Developed by Scott Foresman
††Developed by Milton Bradley The Attack Blasto Hustle ZeroZap Connect Four and Yahtzee are trademarks of Milton Bradley
*Available only until replaced by peripheral card
**Pending FCC approval available late First Quarter 1982
UCSD UCSD Pascal and UCSD p System are all trademarks of the Regents of the University of California

Other products from Texas Instruments:

TI-59 Programmable Calculator $169.95
LESS $20 Rebate from TI - ALSO FREE Library (with TI-59 purchase)

TI-58C Programmable Calculator $99.95
ALSO FREE LIBRARY , $45 value (with purchase of TI-58C)

OLYMPIC SALES COMPANY

CABLE:"OLYRAV" LSA
TELEX: 67 34 77
Main Showroom & Offices:
216 So. Oxford Ave.
Los Angeles, CA 90004
WE HONOR VISA and MASTERCHARGE

YOUR BANK VISA
master charge

ORDER DESKS open 6 Days a Week!
7:00 AM to 7:00 PM Mon-Sat.
Order Desks: (213) 739-1130
TOLL FREE*(out of CA)* 800-421-8045

We stock all TI models, accessories, etc., plus others at discounted prices!

We carry close to $5,000,000 inventory at all times. Call us for any goods not featured in this ad. Corp. accts. invited. We carry many other mfgs., call for info. Good subject to availability; this ad supercedes all previous ads; fob our warehouse; prices subject to change without notice; not responsible for typographical errors; all orders subject to verification; minimum shpg & hndlg $5.95. Send $2 (or $5 foreign) for our famous catalog.

HOW TO START

By Lawrence R. De Rusha, Jr.

*International Home Computer
Users' Association (ICA)
P. O. Box 371
Rancho Santa Fe, CA 92067*

When I first purchased my 99/4, I was glad to have the plug-in modules from Texas Instruments. I could play football or do my household budget without having to learn programming or how to load a program from cassette or disk. This, I was sure, was a "user-friendly" system. Later, I decided to experiment with TI BASIC. After completing my first program, I experienced a great sense of accomplishment. But when I decided to tackle the conversion of a motion-picture budget system into a program for the 99/4, BASIC wasn't basic anymore . . .

I called the local store where I purchased the system, and the best advice given me was to join the local users' group. And, as it turned out, the best place to learn about the system was indeed a local users' group.

Now, as the President of a users' group, I watch new members join with some of the same expectations that I once had. The local group is the best all-around place for new users and for experienced hackers to communicate.

Judging from the mail received at ICA, there are many owners who are interested in starting a local users' group. Therefore, Part I of this series is designed to answer some of the most frequently asked questions about starting such a group. Part 2 will answer questions most often asked about keeping the group running smoothly.

When personal computers were sold almost exclusively as kits, local users' groups sprang up for several reasons: First, an individual with a recently acquired kit could get first-hand aid from other kit owners and builders. Second, many of these first computer users were hardware experimenters, and joining a group meant getting new ideas on modifying the computer, and of new PC board designs.

Some of the users' groups today still remain hardware experimenters' clubs. Now, however, many new ones are forming that include the neophyte computer users. Their needs have created users' groups centered around the brand of computer purchased. At the group meetings, they can hear of problems other people have had, gain early news on new products for the system, learn programming, exchange programs, and generally socialize with others of common interests.

Today, there are more than 25 TI groups around the world. More are surfacing every month, attesting to the growth of the TI-99/4A.

Be prepared! Remember the scout motto? If you're planning to start a users' group, be prepared to spend time and energy and to be frustrated occasionally. You may want to outline a plan of action and determine how much time it will take and how much help you will need.

The following are questions most asked about starting a local users' group:

What is the best way for me to let other TI owners know that I am starting a new group?

Post a notice at all of the local TI dealers in your area. Notify other computer stores and dealers in the area, even if they do not carry the TI-99/4A system. Notices can be posted on bulletin boards at local shopping centers. This may catch the eye of the new TI owner who hasn't begun to frequent the local dealer. Remember, the TI user generally is *not* a computer hobbiest, but merely a new computer user.

Next, notify the local media: radio stations, local papers, even local television stations. Send your notice in care of the community bulletin board or public service director. Be sure you give them details on how the potential members can contact you. Campus newspapers and company newsletters are also solid areas to pursue for potential contact with TI users.

Be sure to contact organizations that can assist you. The *TI User's Newsletter, 99'er Magazine, ICA Update Newsletter, 99/4 Users of America Newsletter,* and the *International 99/4 Users' Group Newsletter* are all valuable sources for help in publicizing your intentions.

Where can we find a place to meet?

While you're a small group, meeting at someone's home will be suitable. As you grow, you may want to try the local TI dealer. Again, you may outgrow this space, at which time you should try for community rooms at shopping centers or chain stores. Local banks also may have meeting rooms available.

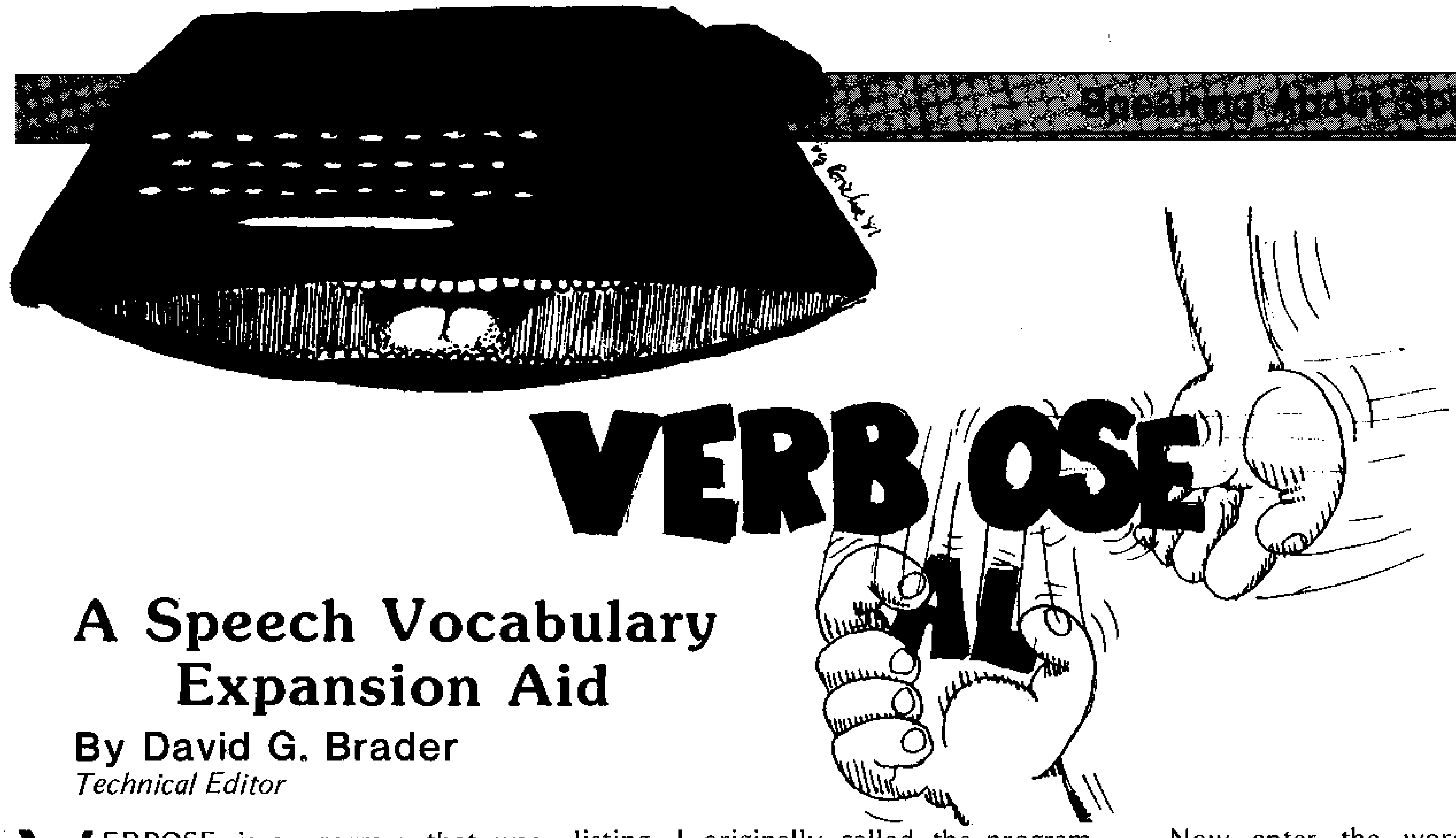
It is important that your group be able to meet at the same time and in the same place each month. There are several reasons for this. First, it eliminates an undue hardship on officers to notify everyone of the time, place and directions to the new meeting facility. It also eliminates constantly having to search out a new meeting room. Further, it makes it easier to recommend the club if people know when and where your meeting is without having to look it up.

Why do we need bylaws and organization plans?

If your group is small and you are sure your group's interests can be served adequately with an informal structure, there is no compelling reason to structure the club.

If you have more than seven or eight members, however, you may find it necessary to collect dues (for a newsletter, notices or meetings, etc.) and to assign group responsibilities to members. At this point, you should set down your bylaws. They can be simple or very complete, but these rules will govern the operation of the group.

In Part 2 of this series, I will answer some frequently asked questions about running a users' group—for example, setting up a software exchange, speakers programs, equipment swaps, volunteer programs, and producing a newsletter.



VERBOSE

A Speech Vocabulary Expansion Aid

By David G. Brader
Technical Editor

VERBOSE is a program that was written in an evolutionary manner. One thing just lead to another. The story goes something like this . . .

One day I decided to make a program to speak a simple sentence. After all, the TI Speech Synthesizer sits there all the time, it must have something to say . . . Well anyway, I came up with a simple sentence (don't remember what it was now). The program was entered and run.

Wow–Almost half of the words in the sentence were not in the resident vocabulary! It was clearly time for me to read the manual that came with the unit. Surprise. I found it had a vocabulary limited to three or four hundred words. That was not enough for me. Further research was definitely called for . . .

Reading the TI Extended BASIC manual, I found a program on page 206 that allowed adding standard suffixes to resident vocabulary words (i.e., ed, ing, s). After playing with this "suffix" program awhile, I realized it would be possible to "concatenate" two resident vocabulary words to produce a totally new word (i.e., therefore, meanwhile, or update). I wrote a routine to do this. Once this concatenation routine was working, it seemed like a speech tool starting to evolve.

It would be nice, I thought, to have the results of the concatenation routine printed in the form of data statements. I could then write these data statements (that contain the new word's speech data) into other programs that needed to speak the new word. So next, I generated a routine to do this, and added it to the concatenation routine.

All of these routines, including a method of building a vocabulary file on disk, were combined into a nice, neat, simple-to-use program. The result is Listing #4. As you can see from the listing, I originally called the program "WORD BUILDER." When I decided to write an article on it, however, the name seemed too mundane. So in a fit of cleverness, I renamed the program "VERBOSE." My wife and other friends just shook their heads and groaned . . .

A TV picture is worth a thousand words right? Well, perhaps not quite, so I have combined some text with screen images to guide you through the operation of VERBOSE. Before you start the VERBOSE program, make sure you have either the TI Extended BASIC or TI Speech Editor Command Module plugged in. VERBOSE uses the SPGET and SAY subroutines that are available in these modules. OK, now you're ready to load VERBOSE and type RUN . . .

```
      + + + WORD BUILDER + + +
ENTER NUMBER OF YOUR CHOICE
   1 – JOIN TWO WORDS
   2 – PRINT SPEECH DATA
   3 – STORE NEW WORD ON DISK
   4 – EXIT
?
```

Here we are at the main menu screen. Let's create a new word by joining two words. The new word that we will generate will be REWRITE and will be made from vocabulary words READ and RIGHT. Enter 1 and press the ENTER key.

```
   ENTER FIRST WORD TO JOIN
?
```

We are asked for the first word that will be used in the joining. ENTER READ and press ENTER.

```
   ENTER SECOND WORD TO JOIN
?
```

Now enter the word RIGHT and press ENTER.

```
   ENTER THE SPELLING OF THE
NEW WORD
?
```

Type in REWRITE and then press ENTER.

```
   TRUNCATE HOW MANY BYTES?
```

OK, don't panic here! VERBOSE just wants to know how much of the first word (READ) to truncate before it combines it with the second word (RIGHT). We don't know how much, so we make a wild guess of, say, 34. What we want is to truncate the AD from READ and combine that sound with RIGHT. As soon as you press ENTER this time, the TI-99/4A will say the new word for you . . .

```
   SAY AGAIN? (Y OR N)
```

Here you can answer the question with Y as many times as you like to check the sound of the new word. After hearing enough of it, enter N.

```
   SAY AGAIN? (Y OR N)N
     1 – CHANGE SOME MORE
     2 – BACK TO MAIN MENU
?
```

I don't think the new word sounded quite right so enter "1" and press ENTER.

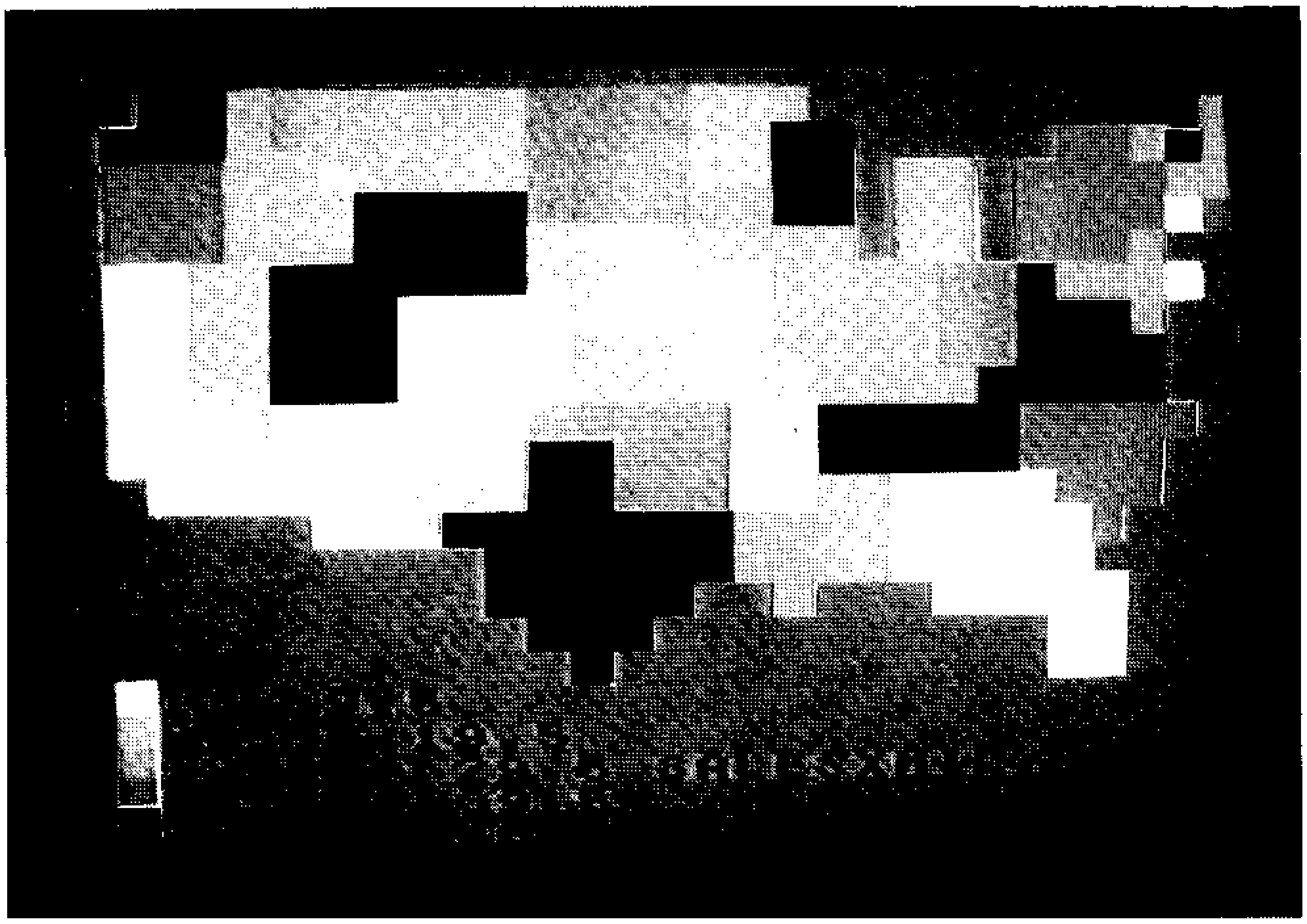
```
   TRUCATE HOW MANY BYTES?
```

This time enter "55" bytes and press ENTER.

Listen to the word as many times as you like. As for me, it sounds close enough to use. When you are satisfied, return to the main menu.

Continued on p. 28

COLOR MAPPING



and the TI-99/4A

By Dr. Borden D. Dent

Professor of Geography and Cartography at Georgia State University.

One of the principal features of the new technology exhibited by low-cost home computers is their graphic capabilities. But an often overlooked potential of these small computers' graphic capabilities is in the area of mapping—especially statistical mapping. Statistical mapping is not new; cartographers have used the methods described in this article for decades, and sophisticated mapping programs that run on large mainframe computers have been available (from Harvard University and elsewhere) for a number of years. Their application for the small computer field, however, especially in the classroom setting, should be further explored and documented.

The program described in this article, "United States Choropleth Map," was written for the TI-99/4. No peripherals are needed, except for a cassette recorder to store the program. Therefore, anyone with the console can get started immediately and experience the excitement of computer mapping. The program should benefit a large number of users: For example, classroom teachers, from the upper elementary grades through college classrooms in geography, can utilize it; sales and marketing managers, and others interested in the spatial distribution of goods and services may also find it especially useful; and political scientists can easily see national election results displayed almost instantaneously.

Choropleth Mapping

Simply defined, choropleth mapping has been likened to a spatial table. Enumeration units, which can be census tracts, counties, states, or other small area "geography" are symbolized by different area patterns depending on the values they represent. Typically, the original data are divided into a number of data classes (map classes). The individual enumeration units will be symbolized according to the map class into which their data value falls. The reason for the classing is that it is usually impractical, or not feasible, to apply an area pattern for *each* data value.

Classing, of course, is similar to a sieve; individual values "fall" into each group depending on the class limits. This results in a generalization, and the final map is a *simplification* of the original data. Nonetheless, choropleth mapping has a number of advantages over a simple table of values. It provides a third, or spatial dimension to a rather dull list of values in tabular format. In the bibliography, I've listed several good books that discuss the methods and rationale of this form of mapping.

4239 Rue St. Michel
Stone Mountain, Georgia 30083

Symbolization on choropleth maps takes on several different forms. In the case of black and white mapping, the enumeration units are symbolized by area *patterns* to differentiate each class from all others. Also often used are different *shades* of grey, ranging from white to black. Color symbolization includes two kinds: (1) different hues (such as green, red, blue, etc.) for the various classes, or (2) different values (shades) of the *same* color. The present program uses the second method.

Main Features of the Program

Figure 1 illustrates the main components of the program's logic, and Figure 2 lists the most important variables. I wrote the program with flexibility in mind: new subroutines can be incorporated as different versions are developed. Lines 10 to 410 of the program are used for an opening screen, displaying the program name.

The first section, Program Instructions, provides the option list and incorporates directions for data input. The present version accommodates only data input from the keyboard. (You may wish to add program statements to read

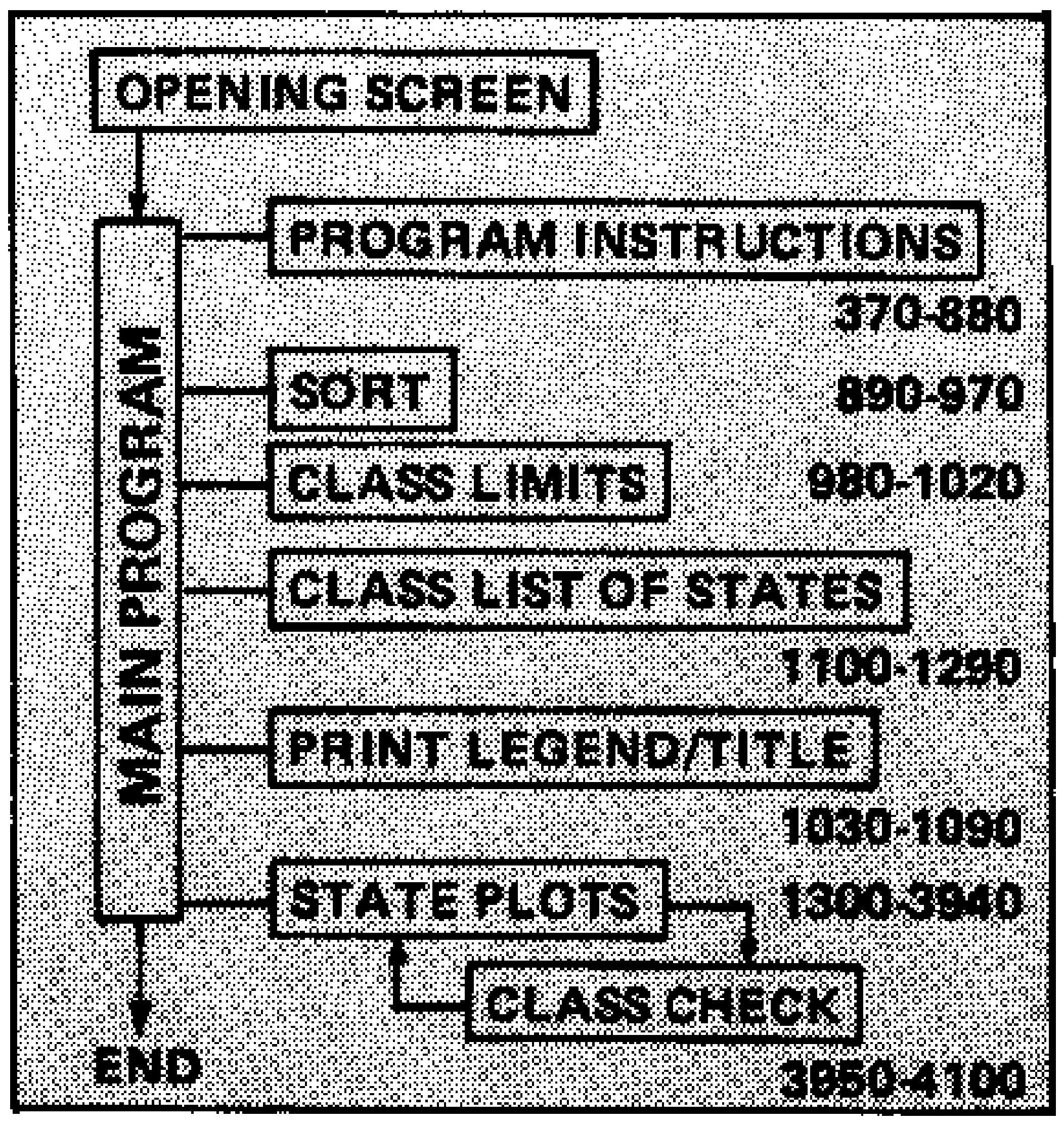


Figure 1 - Main program logic, showing subroutines, of United States Choropleth Map.

SC	- Map background color
MC	- Map color (blue or green)
C(1-6)	- Map class colors
V(1-50)	- Values of each state to be mapped
TTS	- Map title
X1	- Limit for class 1
X2	- Limit for class 2
X3	- Limit for class 3
X4	- Limit for class 4
K(1-6)	- Character sets
S(1-6)	- ASCII character identifiers
SNS(1-50)	- States' names
NN	- State's number

Figure 2 - Principle variables used in mapping program.

data from a file system.) The data is input by entering the values to be mapped for each state, by alphabetical ordering of the states.

After the data is entered, the main program directs the flow to a simple bubble sort subroutine, where the data values are sorted into ascending order. The data values are then classed, and the class limits are selected in the Class Limits section. There are a number of ways in which data may be classed. This program will class the data values into "quintiles"—that is, into *five* classes having the same *number* of values in each class. As the data set has been arrayed in ascending order, the values of the class limits is rather easily computed.

Program flow is next directed to printing. With the TI-99/4 and the Basic language supplied with the standard computer, printed ASCII characters must be displayed *before* the color graphic blocks are called on the screen. If not, scrolling will move the color graphics off the screen. The Print Legend and Title subroutine displays the classed values and user-chosen title on the lower portion of the screen.

State Plots is next. Each state is assigned an ordinal number based on its alphabetical rank (1-50). As each state is encountered, flow is directed to a subroutine, Class Check, in which the state's ordinal number is used to determine which color the state should be.

Outlines of the states are not variable, but, of course, the color (symbolization) varies depending on which class each state falls. Flow continues until each state has been displayed on the screen. A color graphic block is displayed adjacent to the printed legend values at the bottom of the screen. The program ends with a GO TO statement; the screen will display the map until the user presses **SHIFT C** or **FCTN** = to BREAK the program.

Mapping on the TI-99/4

The color graphic capabilities of the TI-99/4 include a screen which is divided into 32 columns and 24 rows—each block addressable by a row and column identifier in the CALL HCHAR and VCHAR commands. Any of 16 colors can be specified (including transparent). Further resolution is possible by using the CALL CHAR command, with which, through the use of hexidecimal code, the user can specify the "on" or "off" condition of 64 dots in each graphic block. The present program

Continued on p. 54

Verbose . . . from p. 25

```
        +++ WORD BUILDER +++
ENTER NUMBER OF YOUR CHOICE
  1 – JOIN TWO WORDS
  2 – PRINT SPEECH DATA
  3 – STORE NEW WORD ON DISK
  4 – EXIT
?
```

Here we are back at the ranch. Let's print the data for our new word by selecting Option 2. Don't forget to press ENTER. I'm not going to remind you about that anymore 'cause you've got the ENTER key down pat . . .

```
  ENTER THE WORD WHOSE DATA
  YOU WANT TO PRINT – –
?
```

After you enter REWRITE and press the you-know-what, the printer will output what you see in Listing #1. It didn't work? Well your printer must be set up differently from mine. Go to Listing #4 and modify line 870 of VERBOSE (the OPEN statement for the printer) to match your setup. If you don't have a printer, delete lines 870 and 1070. Also modify lines 940, 950, 990 and 1060 by deleting the "#1:" of each print statement. Now, instead of going to the printer, everything will go to the screen of the TI-99/4A. The last change is to enter this line:

```
1070 INPUT F$
```

Now it will all stay on the screen (so you can copy it on paper) until you press ENTER.

For those of you that got here without problems, let's go on; the others can catch up later . . . .

Look over Listing #2. This is a sample TI BASIC program that shows how the data statements for VERBOSE can be used. You will note the data statements for the word REWRITE are entered in lines 360-490 of Listing #2. Lines 280-330 build the string "E$" which will cause REWRITE to be spoken. The FOR-NEXT loop here is terminated when the last byte is read. This number (133) was the number of bytes printed out for REWRITE by VERBOSE. The subroutines SAY and SPGET are explained in the speech synthesizer manual.

```
        +++ WORD BUILDER +++
ENTER NUMBER OF YOUR CHOICE
  1 – JOIN TWO WORDS
  2 – PRINT SPEECH DATA
  3 – STORE NEW WORD ON DISK
  4 – EXIT
?
```

It is very tiring to enter all those data statements of the previous sample program. For those of you with a disk system, an easier method of saving and using words from VERBOSE is available with Option #3. Go ahead and select it now.

```
    PUT THE DISK WITH "WORDS"
  FILE IN DRIVE ONE.
  PRESS ENTER WHEN READY
```

The disk that you wish to keep your new vocabulary words on should now be placed in disk drive number 1. The words that will be saved will be appended to a file called WORDS on this diskette. See line 1160 of Listing #4 for the OPEN statement for this file.

```
    PUT THE DISK WITH "WORDS"
  FILE IN DRIVE ONE.
  PRESS ENTER WHEN READY
  ENTER THE WORD WHOSE DATA
  YOU WANT TO SAVE – –
?
```

Enter the word REWRITE to save. The disk drive will run and then VERBOSE will return to its main menu. Use this option to save a few more words that you choose. Then run the Spelling Test Game in Listing #3 using the resultant WORDS file.

The Spelling Test Game program will accept up to 20 words for the WORDS file. It then speaks each word, checks the spelling that is input, and keeps score. Any children in your home should find it useful for spelling drill.

Study Listing #3 and notice lines 230-270. The WORDS file has a pair of strings for each word saved. The first string contains the spelling of the word. The second string contains the actual speech data.

As mentioned earlier, the program listing for VERBOSE is listing #4.

A final note of caution:
Once you start that TI-99/4A talking, BEWARE—you may have trouble getting a word in edgewise . . .

Listing 1

```
100 REM ++++++++++++++++++++
110 REM + SPEAK-- "THIS    +
120 REM + PROGRAM HAD A    +
130 REM + REWRITE."        +
140 REM +                  +
150 REM +                  +
160 REM +THIS SAMPLE SHOWS+
170 REM +HOW THE OUTPUT OF+
180 REM +"VERBOSE" IS USED+
190 REM +IN TI BASIC....   +
200 REM +                  +
210 REM +                  +
220 REM ++++++++++++++++++++
230 REM
240 CALL SPGET("THIS",A$)
250 CALL SPGET("PROGRAM",B$)
260 CALL SPGET("HAD",C$)
270 CALL SPGET("A",D$)
280 RESTORE 360
290 E$=""
300 FOR I=1 TJ 133
310 READ X
320 E$=E$&CHR$(X)
330 NEXT I
340 CALL SAY("",A$,"",B$,"",C$,"",D$,"",E$)
350 REM ** REWRITE **
360 DATA 96,0,42,161,19,49,92,60,149,149
370 DATA 78,86,51,117,147,223,26,61,196,197
380 DATA 69,253,170,93,103,231,176,108,167,10
390 DATA 158,83,211,151,156,188,40,21,157,106
400 DATA 180,178,42,89,125,96,0,85,162,101
410 DATA 33,221,57,28,139,154,142,144,176,116
420 DATA 172,106,58,92,162,67,137,105,248,82
430 DATA 142,49,39,169,209,7,179,84,220,175
```

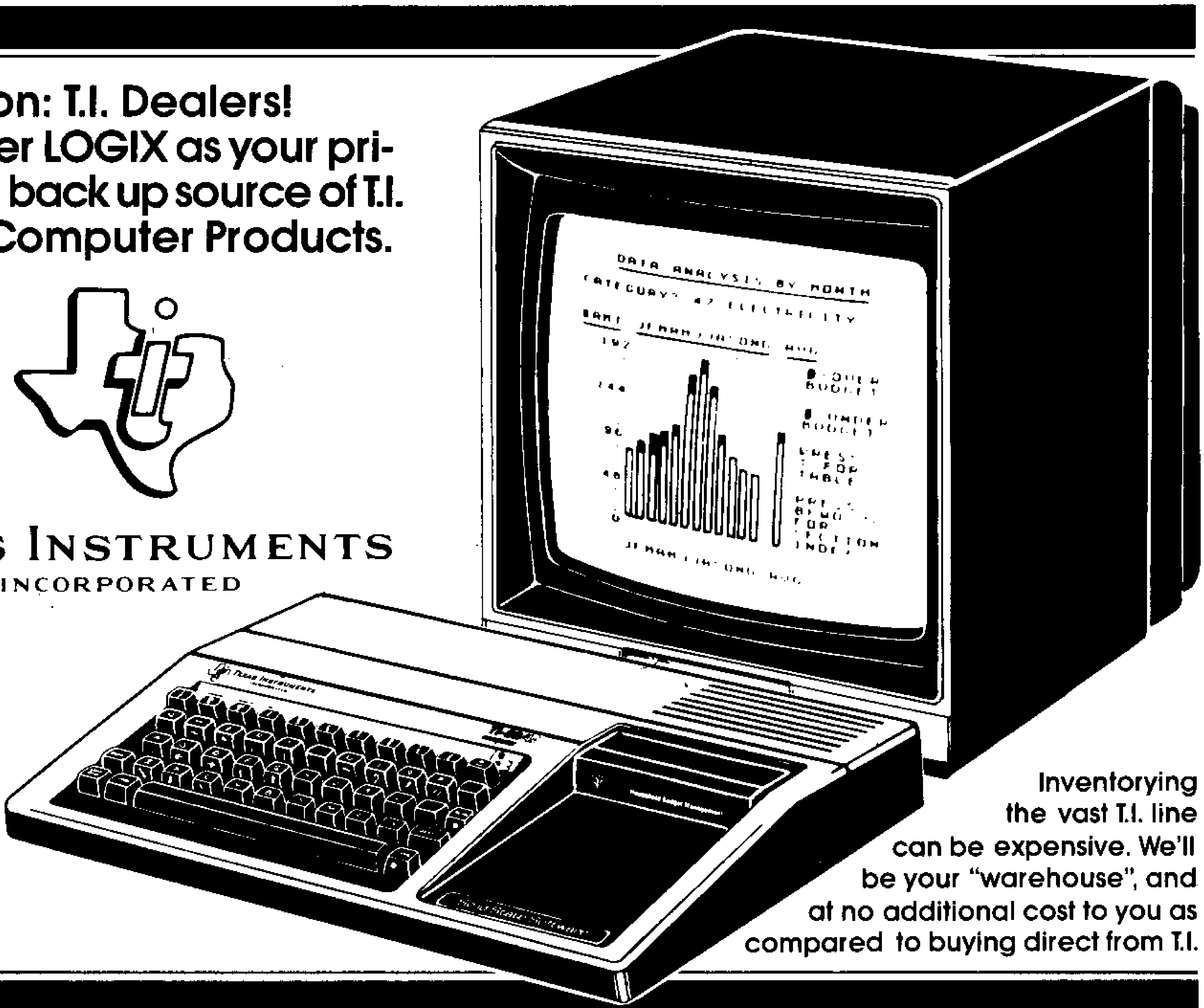
Continued on p. 56

LOGIX

One of the largest full line distributors of Texas Instruments 99/4A Home Computer Hardware, Software, Peripherals and other T.I. Products.

Attention: T.I. Dealers! Consider LOGIX as your primary or back up source of T.I. Home Computer Products.

TEXAS INSTRUMENTS
INCORPORATED



Inventorying the vast T.I. line can be expensive. We'll be your "warehouse", and at no additional cost to you as compared to buying direct from T.I.

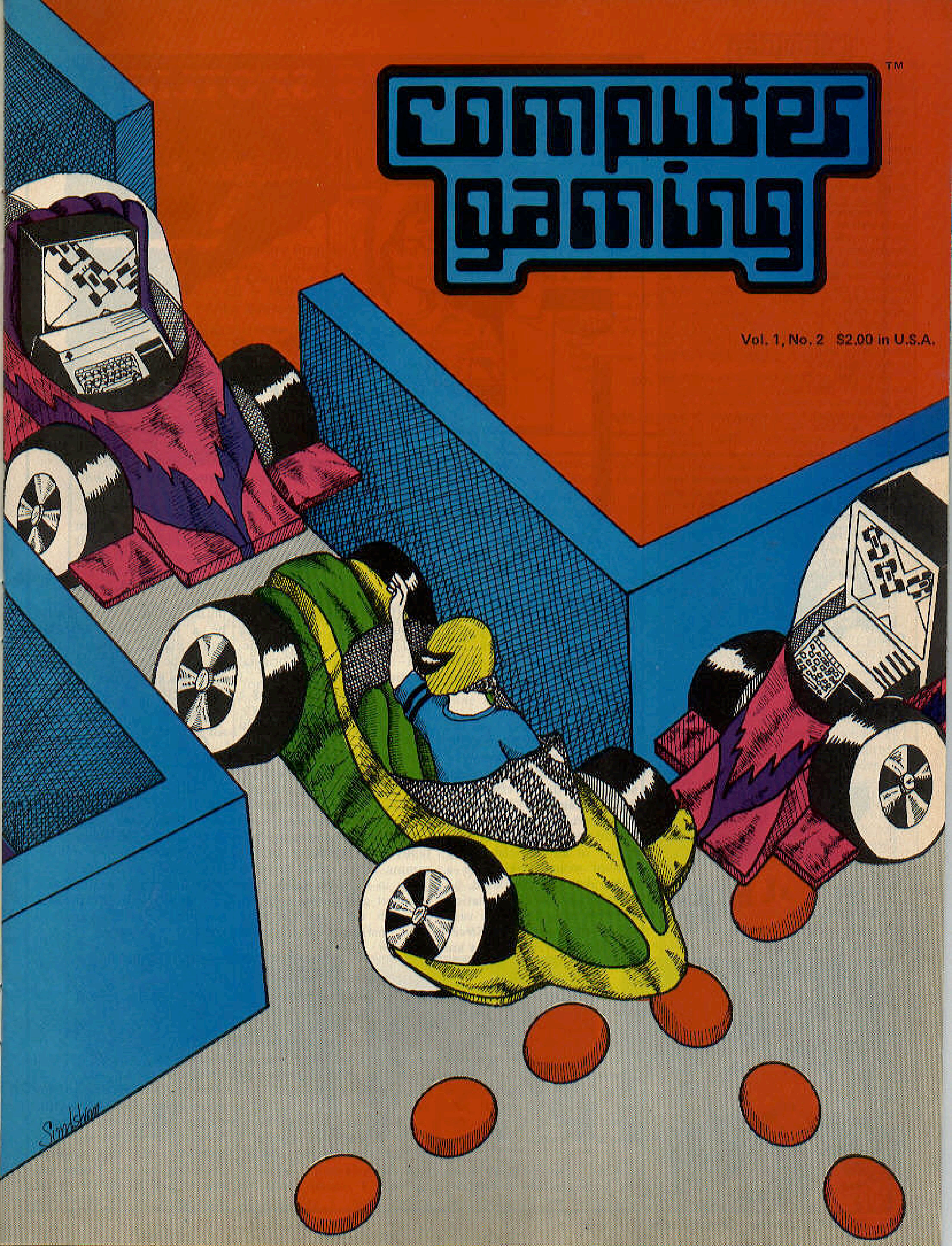
Call for information on availability and ordering. Please have resale Tax Number ready when calling. Sorry. No retail sales—we sell wholesale only to qualified dealers.

LOGIX
A Division of RTA Corp.

PO Box 4107, 991 Broadway, Albany, New York 12204 Call (518) 463-3251 Between 9 AM-5 PM EST
Telex 710 441-8299 OR In The States of VT, RI, PA, NJ, NH, CT, DE, and MA Call (800) 833-3420 Toll Free

computer gaming™

Vol. 1, No. 2 $2.00 in U.S.A.

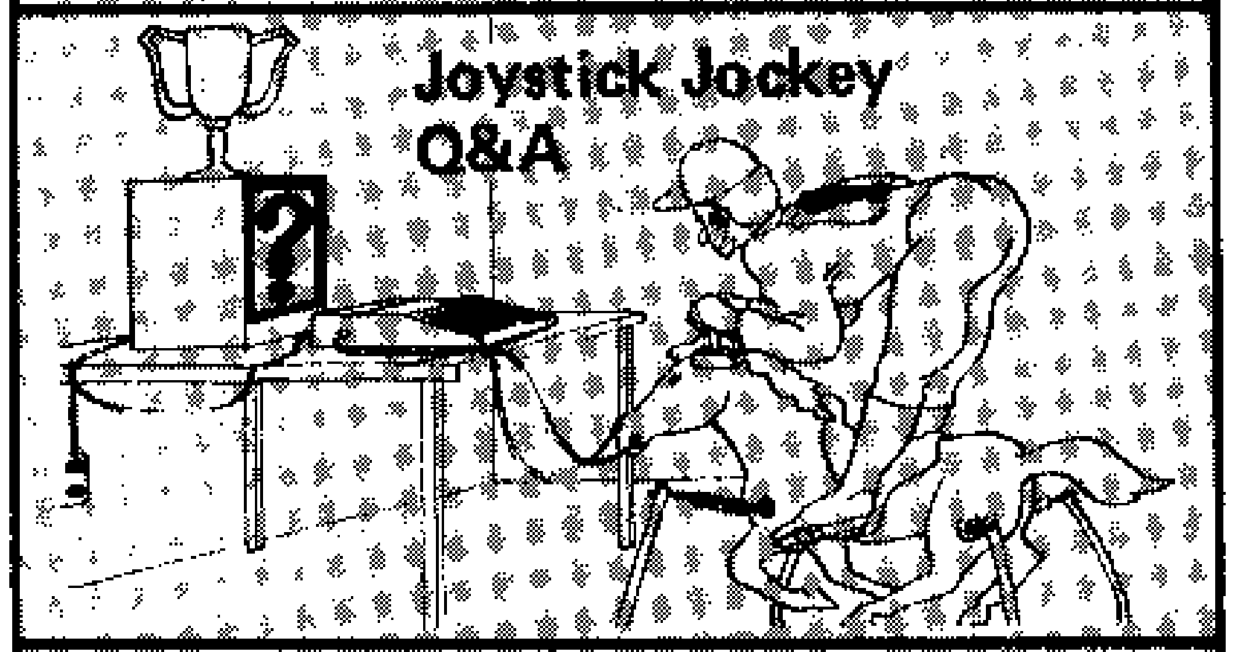


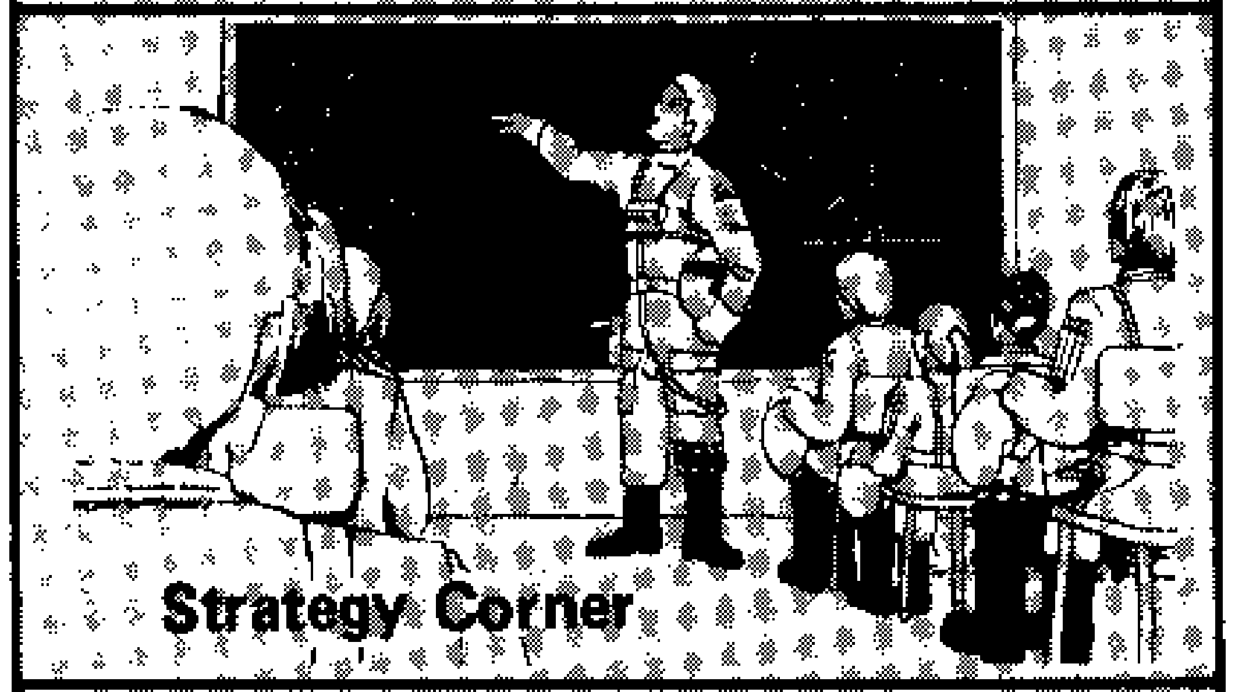
Computer Gaming™

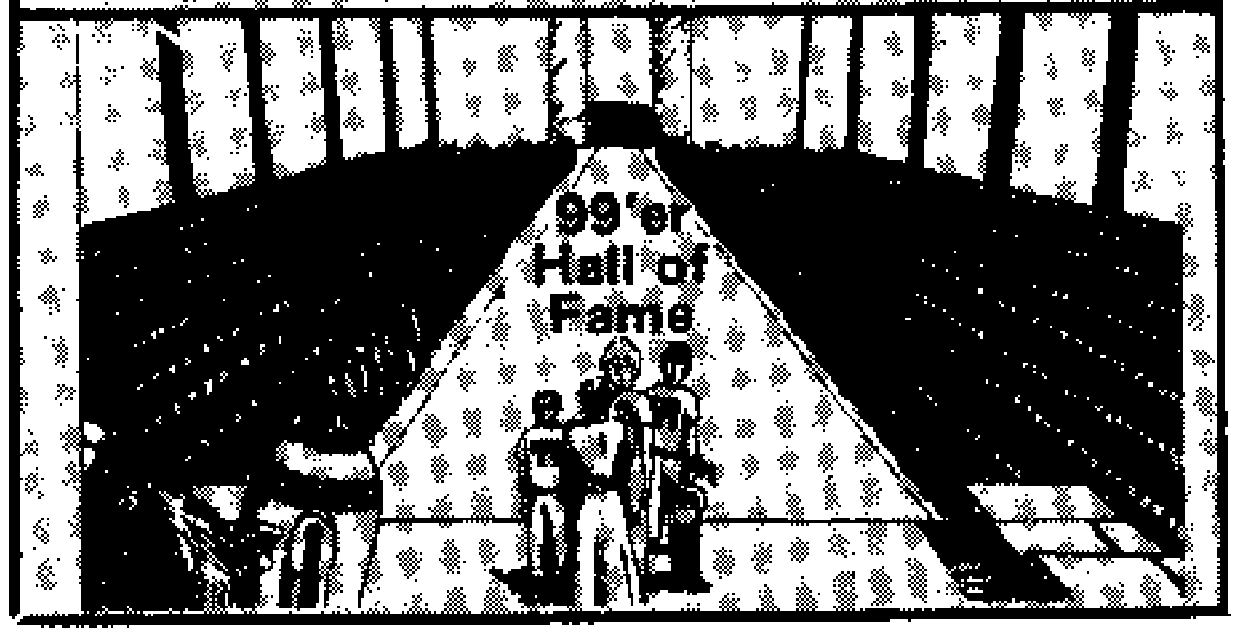
Computer Gaming is a magazine for all game lovers—players, designers, and programmers of microcomputer games. Regular features include product reviews, letters to the editor, player strategy, a question and answer forum, a Hall of Fame for high scorers, tutorial articles on game design and programming, plus interviews with professionals in the world of computer gaming.

All submissions for *Pros on Programming* are governed by the same conditions and payment rate as manuscripts sent to other departments of *99'er Magazine.* Materials submitted for the features shown below are treated the same for Copyright purposes as *Letters to the Editor* in *99'er Magazine* (as explained in the Table of Contents pages); if chosen for publication, the material (except for *99'er Hall of Fame*) will earn for its author a free computer game (either TI or third party) and/or a one-year subscription to this magazine.

99'er Hall of Fame candidates with high scores in TI, third-party, or *Computer Gaming* games must completely describe the conditions under which their scores were achieved (i.e., skill level, keyboard or joystick use, screen number, partner participation, appearance of screen, etc.). Candidates may not be directly related to or affiliated with the programmer of the game or the publishing firm. No compensation will be provided to new inductees whose names are chosen to be immortalized—Fame is its own reward . . .







DESIGNER'S SPOTLIGHT

An Interview With Charles M. Ehninger

Game Designer And Programmer

Future Software

By Gary M. Kaplan

GMK: *You are known for both your gaming and business & professional software. This might seem somewhat incongruous to many other programmers . . .*

CME: Actually, I see the two as being very compatible. I've had many years of simulation and mathematical modeling which is effectively what gaming is all about. When you design a game, you're essentially creating a reality or an artificial reality out of your imagination—a scenario that is simulated. So my background in simulation and modeling comes in very handy there. My *Wall Street* and *SAM Defense* games are obvious examples.

GMK: *Does the fact that you've done a lot of professional programming and business packages hinder your creativity in the gaming area—creating an original game from scratch as opposed to just translating what you have done or seen in the simulation business?*

CME: No, I think quite the opposite. My experience works for me because some of the problems I have encountered in the business world—while not of my design—have required original solutions. For example, I spent one year developing a program to simulate a cryogenic plant. The plant design and the parameters were there, and I had to adhere to them, but the approach to the simulation was entirely left up to me. The creativity it took to develop and optimize that system certainly has relevance in the gaming area. So actually, being in a problem-rich environment and having to come up with solutions has been very instrumental in game creation.

GMK: *We've talked about some of the direct experiences you have had contributing to the design of your games. What about the other areas of your life. . . are you an artistic or creative person in other fields or activities?*

CME: Yes, I think I am. When I was very young, I aspired to become the modern-day Michelangelo; when I realized I could only be second best, I switched to music, where I was going to be a Carlos Montoya. My guitar playing, however, left a lot to be desired, but I did acquire a knowledge of music. And so then I stumbled into computing . . .

GMK: *What was your first game attempt on the TI-99/4?*

CME: My first attempt was *All Star Baseball* written in TI BASIC, of course. The Extended BASIC was not out yet; it had not even been announced. So that game was written entirely in TI BASIC for the 99/4— a machine that intrigued me with its graphics and logic capabilities.

GMK: *What were you concerned with during the preliminary design of the game?*

CME: Well, the idea, of course, was for a baseball simulation that stayed as close to the real rules of the game as possible. And so the design was fairly straightforward. I started with the diamond and the players—centering on the batter and the pitcher. My first problem was what to do about th fielding. I tried various possibilities—finally settling on just switching the scenario completely into either the infield for a ground ball, or to the outfield on a fly ball. So the play alternates between the field and then back to the diamond.

From that experience, I learned that scene switching is very slow in TI BASIC; using the HCHAR to set up your screen is a time-consuming operation. But fortunately, I also discovered a lot of programming techniques that could be used in a BASIC environment. I was not that conversant in BASIC when I started, since I had been programming in other languages.

GMK: *How long did it actually take you, from your initial desire to design and write this game, to actually accomplish it?*

CME: I would say probably at least a month—not full time, however. I was operating under two handicaps: one was that it was my very first experience with the 99/4 and BASIC; but the biggest handicap was that I did not have a printer at the time. The program turned out to be quite long and very difficult to debug by just displaying it on the screen. I actually completed it without the use of a printer.

GMK: *How did you get around the problem of using the slow HCHAR method of putting graphics on the screen.*

CME: I finally settled on just establishing strings and using the display—turning the screen black and turning all the colors black, displaying 24 lines, and then switching on the colors. So that rather than seeing the thing slowly develop, players see the entire screen at once.

GMK: *How long did it take from the time you actually started programming until you finished the coding and were ready to start the debugging process?*

CME: That was probably about 2 weeks . . . It's hard to tell, since I was debugging as I went along. I would establish a new routine, debug it, and go on that way.

GMK: *So what you're saying then, is that you did the program in discrete modules and debugged it as you went along.*

CME: Exactly.

GMK: *When you finally got down to the end of the game, I imagine that you still had a lot of play-testing to do to get rid of any final bugs . . .*

CME: Yes, indeed. That's where I found out that the TI-99/4 BASIC has so many aids to the programmer. I used extensively both the TRACE feature of the BASIC and also the capability to set breakpoints. Take a breakpoint in a given time and you essentially interrogate variables, alter the contents, and then you continue to the next breakpoint. That, plus the TRACE, I found to be very excellent debugging tools.

GMK: *Then the actual play-testing period, where you played the game extensively and did the final polishing, took how long?*

CME: That's difficult to determine, because when I wrote this game it was strictly for my own entertainment. So I played it with friends and relatives and I kept modifying it as things went along. It's really difficult to say how many hours were involved in this.

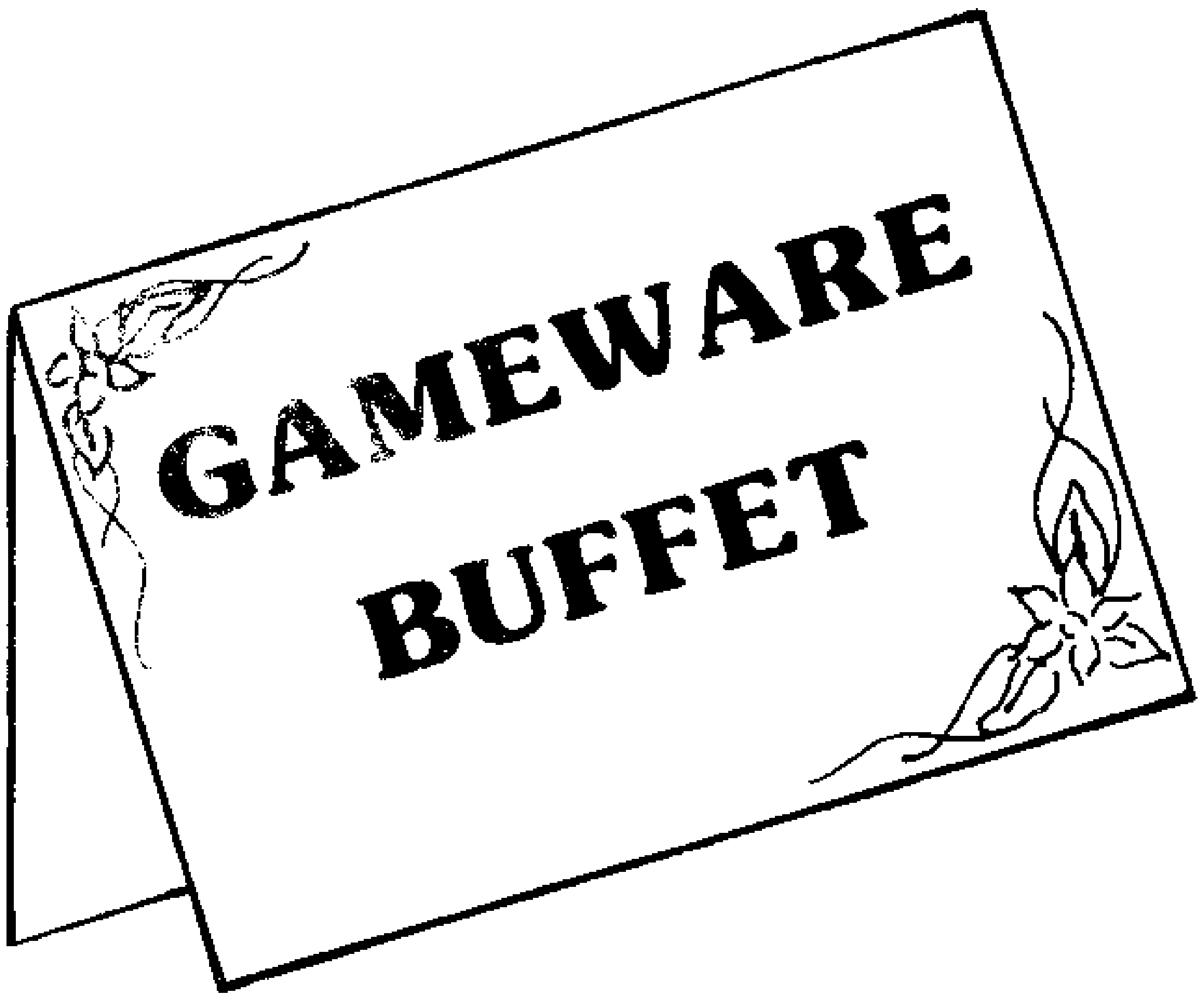
GMK: *Since you didn't intend to originally sell the program, what made you de-*

Continued on p. 44

". . . both countries have a surface-to-air missile defense system and . . . are a secret. If I'm going to divulge any secrets, I'd rather divulge the enemy's and keep the FBI and CIA on my side . . ."

". . . the biggest motivating factor was when I was lucky enough to win the $3000 first prize in TI's author incentive contest. That just gave me the wings to fly . . ."

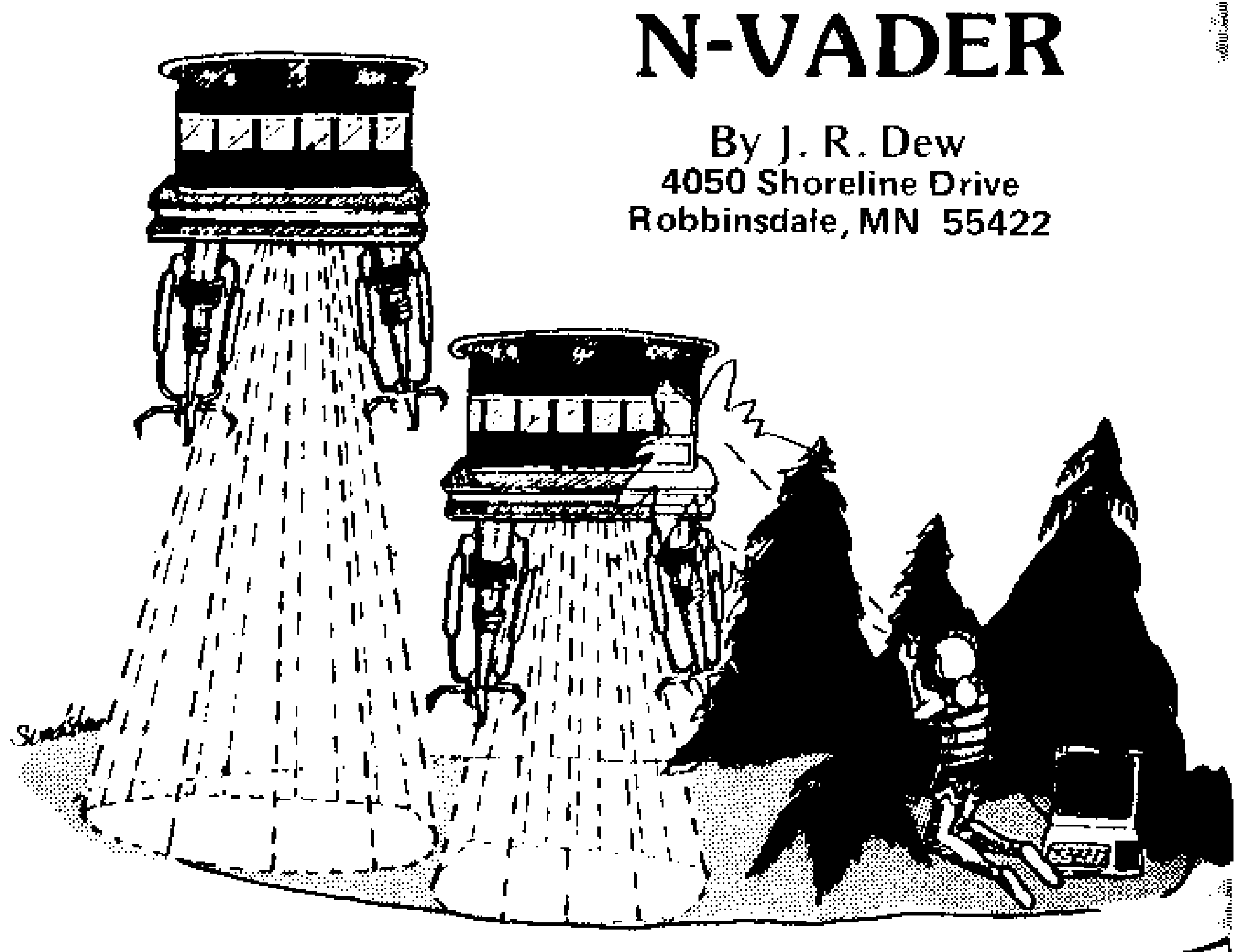
". . . you can't sell vanilla to the people who like strawberry. If they like strawberry, you have to provide strawberry. So we need to provide games for all tastes. "31 Flavor Games"—this is going to be our new motto . . ."



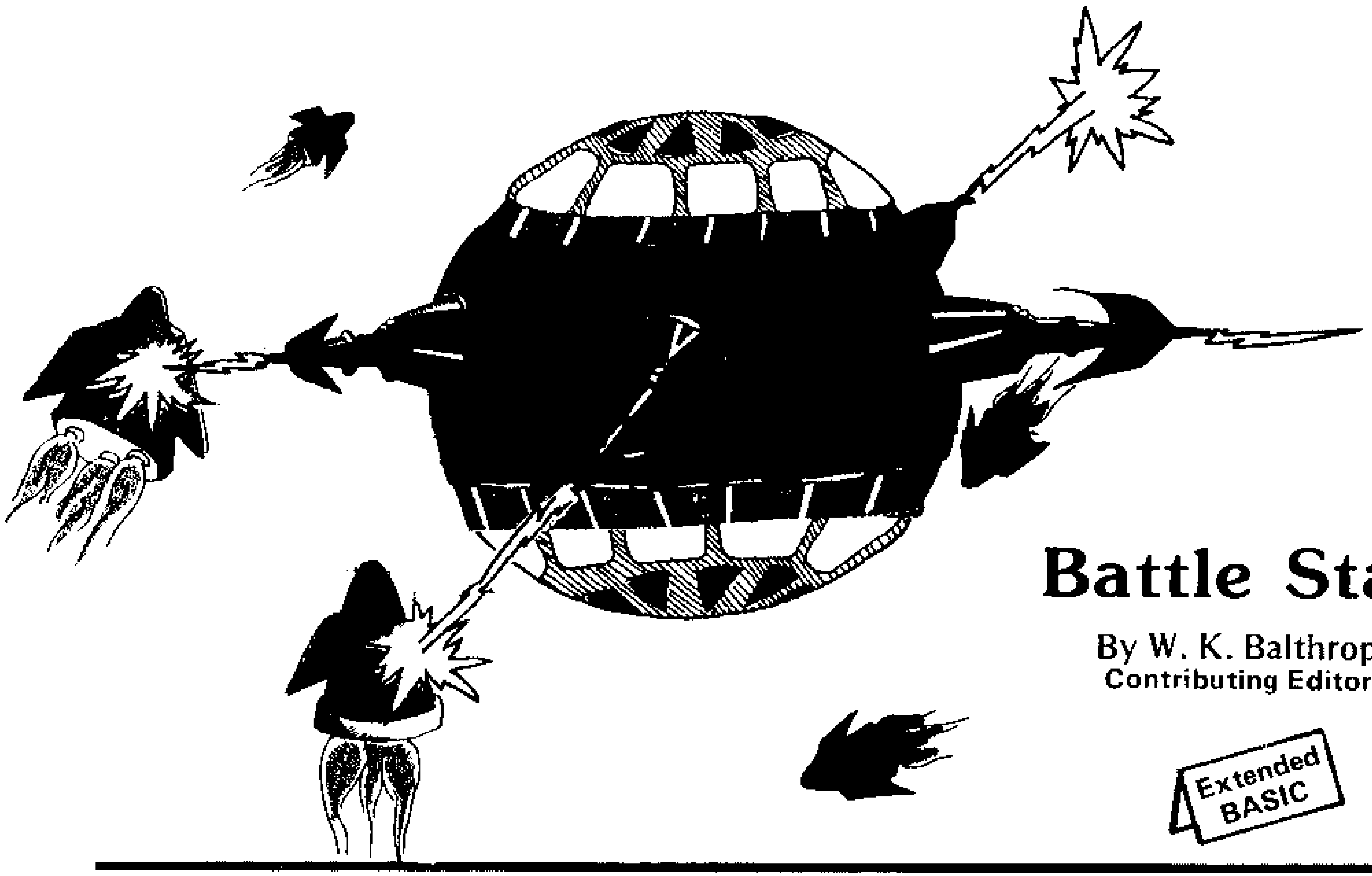
Also See Chuck-A-Luck Part III
In Pros On Programming on page 72.

N-VADER

By J. R. Dew
4050 Shoreline Drive
Robbinsdale, MN 55422



Extended BASIC



Battle Star

By W. K. Balthrop
Contributing Editor

Extended BASIC

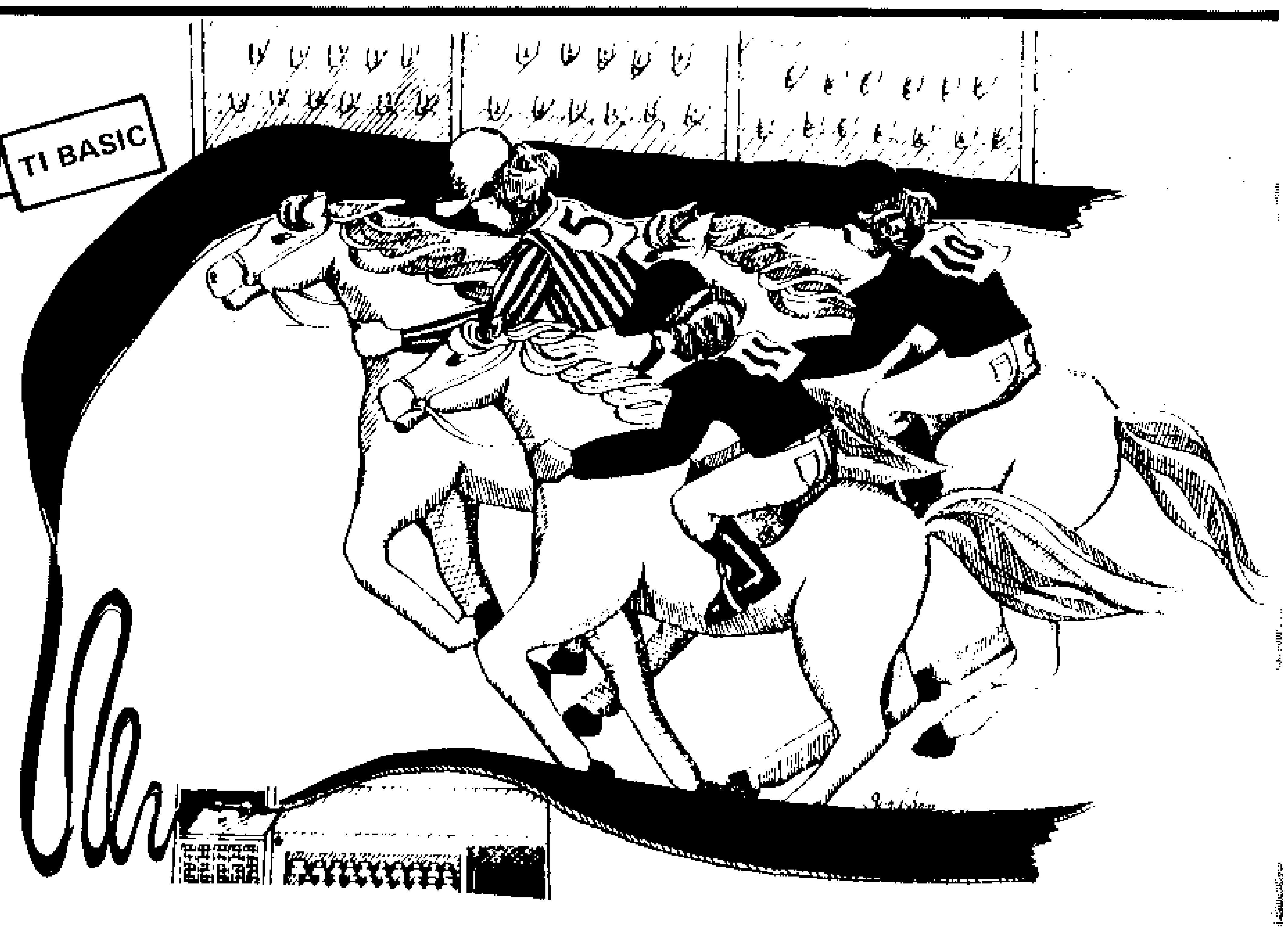
You are the chief security offic in charge of defending the Earth newest Battle Star from ali attack. At first, the aliens are few trying only to probe your weak point Later, they attack in force from all fo directions. It's their somewhat ancie nuclear missles against your las battery. One hit by a missile, howeve and the entire Battle Star is obliterate The speed at which you can react a move your fingers is the only thing th stands between victory and total d struction . . .

To fire a laser in any one of fo directions, press any of the arrow key These are the only keys used. You m not move your Battle Star because your geosyncronous orbit and lar size. The entire game is an eye-ha coordination exercise. At one point the game, the aliens become so fast y may not be able to move fast enou

County Fair Derby

By John Gunter
Route 2, Box 193
Rusk, TX 75785

TI BASIC



County Fair Derby is a party game for up to eight players. Five horses participate in the color-animated race. Our family finds it quite exciting—especially with three or more players. There needs to be only *one* keyboard operator, however, with the rest up to the computer. In addition to running the horses, the computer keeps tabs on each horse's track record, plus the bankroll of each player.

The program operation is simple and self-prompting. To break the input loop, the word LAST must be entered. If this word is misspelled, it then becomes just another player's name.

N-VADER is a game for one or two players written in Extended BASIC. Each player controls a "defense ship" whose mission is to prevent alien creatures from reaching Earth. The game is played using either the keyboard or joysticks. If joysticks are used with the TI-99/4A, be sure to put the ALPHA LOCK key in the *up* position *after* setting the parameters of game play in response to the screen prompts.

Aliens are destroyed by positioning the defense ship in the immediate vicinity of the alien. No fire button or key is needed. Every time an alien is destroyed, the player scores a point and another alien is introduced at the top of the screen. Whenever an alien reaches Earth (bottom of the screen), the aliens score.

One unusual feature of this game is its flexibility. When the game starts, the player(s) can select the number of aliens, their speed, the speed of the defense ship and the defense range. Defense range defines the proximity of a defense ship to an alien in order for the alien to be destroyed.

The game can display instructions, and suggests values for the game parameters. Although the suggested values provide an arcade-type game, you can produce an altogether different (and interesting) type of strategy game by using a small number of aliens and slow speeds.

Features of Extended BASIC Used in N-VADER

Several Extended BASIC features are used to make N-VADER work. Sprites are, of course, fundamental to the program. CALL DISTANCE is used to determine the proximity of alien and defensive ship(s). CALL COINC is used to determine when aliens reach Earth.

Because sprites move independently of the program, alien destruction is sometimes delayed or missed altogether. Aliens can also descend through the Earth for the same reason. Fortunately, these quirks of sprites actually make the game more enjoyable. For example, it is sometimes possible for a defense ship to swoop down *into the Earth* and pick off an alien at the last possible instant!

A subprogram (lines 1390-1510) is used to allow keyboard input to be processed by the main program as joystick input. Programmers with diskette may want to save this subprogram in a MERGE file for inclusion in other programs.

Listings on p. 48

to prevent annihilation. There is, however, an "automatic speed check" put into the program; if you can reach this level and maintain it, the endurance of your fingers will be your only limiting factor. If you wish to make the game even more difficult, you could adjust the limiting speed of the missiles. This is done in line numbers 730, 760, 790 and 820. The X and Y velocity in the sprite being defined (whichever X or Y is not zero) can be adjusted. For example, in line 730 the X velocity formula 11–(L/10). This will allow no speed greater than 10. Change this to 15–(L/10) and the maximum speed will be 14, with the initial speed being 5. If one line is changed, related lines must all be changed.

The Program

The program is very short and simple —requiring only 3K memory and Extended BASIC. There is plenty of room for a good programmer to experiment and try adding to or improving the features. The action is simple, though can become fairly rapid—thus making the game very challenging. A Battle Star is positioned at the center of the screen, and made up of 9 sprites (3 x 3). I did this for dramatic reasons: when the Battle Star is hit, each section of it blows up and flys in a different direction. An alien ship will appear to the left, right, above, and below the Battle Star. At first, only the ship; then later, the ship and a nuclear missile. For every missile knocked out of action, your score will increase by 20 points. For every alien ship destroyed, you will receive 50 points.

The trickiest part of the program was to make the laser rays coming from the Battle Star stop after encountering a missile. Since the missile is a sprite, its location is checked using the CALL POSITION statement. Then, calculating the distance from the Battle Star's cannon and dividing by 8 gives me the distance (in number of characters) of the missile. I then use a CALL HCHAR, or CALL VCHAR—first with the ray bolt, then CHR$(32), a space. Finally, I delete the given sprite. The result is a fast laser bolt and increased program speed.

One problem I encountered when the missiles were traveling at high speed was that they sometimes passed *through* the base without a hit being detected. This problem was alleviated by checking POSITION instead of COINC—i.e., if the position was past the edge of the Battle Star, the missile would blow up.

Listings on p. 49

When this TI BASIC program was loaded into Extended BASIC for the purpose of checking available memory left, the SIZE command revealed that there were 4873 bytes left. This leaves enough memory for you to add to, or modify the program—such as giving the computer a fixed amount of money before the races start and having the players try to "break the track." Other bells and whistles I leave to your imagination. Here's hoping you enjoy the program as much as I enjoyed writing it [and the 99'er staff enjoyed betting on it—Ed.]

But don't waste another minute. It's already post time—the horses will soon be off and running . . .

Listings on p. 48

EXPLANATION OF THE PROGRAM
County Fair Derby

Line Nos.	
140-340	Introduction display and odds table.
350-420	Introductory music and wait for key.
430-1060	Initialization and define characters to be used for display.
1070-1560	Input routines: players' names, choices for: horse selection, kind of bet and amount. Typing LAST for player's name breaks the INPUT loop.
1570-1810	Draws track with lane numbers and plays post-time tune.
1820	Z is a switch to control RETURN from subroutine at 2490.
1830-2020	Positions horses on the track in the proper place and color (subroutine at 2490 draws horse and RETURNs if Z equal 1).
2030-2190	Rests Z; sets starting coordinates for horses (K and S are variables used later in determining win, place and show.) Waits for "S" key to start.
2200-2460	Generates random number from one to five to determine which horse to move. Line number 2220 (ON N GOTO) finds position of horse, sets coordinates for move routine and jumps to move routine. (If the vertical coordinate has been set to zero, the horse has finished and the program jumps back for a new random number.)
2470-2600	Moves horse through an animation loop and redraws it two positions forward from where it started. ("Q" is used as a control switch to pass through the loop twice.)
2610-2830	Check if V >28 (end of race for that horse); if not, sets new coordinate values and jumps back for new random number.
2840-3120	Calculates the finishing horse (D). If S equals 0, the horse wins. Set S equals winning number. Line 2870 (ON S GOSUB) sets color for winning announcement. Line 2990 (ON S GOTO) sets column to zero to remove horse from race; jumps back for a new random number and continues. If S<>0, then the finishing horse becomes K for second place and, (except for setting color) a similar routine is followed. If K<>0, then D becomes the third place horse and the race stops.
3130-3510	Displays win, place, show announcement and waits for key.
3520-3870	On KI(X) goes to the kind of bet player No. (X) made. Checks to see if player (X) has won and calculates the amount. If there are winnings, goes to subroutine 4090. For no winnings, GOSUB 3970. On return goes to 3880.
3880-3960	Increments (X). Checks to see if four results have been displayed; if so, goes to 4130 and waits for key before returning for next results (3550); if not over four, goes direct to 3550.
3970-4010	Subroutine to update and display losers.
4020-4080	Subroutine to update and display losers in debt.
4090-4120	Subroutine to update and display winners.
4130-4160	Wait for key and check for LAST before continuing.
4170-4290	Update past records and display for players betting on trends. Wait for key.
4300-4340	Loop back for INPUTS of next race.
4350-4380	Data for music. Use "break" key to end program.

CROSS COUNTRY CAR RALLY

Reviewed by Steve Schwartz
249 Langton Lane
Bloomingdale, IL 60108

Author:	Larry Norton
Program type:	Arcade "Road Adventure"
Language:	Extended BASIC
Distributor:	Norton Software Box 575 Picton, Ontario CANADA K0K 2T0
Price:	$12.95, cassette

I can say without hesitation that the best game' I've seen so far for the TI-99/4 is *Cross Country Car Rally* from Norton Software (Picton, Ontario). If you have Extended BASIC, this is one fantastic game you won't want to pass up. It has everything you could possibly be looking for in a computer game . . . and much more! This arcade-quality game is relatively easy to learn, challenging to play, visually exciting . . . and it actually seems to become more fascinating the more you play it. Just when you think you've got the game mastered, some surprises are thrown at you that make you feel like you've just loaded it onto your computer for the very first time.

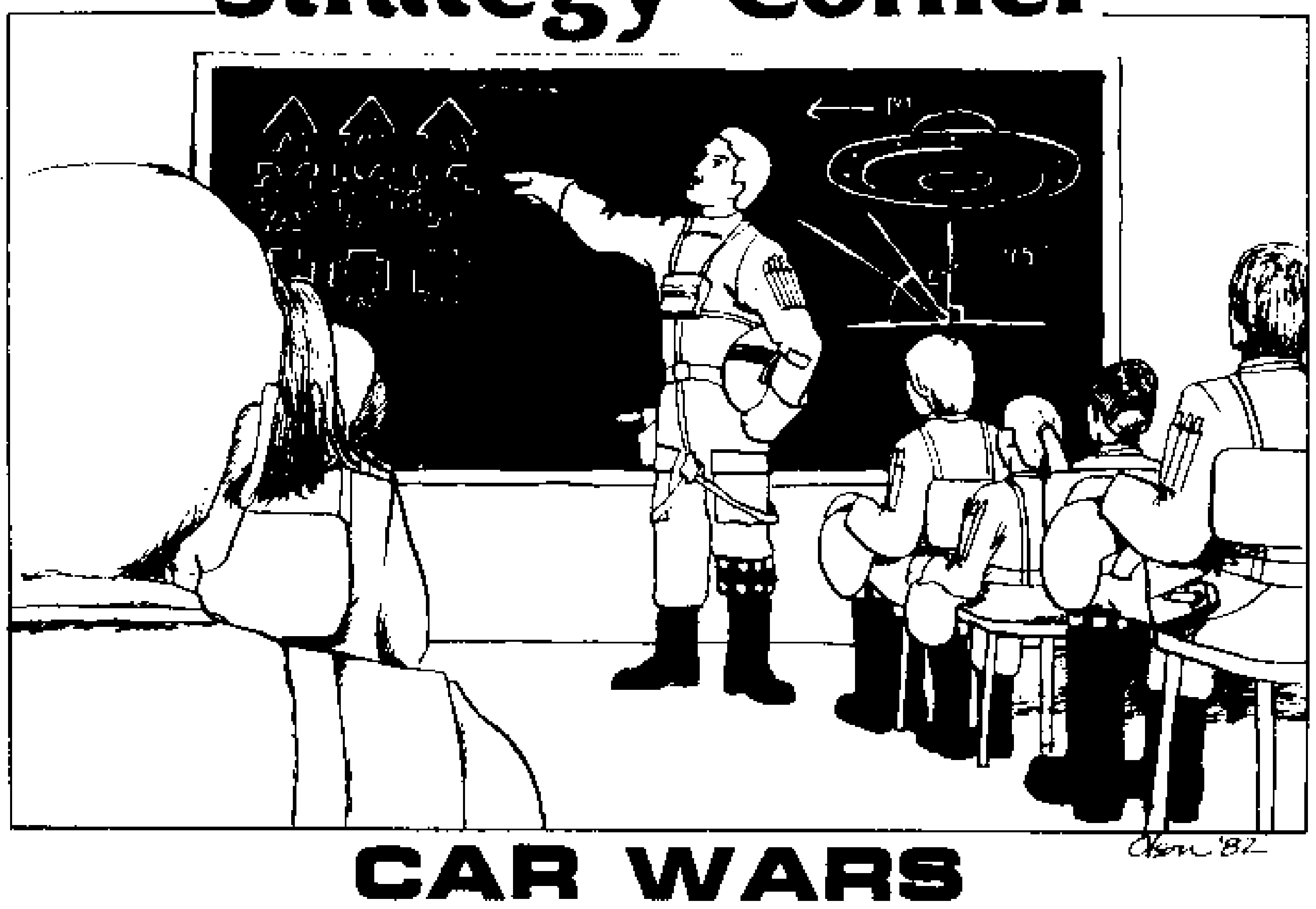
Basically, the scenario is this: You're driving a car from California to New Jersey and you have a limited amount of money to pay for traffic tickets and car repairs. You start off at the left side of your screen and accelerate (keyboard input) until you're flowing with the other traffic. You'll have to do some weaving from lane to lane to get ahead as you slowly inch your way across the screen to the right-hand side (though it seems like you're going much faster due to the passing scenery).

If you're able to make it to the right-hand side—presto! —the screen changes and you're now driving through Nevada *at night*. Utah (if you're able to get there) has different graphics and, presumably, every other state has its own unique graphics layout and its own type of traffic flow. For example, in California and Nevada, you're on a two-lane eastbound highway. But when you get to Utah, there's one lane eastbound and one lane westbound, so you'll have to pass with extreme caution or you'll wind up in a head-on collision for sure. (I'm afraid I can't tell you what the other states are like, because I've never been able to get across the state of Utah; but give me a break!—I've only been at it for a few hours . . .)

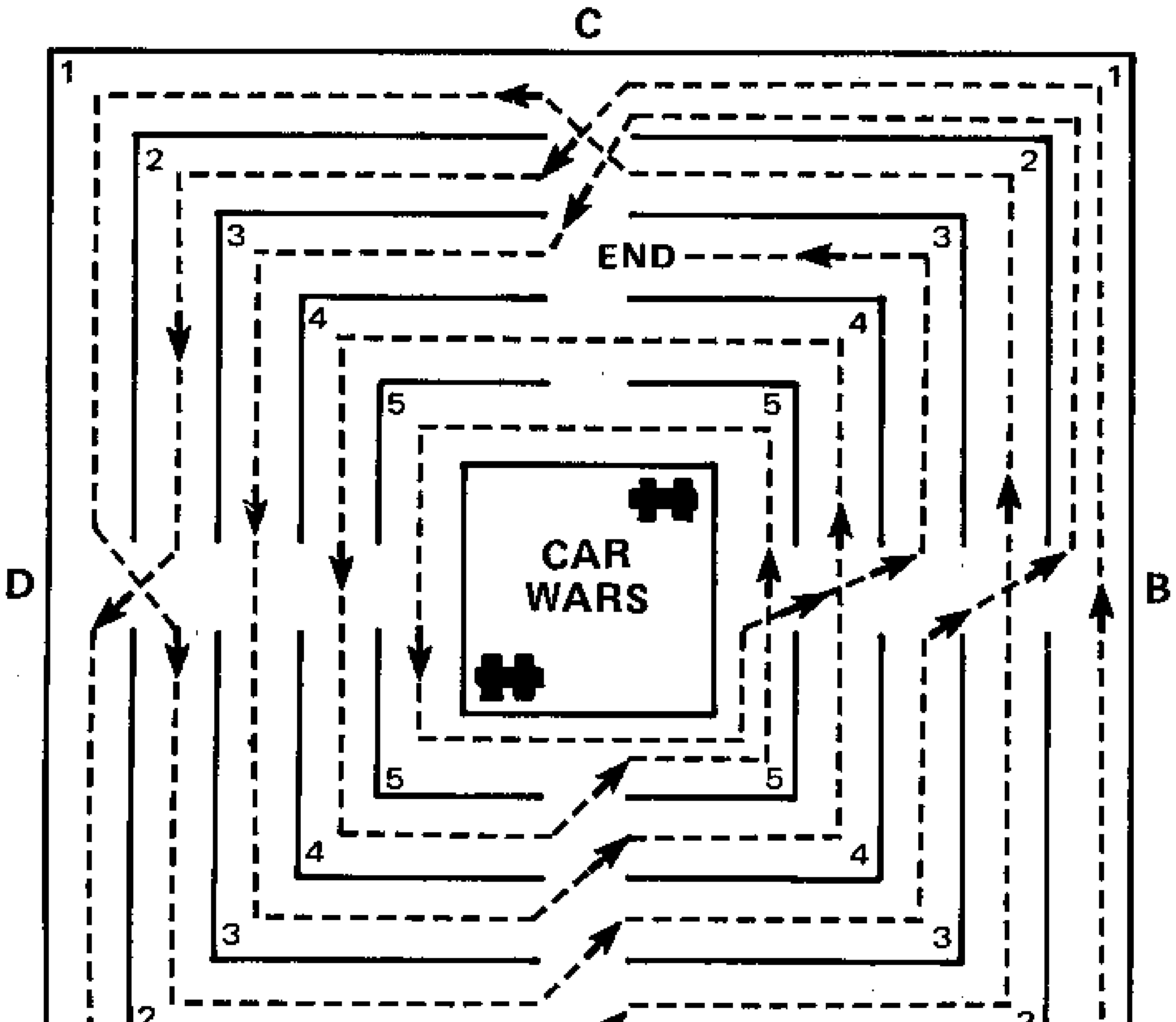
To make things even more interesting, there are detours you have to take, and there are gravel-road "shortcuts" which, as in real life, often turn out to be a *longer* way of getting where you're going. Then there are the police cars that spot you speeding along and warn you to stop: You can try to outrun them (they'll start shooting at you if you don't stop), or you can stop and either pay the fine, or try to bribe the cop.

When you run out of money, the game comes to an end, and you're assigned a point value—ideal for competitive play or for trying to beat your old score. The game loaded from cassette the first time I tried, which was a pleasure in itself. To sum it all up, *Cross Country Car Rally* is the type of game that makes you wonder, "How did they manage to put such a *big* game into a program that runs on a 16K computer?" I don't know how they did it, but I'm very glad they did!

Strategy Corner



CAR WARS



CAR WARS
Strategy by Joe Dyleski, Age 11
Cincinnati, OH
As Reported by Ed York
President of Cin—Day Users' Group

DESCRIPTION: *Car Wars* is a multi-level game that requires good eye-to-hand coordination, and the ability to dynamically plan routes of travel while monitoring and avoiding the adversary vehicle(s). Your objective is to go from start position and score points by clearing the lanes of their dots. Beware: You are not alone! At screen one, another car, guided by the computer, starts from the same point but in the opposite direction. Its only purpose is to crash into *you*. At screen two, the starting position of the computer car changes, thus increasing the challenge. By screen three, the demands on your alertness have doubled—there are now *two* computer cars seeking to crash into you. Once you have mastered screen three—easier said than done—the fourth screen sets those two computer cars against you from *totally different* and *random* starting points. And if this isn't enough, take heart—the next screen has *three* cars awaiting you!

STRATEGY: Joe's strategy for screen one is based upon two elements: 1) planned lane changes, and 2) speed adjustments in order to move your car to selected positions by the time the computer car has arrived at predictable positions. With this in mind, the following approach and diagram should have you clearing all the dots and ready for the bonus points.

1) Starting Point: A, Center, Lane 1. Your car's direction is toward side B. Computer car's direction is toward side D.
2) Accelerate your car to about 1½ times faster than computer car in order to arrive at opening on side C when computer car comes to opening at side D.
3) Switch to lane 2, proceed to side D.
4) Switch to lane 1, proceed to side A.
5) Switch to lane 2, proceed to side C.
6) Switch to lane 1, proceed to side D.
7) Switch to lane 2, proceed to side A.
*You have now cleared lanes 1 and 2 of all dots.
8) Switch to lane 3, proceed to side B.
9) Switch to lane 1, proceed to side C.
10) Switch to lane 3, proceed to side A.
11) Switch to lane 4, proceed around lane 4 back to side A.
12) Switch to lane 5, proceed around lane 5 back to side A.
*You have now cleared lanes 4 and 5 of all dots..
13) Continue in lane 5 to side B.
14) Switch to lane 3, proceed to side C, END.
All dots have been cleared.

By Gary M Kaplan

Just the sound of the name Walt Disney conjures up images of all those fantastic animated movie classics spanning over a quarter century of entertainment for young and old alike. But this summer, the celluloid magic of the Disney Studios has taken on a new dimension with the release of their eagerly awaited science-fantasy, *TRON*—an incredible computer graphics extravaganza in which fantastic vistas of texture and light are generated artificially by computers. As movie-goers worldwide continue to be awed by *TRON's* video warriors and computer programs fighting for survival in an electric universe, a new awareness of computers—and in particular, the mind-boggling possibilities of computer-generated imagery—permeates the consumer cosmos. With one wave of Disney's digital wand, the glass of Cinderella's slipper has been magically transformed into the cathode ray tube of a video monitor.

This heightened awareness is the death knell for manufacturers of consumer computers who do not provide sophisticated color graphics and animation capabilities. Fueled by *TRON* (and the horde of video clones that are destined to follow), the public's demand of, and expectation for more visually spectacular video games and educational displays will surely take quantum leaps. How can computer manufacturers and software houses ever hope to satisfy this demand? That's one tough technical question that some of the finest design teams in the world are currently tackling. One thing is obvious, though—more and more special effects that are usually implemented through *software* must instead be "integrated" in the *hardware*. This means more powerful, and easier-to-control VDP (video display processor) chips—the silicon workhorses responsible for the displays.

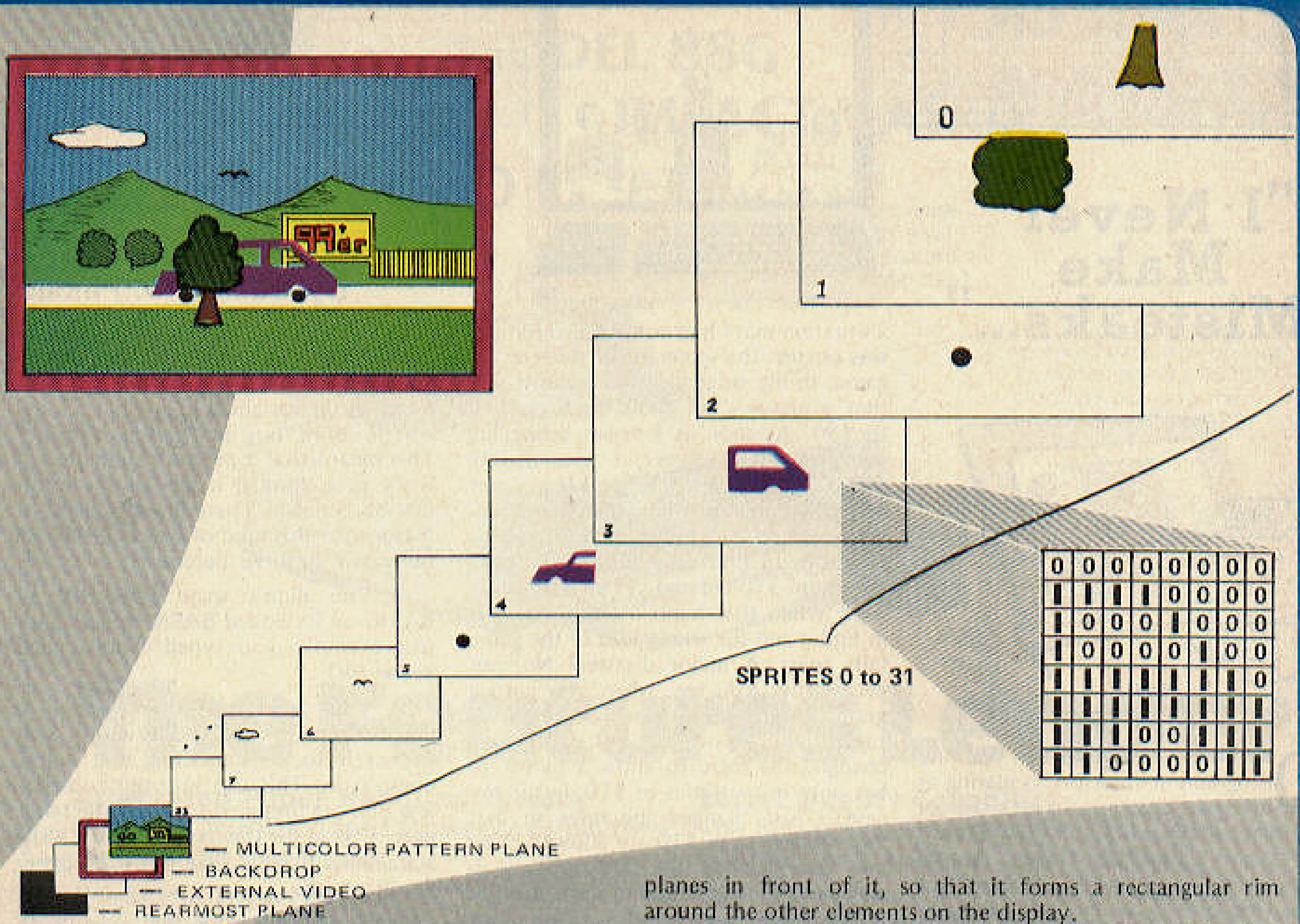
The easier-to-control requirement doesn't necessarily mean easier for highly-trained, professional *programmers* to control. There will have to be a way for people such as artists and "graphic gurus" with fantastic imaginations to interact directly with the display system—a way that requires only a bare minimum of "programming" experience to implement sophisticated visual effects.

To anyone familiar with the interactive graphics capability of the Texas Instruments 99/4A Home Computer, it is obvious that TI has already made great strides toward this design goal—great enough, in fact, to cause at least two other well-known computer manufacturers to attempt to emulate TI with their "newly-discovered," smoothly moving graphic patterns now known universally as *sprites*. Color sprites as implemented on the TI-99/4A, however, have yet to be equaled in their versatility and ease of use in a multi-language environment (Extended BASIC, TI LOGO, UCSD Pascal, 9900 Assembly Language and TI Pilot).

A Flat, Yet 3-D Sandwich

The wonder VDP chip behind sprites and other video effects that the 99/4A is capable of producing is called the TMS9918A. This complex LSI (large-scale integrated) chip represents the next generation beyond the many small- and medium-scale integrated circuits that formerly had to be assembled to achieve a display with a minimal level resolution required for video games. But the consolidation of many-into-one wouldn't merit an entire article here if it wasn't for the chip's novel approach to dramatically simulating a 3-dimensional animated graphic display: It does this by creating nearly three dozen flat, "stacked" geometric planes that are sandwiched one on top of the other onto the picture tube of your TV or color monitor.

On *each* of the first 32 planes (numbered 0 to 31), we can define the image of *one* sprite, give it one of the 15 standard colors (the 16th is transparent), and then set it in motion quickly and smoothly. We do *not* have to redefine the imagery over the screen to simulate motion, because once set in motion, a sprite can continue to move without further program control. When a sprite of a *lower* numbered plane (closer in the foreground) comes into contact with another sprite on a *higher* numbered plane, it progressively blots it out and creates the illusion of passing in front of it.



0	0	0	0	0	0	0	0
1	1	1	1	0	0	0	0
1	0	0	0	1	0	0	0
1	0	0	0	0	1	0	0
1	1	1	1	1	1	1	0
1	1	1	1	1	1	1	1
1	1	1	0	0	1	1	1
1	1	0	0	0	0	1	1

For example, in the figure shown here, the moving car that is composed of four sprites set in motion together on plane numbers 2-5 will pass *behind* the stationary tree (composed of 2 sprites on plane numbers 0 and 1), and *in front of* the billboard which is drawn on the plane immediately behind the rear-most (number 31) sprite plane. By the same design rules, the cloud (plane 7) will mask the color of the sky behind it, and a bird (plane 6) will both mask the sky behind, and appear to fly *in front of* the cloud. And since sprites move in a transparent surrounding, the scenery in the background behind the car may be seen *through* the "windows" of the moving vehicle! The entire scene has the appearance of depth and simulates a 3-D animated color movie.

The Multicolor or Pattern Plane is used for textual and fixed-graphics images. It is this plane (containing the sky, mountains, bushes, billboard, fence, roadway, and grass) that the sprites on the remaining 32 planes appear to pass directly in front of.

Immediately behind the Multicolor Plane is the Backdrop Plane—solid colored and slightly larger than the other 33 planes in front of it, so that it forms a rectangular rim around the other elements on the display.

The rearmost plane is pure black, so that when the other planes are set to transparent, the screen appears to be black. Although there is no provision in the current version of the TI-99/4A for *simultaneous* on-screen mixing of external video with computer-generated graphics (e.g., sprites or fixed graphics mixed with input from a video cassette recorder or video disk player), the TMS9918A chip *can*, in fact, accommodate external video; this video would be displayed on the rearmost plane with part or all of it masked by computer-generated graphics until needed (e.g., as subtitles for the deaf or foreign language translation, or perhaps a "real-life" video-taped space movie scene viewed through the scanner screen of a computer-generated starship command center). Add to this the capability of chaining together *multiple* 9918A chips, and you have the potential for a visual gaming or educational environment (in future versions of the TI Home Computer) that is simply mind-boggling!

Those Magical Sprites

When sprites are on the screen, the 9918A chip organizes the display into a high-resolution pattern of 256 by 192 little boxes or picture elements called "pixels"—the smallest controllable elements on the display. Each one of these 49,052 pixels represents a possible address for a sprite to reside at, or pass through when moving across the screen.

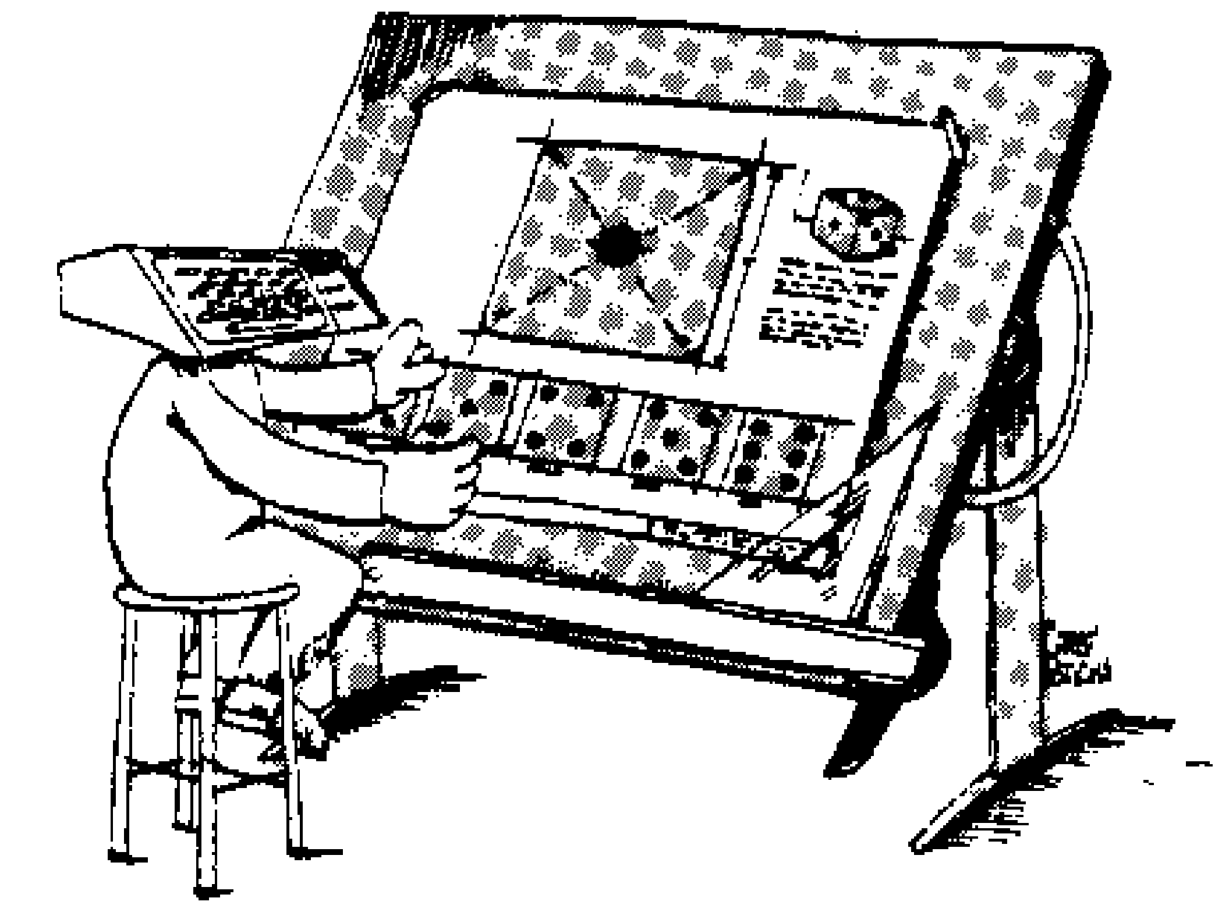
Continued on p. 58

"I Never Make Misteaks..."

CHUCK-A-LUCK

PART 3

By Samuel D. Pincus



Don't laugh. All too often, you find programs with errors as glaring as my first sentence. So let's correct it: I never make mistakes! Now, doesn't that sound egotistical? Nobody would have the nerve to say it out loud. But some people who write programs act like they never make a mistake while programming! The best programmers that I have ever met not only admit that they do make coding errors, but they also have developed ways to find these inevitable mistakes quickly and efficiently. They do this by setting up a "debugging" plan. What's a "bug," you ask? That's what the programming fraternity calls anything in a program that doesn't work correctly. And like anything else, we need a good plan to catch them. Yup, we still have planning to do on our Chuck-A-Luck game.

First refresh your memory about our game. In the last issue, we wrote a large percentage of the code required to play the full game. As a matter of fact, the only important module not coded was the graphics routine. So obviously, it's time to bring on the bugs! WHOA! First we have to figure out how to test for the various bugs I KNOW are in there. Before we do this, let's stop and talk about the different type of bugs found infesting even the best programs.

The first bug that must be eradicated is the "Baddus Plannus." This bug hits programs that do everything (according to the design) correctly, but don't achieve the desired result or implement all the rules that you originally laid out. For example, as soon as I began testing my original code for Chuck-A-Luck, I hit a situation that I had not planned for and was outside the scope of the rules of the game. In my original list of rules, I said that a player's bet could be from $10 to $50. As soon as I began debugging my program, however, I immediately saw a flaw in the whole idea! If a player bets other than $10 units, he may eventually wind up with less than $10 in his bankroll. In that case, he can't make a minimum $10 bet and yet he isn't bankrupt. When that happens, the player is in limbo and the whole idea of the game falls apart. A major disaster? No, not necessarily. You see, when you have a good design, these kinds of problems can usually be overcome. I could have changed the logic to allow a player to bet only in multiples of $10, but in my case, I just changed the rules so that bets of less than $10 are allowed. You may have noticed that this change is already in the code found in the last issue.

Note that I am not ashamed to admit this error. Instead, I expect something of the sort to happen whenever I write a program. So when I set up my debugging plan, the first few items on my list are tests of the rules. These items don't have to be the first things actually tested, but they must be tested by the time we finish debugging.

The second bug that creeps inside programs is the very evil "Baddus Designus." This guy shows up when the code almost does what you want. A sure sign that your program has this problem is that it doesn't do everything that you wanted it to. It may mean that you left out some modules needed to get the program running correctly. It could also mean that a piece of code needs more information (or variables) to do its job. In other words, you forgot (or missed) some facts when you were designing your code. This kind of bug is uncovered by making sure that each routine is thoroughly tested, and also by ensuring that each routine is tested using different values in the variables.

The third bug is "Baddus Codus." This means that a piece of code doesn't work even though it has all the information it needs. There are a number of reasons for this kind of bug, but they all boil down to three major ones:

1. You didn't write code that TI BASIC or Extended BASIC understands (for example—you typed in misspelled keywords).

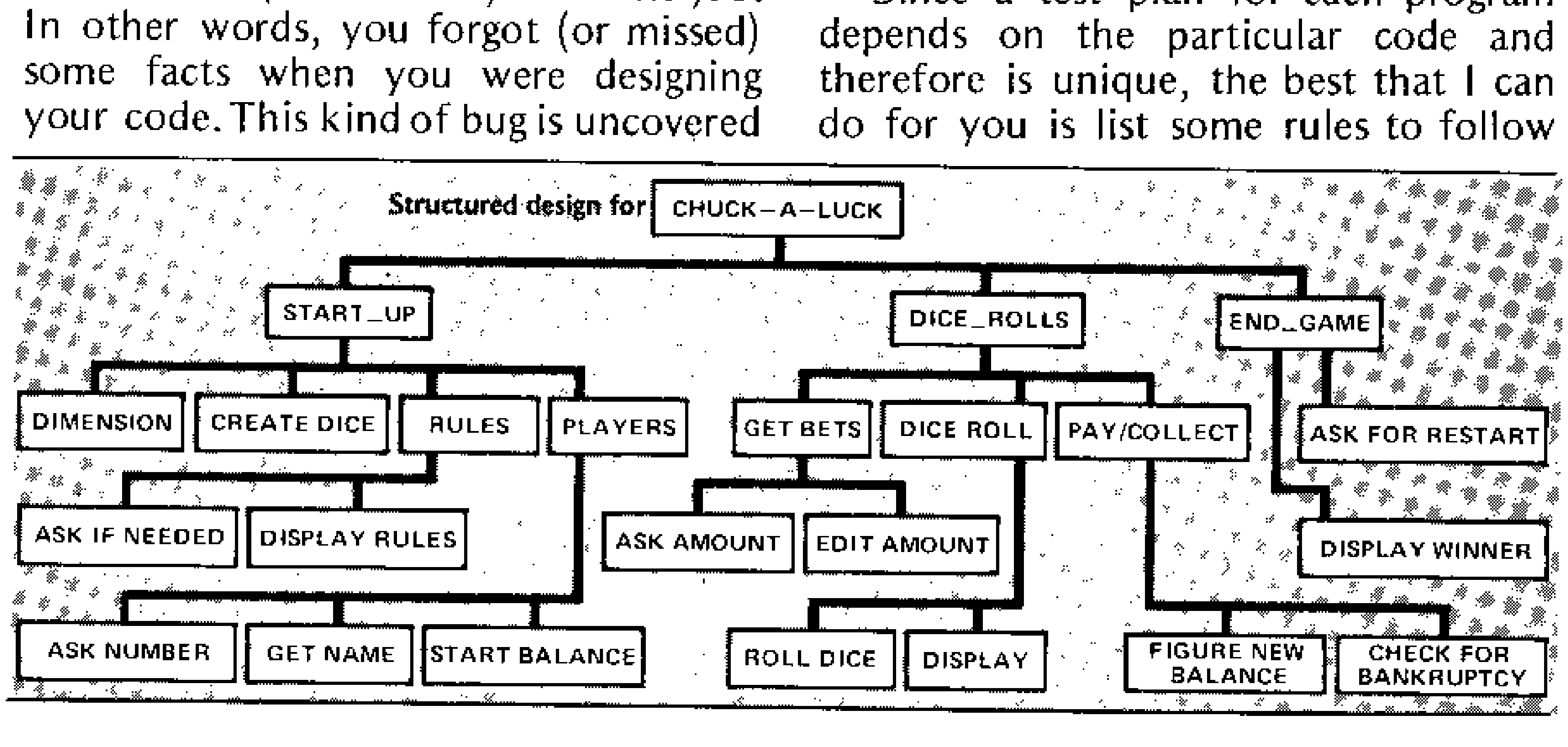
2. You don't really know how a particular feature of BASIC works. You expect it to do something that it just doesn't do. This can hit your code unless you are prepared to check the reference manual for the usage of any BASIC statements that you are not thoroughly familiar with.

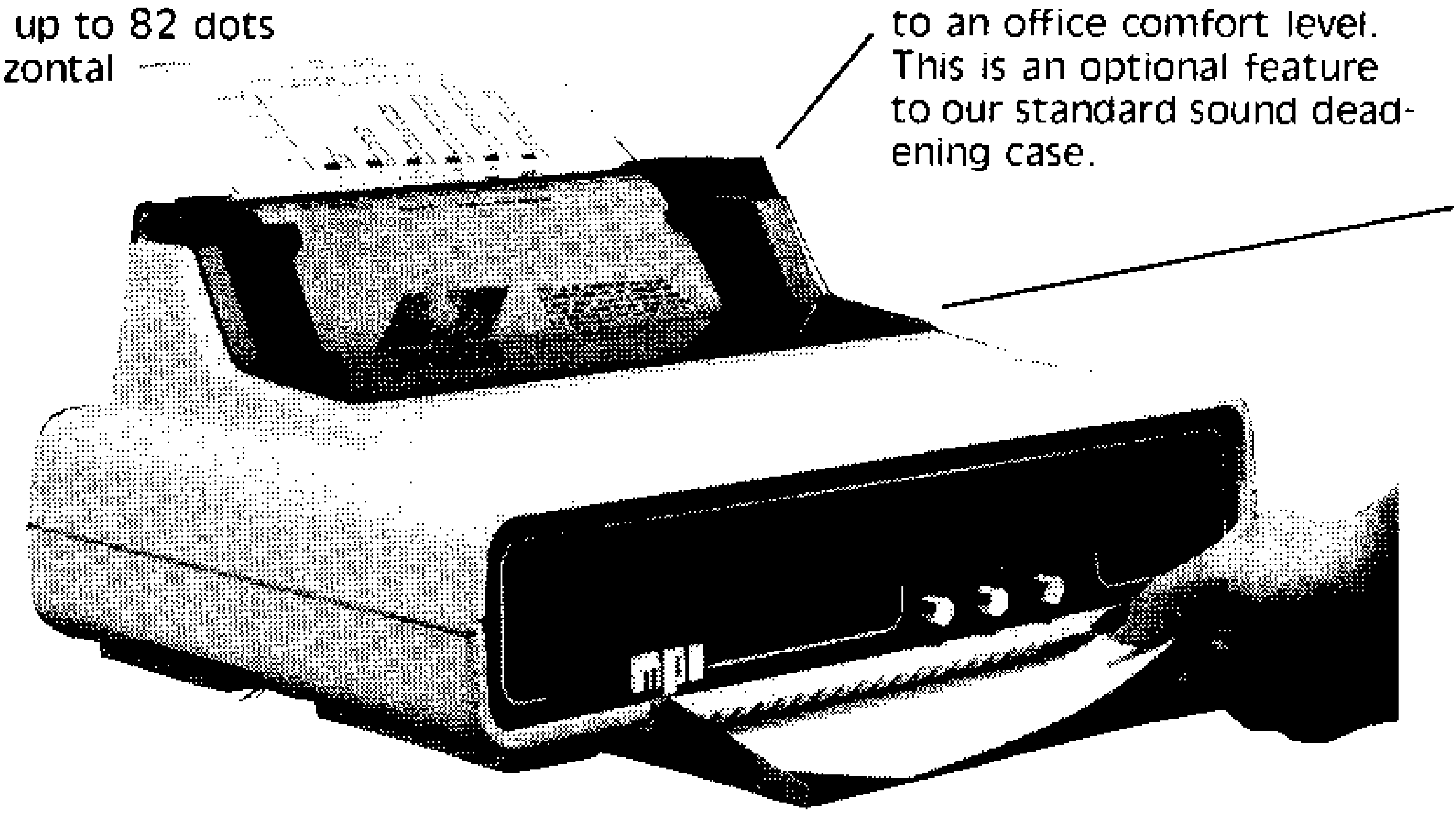
3. You wrote code that doesn't do the job. The code may be in the wrong sequence (i.e.—you are zeroing out a number just before printing it out on the screen), or a piece of a code line is missing, or you typed in the wrong variable name, or even keyed in the wrong letter, etc. It all boils down to normal human error.

Bug Catching

If you are lucky, TI BASIC or Extended BASIC will catch some of your errors for you. But don't rely on it. The only good way to check for a case of "Baddus Codus" is to look over your code before running it and then carefully watch how your program behaves when you run it.

Since a test plan for each program depends on the particular code and therefore is unique, the best that I can do for you is list some rules to follow





when making up your test plan and debugging your programs.

A. List the program and visually check the code. Review your code looking for incorrect spelling of variables, miscoded statements (i.e., missing double colons between statements in Extended BASIC), and incorrect CALL names. Fix any errors you find immediately. After you have done this, do it again. Then save this copy of your program to disk or tape before you run the program. This will protect your hard-earned code if your computer decides to "eat" your program on the very first test. Label this version as Version 1.

B. Write down the function of each major module. Under each module, list the range of valid variables. This should be done so that when you begin debugging, you can set up your tests using both the largest and smallest values possible for each module.

C. Set up a test for each major module. Write down what values you will input and what you expect the output values to be. If you don't write it down before you begin your test, you won't really know if a module is working correctly while you are debugging.

D. Decide whether or not you can use the BREAK command to test the module. In many cases, a routine or module can be tested locally. By that, I mean that the module uses only a few variables and that you can set some values for these variables at the start of the module and BREAK at the end. Then you can check to make sure that the results are correct by PRINTing them on the screen when the computer stops at the BREAK point. For example, suppose a routine starts at line 1000 and uses the variables X and Y as input. The routine is supposed to use these values to calculate the variable Z using some formula. You can test this routine locally by adding the following code in the front and back:

```
1000 BREAK (replaces the REM
statement at the start of the routine)
 . . . . routine
 . . . . code is . . . . .
 . . . .
 . . . . . here
1100 BREAK
```

Now RUN your program and make X and Y whatever values you want them to be when the program initially stops at line 1000. When you type in CON, the machine will execute your routine and stop at line 1100. Just wait for the second BREAK at statement 1100 to be hit and when your program stops, type in PRINT Z so you can look at Z's value. In fact, you may want to initially add a program statement after the second BREAK that says something like 1110 GOTO 1000. In this way, the routine will continually repeat so that you can test your code using a number of input variables without the trouble of having to execute the rest of the program each time. That's why I call it a "local" test.

Just make sure you remove that extra GOTO statement as soon as you finish testing that module! Of course, this technique isn't possible with all routines, and in some cases, it's not worth the effort. Just keep in mind that it's one debugging tool that you can use. By the way, it shows one reason why it is good to get into the habit of writing your code in a very straightforward way. Try to minimize the amount of code in a routine that GOTO statements located outside the routine. Just imagine if there were 4 or 5 statements between lines 1000 and 1100 that jumped outside the routine. It would be almost impossible to locally test the routine because you could never really be sure that your program would get to statement 1100. That's why I try to code all my routines with only one entry point and one exit point. Sometimes it just isn't possible to do it this way because program size limitations force me to reuse code for different reasons, but wherever I can do it, I make sure to write my routines this way. It may cause a little extra effort when first coding, but it makes the debugging a lot easier.

E. Begin your tests. Carefully note any time that a routine did not give a correct result. Don't stop the program (using the Shift C or FCTN4 key) each time you notice a problem, just note the nature of the problem, and what the program was doing at the time. For example, if you notice a problem in a routine only when the second player is betting, or if the dice roll was a 6, this is very important information and you should make sure that you write it down. Wait until you have uncovered a number of problems or until the computer stops with a BASIC error message.

F. Check each routine where an error occurred. Mentally "walk" through the code by doing each instruction or calculation on a piece of paper. Usually, you will find your errors this way quite easily. When you locate the error, write down the line number and the solution BUT DON'T KEY IT IN! This is because as soon as you change any of the code in a program statement, BASIC will reset all of the variables to 0 (for numbers) or empty (for strings). This may make it impossible to debug some other routine during this same test run. If you cannot find the bug by walking through the code, look at any intermediate results that may be available by PRINTing any intermediate variables. You may be able to find your mistake this way. It works especially well in complex code with a lot of intermediate totals.

G. If you get to a very difficult spot where the code looks OK but you are sure contains an error, don't panic! Begin to use the BREAK xxx (where xxx is an actual program statement line number in your routine) command. This allows you to stop the program every two or three lines. Each time the BREAK is hit, PRINT any variables that you think are important. Write them down on a piece of paper along with the line number that you stopped at. Then type in another BREAK xxx command for a line number two or three lines further along in your routine. Type in CON and wait for the program to stop again. You can usually narrow down the problem to a single line this way. Once you get to the offending line, read it over carefully. It probably has a misspelling or other typographical type of error. If you still cannot find it, then re-enter the program line very carefully when you have finished this round of debugging. This will likely fix the error (as long as the code you are entering is good code).

H. When you have gone as far as you can in this test, fix all the bugs that you have discovered. Check off any of the tests that have successfully been concluded.

I. Save this new version of your program to disk or tape. I usually have a version number in a REM statement in the front of my programs. I increase this version number every time I change my code. This allows me to know what version of the program I have read into the machine when I begin my tests the next day. If you are saving to cassette tape, make sure you label the tape with the new version number. If you are using a disk, you may want to add the version number as part of the program name (i.e. SAVE DSK1.CHUCKV3). Making the version number part of your SAVE routine can save you some agonizing problems. There is nothing worse than realizing that you are debugging the same code that you fixed the day before.

J. As your program runs, review its actions against the rules and requirements that you originally set up when you began your plan. See if the results are what you expected. If they aren't, immediately stop testing and try to figure out why. You may have to change the rules. You may even have to redesign part of your program. It isn't worth testing any more until you fix this kind of problem.

K. If you get an idea to improve your program, write it down. Don't stop testing to make minor improvements. You may overlook a major flaw while adding a small feature. Add any of these improvements at one time and revise your test plan to retest the old code as well as test the changes.

Someone once asked me if there is an easy way to debug programs because he seemed to spend an awful lot of time at it. After showing him the REM header of a version (VERSION 56!!!) of my latest program, I told him that most program development time was spent on testing and debugging. A professional programming department usually knows that a programmer codes *three* lines of code for every one line that eventually finds its way into the final version of a program. In other words, two-thirds of the programming effort is spent getting the code debugged. Although a good design and good coding habits can make it easier (and drop that ratio down to 50 percent), debugging is frequently a slow task--a task, however, that *must* be done.

After my initial debugging, I began to add some of the modules that I left out the first time. The first routine I added checked to see if the game is over. This feature was added in lines 750,770-890, and 5000-5400. I do this by checking each player's cash balance. If a player has a balance greater than zero, I increase a counter which tells me how many players are still in the game. I also save that player's number. That way, if only one player is left at the end of a round, I know who it is. If the game is over, I check to see if a replay is wanted. I also added the code at 21100-21500 which displayed the rules. I then retested the program to check both that the new modules worked, and that they did not cause any damage to the old code in the rest of the program.

In the next issue of *99'er Magazine*, I'll explain how I added the graphics for both the TI BASIC and Extended BASIC versions. For now, you can study and type in the complete TI BASIC game listing that follows. 99'er

Listings begin on p. 59

*

*Ad placed this way on advertiser request.

Brader's TIPS:

TINY TUTORIALS & TIMELY TROUBLESHOOTING FOR YOUR TRIALS & TRIBULATIONS

In this column, David Brader answers questions on any area of TI-99/4(A) computing. The most representative questions received will be answered and printed in this column. Do you have a question? Send it to:

BRADER's TIps
99'er Magazine, P. O. Box 5537, Eugene, OR 97405

Can I make my own Command Modules?

With the new TI Mini-Memory you can make your own Command Module. By using the Line-by-Line Assembler supplied on cassette tape with it, you may even write the software in assembly language for super speed. After you have completed the software, it can be stored in the Mini-Memory (assuming it will fit in the 4K of Random Access Memory in the module). The Mini-Memory has a battery inside that will maintain the program for a very long time. I understand that TI may in the near future make tools available (both hardware and software) to facilitate the production of real Command Modules by third parties.

How do you reduce the influence of static charges to the computer?

First let's review the type of problems static electricity can cause. One type of problem is "computer forgetfulness." Perhaps you have been in the middle of a program, touched the keyboard, and the computer just went bananas. A slight discharge of static (that you may not have felt) traveled from your body directly to the computer's random access memory, causing "altered states." This type of problem usually does not cause permanent damage to the system (but it may destroy the program . . .). Another less common problem from static discharge is permanent damage to a portion of the computer system. This could happen even without power turned on the computer. TI has done a very good job designing the computer circuits (in my opinion) to greatly reduce the possibility of permanent damage. Your job is to safeguard your system by providing a low static environment. Here are some hints:

1. Control the humidity in the room. During cold winter months and in areas of low relative humidity, static charges are very likely to build up. Install a humidifier in your home (this may be good for you, too).
2. Remove carpeting from the computer area or move the computer to a room without carpet. A vinyl floor covering or special anti-static carpeting could be used.
3. Place anti-static mats near the computer work area. This is if your entire house is carpeted and it is not practical to remove it. The anti-static mats will help reduce the problem. These mats should be around 4 x 5 feet.

For those folks living in a place like Phoenix, Arizona during the summer with a fully carpeted apartment, "May the Force *not* be with you!"

Spotlight . . . from p. 33

cide to go into the software business . . . with this as your first product?

CME: Well, it was actually people that saw the game who convinced me. They felt that I could write programs that other people would enjoy. However, the biggest motivating factor was when I was lucky enough to win the $3000 first prize in TI's author incentive contest. That just gave me the wings to fly.

GMK: *That was your Household Inventory program wasn't it?*

CME: Right. That particular program has been very well received. It's been one of our best sellers probably because it could justify the purchase of a 99/4 just for the insurance value.

GMK: *So once you made the decision to enter the software business, you must have started looking for other ideas—games that you could produce and so have more products available for sale. What was your next game?*

CME: The next game was *SAM Defense*. That one was designed while keeping in mind all the shortcomings with BASIC that I had found when working on the baseball game—for example, the slowness of switching screens. *SAM Defense* is an ideal game for TI BASIC because it is generally a static screen: You can design a beautiful screen and leave it on. There are very few moving objects, yet it is very challenging to me. It's like a game of chess—trying to locate an enemy airplane, put it in the crosshairs, and then shoot it down.

GMK: *Where did the basic idea for this game actually come from?*

CME: *SAM Defense* is to the best of my knowledge an almost exact replica of a Russian SAM (surface-to-air missile) site. I had access to both the American SAM sites and the Russian ones, but out of better judgement, I decided to display the Russian one.

GMK: *Would you please elaborate on that . . .*

CME: Well, it's in the sense that both countries have a surface-to-air missile defense system, and that both are a secret. If I'm going to divulge any secrets, I'd rather divulge the enemy's and keep the FBI and CIA on my side.

GMK: *Is the game more a game of skill, or strategy, or a combination of both?*

CME: Well, as I see computer games, there are either games that require a lot of visual perception and manual dexterity—where you have to move fast and shoot things fast—and games like chess that are all strategy. *SAM Defense* is a combination of both; you need to move the radar screens fairly fast to try to locate the enemy, but you have to use cunning, and have to anticipate what the enemy is going to do.

GMK: *What do you feel is the most challenging aspect of the game?*

CME: The fact that the enemy is not a dummy. He is going to jam your frequencies and you have to switch frequencies and try to relocate him. Everytime he jams, he is going to switch positions on you. Even after you get him in your crosshairs and you lock your radar on him, if he jams your radar, you will have to hand-guide the missile. And in the real world, hand-guiding a missile is not an easy task by a long shot . . .

GMK: *Then SAM Defense was actually the first game that you wrote with the intention of selling?*

CME: That's right. It was our first commercial venture.

GMK: *I understand that it's still one of your most popular games after all this time. To what do you attribute this great popularity?*

CME: I believe that it's due to the challenge that people experience and the value that they recieve. I feel that if you could spend $4.00 to go see a two-hour movie, then for a $20.00 game you should get at least five times the two hours of entertainment. But *SAM Defense* provides many more hours of entertainment than that because every time you play, it is a different scenario: the airplanes come in at different altitudes; they jam at different times; and it's just a constant challenge. As a matter of fact, the first version was so challenging that we had to come out with the new version that provides three levels of difficulty. We call these (1) Boring, (2) Challenging, and (3) Mind-Blowing.

GMK: *After having taken TI BASIC to its limits with your particular style of game design and programming, how did you make the transition to the Extended BASIC language when it became available?*

CME: Well, the motivation for the transition was the challenge of using something new–sprites! I believe that the advent of sprites is what brought the arcade-type game to the 99/4. Truly, games in Extended BASIC are still much slower than real arcade games–a situation which has now been overcome with the availability of assembly language–but still, this language was the first step . . . and as such, I became challenged and intrigued. *Galactic Wars* was the first Futura game written in Extended BASIC.

GMK: *What did you learn from the experience of working with sprites and designing Galactic Wars?*

CME: I learned first, of course, that you can obtain beautiful graphics. But there is a tremendous amount of difficulty in controlling too many sprites–especially their coincidences. If you are moving sprites a little too fast, by the time you check the coincidence [to determine, for example, if objects hit or crashed], the coincidence has come and gone. And by the time you determine which of the many sprites were involved in coincidence, it is just way too late.

So therefore, you have to design a game that is well cordinated. Your coincidences should be kept minimal, or at least the number of sprites that are involved in coincidences should be kept minimal–all the while maintaining a challenging and exciting game.

GMK: *After your initial exposure and experience with sprites in Extended BASIC, did you go back to a game like All Star Baseball and see if you could improve it with the new language?*

CME: Oh definitely, because Extended BASIC now gave me in the game of baseball all the things that I had wanted to have when I was writing it. For example, rather than painting the players on the field with HCHARs, I actually used sprites. And that gave me the ability to move them around with a minimum of difficulty and a minimum of time.

GMK: *Then you feel that the version of All Star Baseball in Extended BASIC is a much more realistic game?*

CME: If not actually more realistic, it is much faster which makes it *appear* more realistic. That's because baseball is a slow game until the pitch is thrown and the batter hits it; then it becomes a very fast game. That is the point where sprites are necessary.

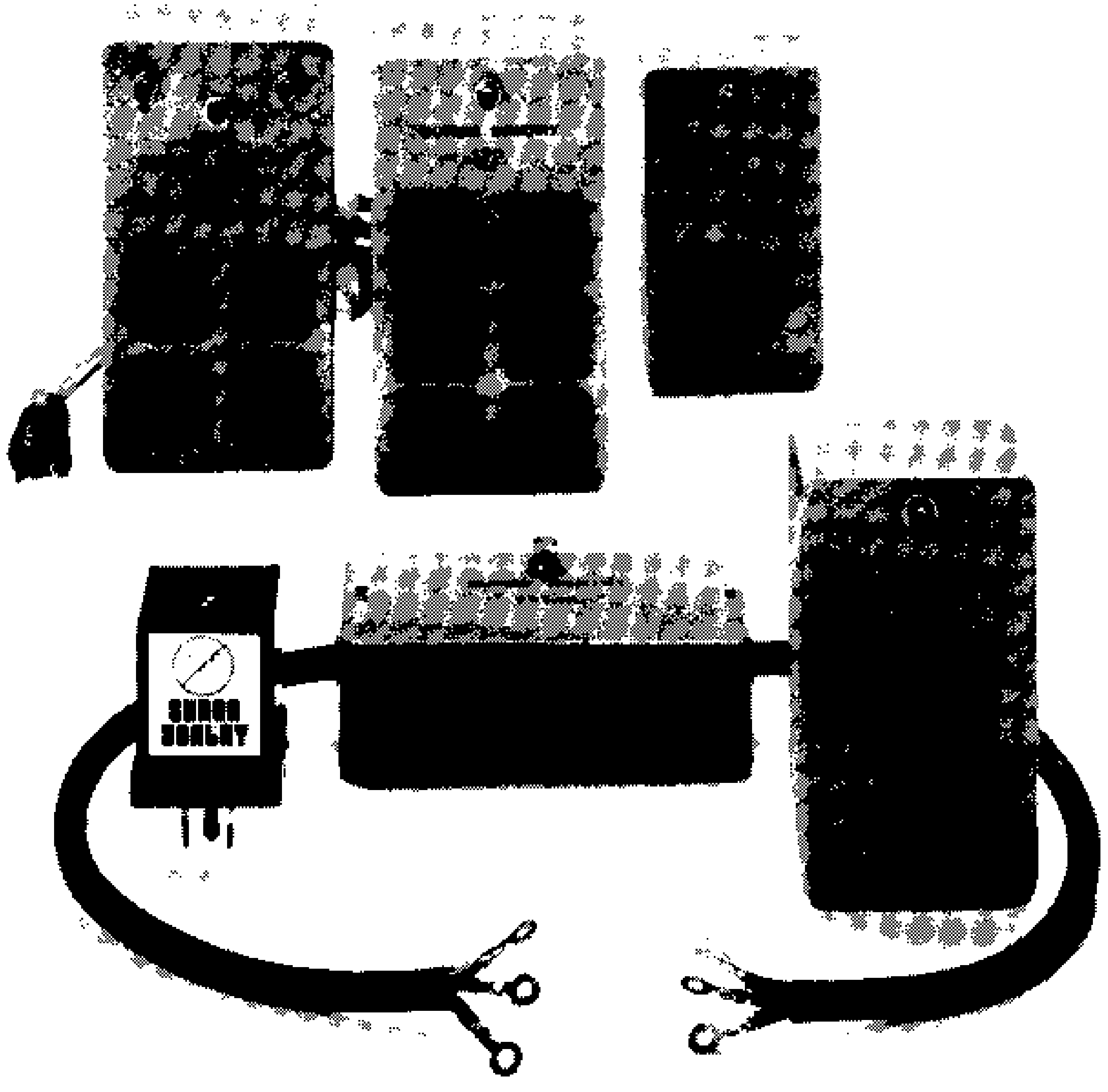
GMK: *Where did the idea for All Star Bowling originate . . . were you simply looking for another sport to simulate?*

CME: Not necessarily a sport; the fact that it turned out to be a sport was merely an accident. What intrigued me about bowling was the challenge of the vectors involved when a ball strikes the pins, and each pin–depending on the angle at which the ball had hit it–would hit other pins. Trying to approximate the reality of it all was a very challenging situation.

GMK: *What was your initial design approach in this project?*

CME: I approached it again thinking about reality. Even a very good bowler does not bowl 300-point games all

Continued on p. 52



Adventure Registry

WIZARD'S DOMINION

An Adventure Successfully Completed

By Sharlee Brader

143 Sunset Hwy.
Electric City, WA 99123

I found myself in the garb of an Evil Wizard—standing in the entrance of a deep, dark corridor in a dangerous cave with exit passages intermittently visible along its ominous length. I've come armed with the "Magic Powers of Transport and Map" so that I can mystically transport myself to other parts of the cavern to avoid the death—dealing Vorikta Ogre, or catch a bird's eye view of the cave so that my precarious position may be better understood. I'm carrying a sharp knife and a sword which I wisely purchased before I began my odyssey, but I still feel a little apprehensive about passing through this particular corridor . . .

Wizard's Dominion is an adventure game by American Software Design & Distribution Co. (Cottage Grove, MN). It is different from the Adventure International series (sold by Texas Instruments) in that it makes use of screen graphics as an integral part of the action. These graphics include three-dimensional views of maze-like caves, as well as ogres and pots of gold. By turning left or right, you can view any direction before deciding which corridor to move forward into.

There are 1,000 caves in each 10 levels—with each level more difficult than the last—so you won't quickly run out of new caves to explore. Each cave has a bag of gold and a Vorikta Ogre. Your objective is to find the gold, and exit into the Wizard's Chamber where you will be rewarded with one magic power. In addition to Transport and Map, you can acquire the power of Lightening Bolt (to destroy a weaker Ogre or weaken a stronger one), Magic Sword (to increase your chances of hitting the Ogre during battle), Magic Armor (to increase your protection factor), Wall (to build a wall between you and your enemy), and Extra Strength.

You begin the play by preparing the character you choose. You may be a Wizard's Apprentice, an Evil Wizard, a Hero, or an Evil Prince. Each character has different strength factors, weapons, money or magic powers which you feed into the computer. As you successfully complete cave explorations, you increase your magic powers and/or discover treasure with which to buy more weapons.

The second aspect of play consists of exploring the cave in search of both gold and the Wizard's Chamber, while attempting to avoid the Vorikta Ogre. Without the power of Map, finding your way through the caves can be more then challenging—perhaps even mind-boggling.

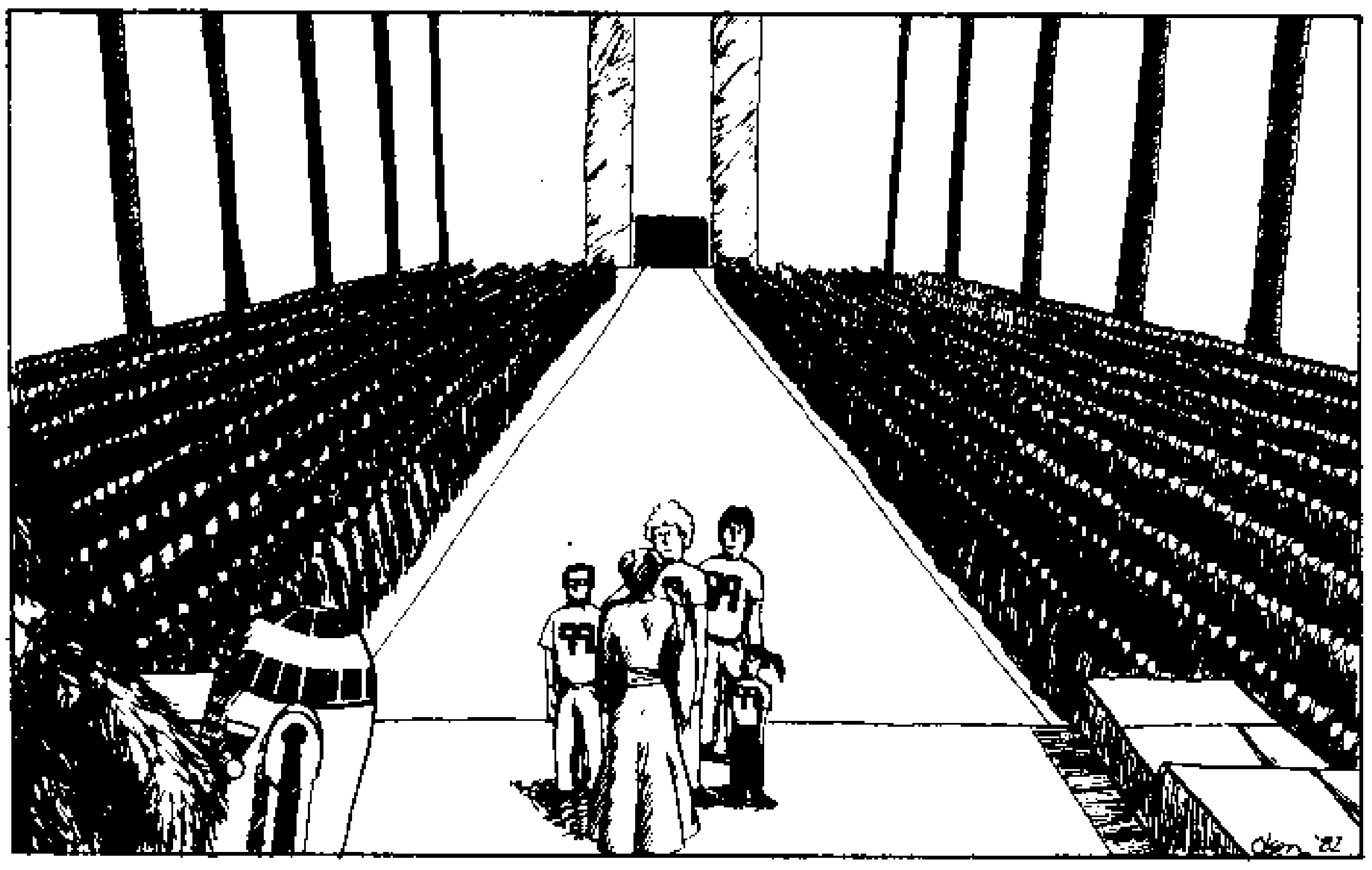
The battle phase begins when you make contact with a Vorikta Ogre. He may pop up suddenly or you may see him in the distance and have time to run or transport yourself to another part of the cavern. If you decide to stay and fight, the computer will ask you to choose your weapon. Offensive and defensive probability charts help you decide the best odds of success for each weapon and each form of attack. Note that the battle phase can become monotonous unless you vary weapons and try different strategies. Luckily, there are surprises, such as broken weapons and the possibility of hitting the Ogre in the heart, which add variety to this phase of play. If you defeat the Ogre, you can continue through the cave looking for the gold and the Wizard's Chamber.

The 3-D graphics consisting of mazes and corridors are satisfying to view and explore. Unfortunately, you quickly learn that the only surprises are *one* Ogre and *one* bag of gold per cave. You can't expect to find different monsters or varied rewards, but must instead be satisfied with the challenge of negotiating the cave with your limited powers, and fighting the Vorikta Ogre with the weapons at hand. Once you've defeated the Ogre, your path is clear. But even that is small consolation when you're feeling hopelessly lost . . .

If you like puzzles and a little bit of adventure magic, you'll surely enjoy the *Wizard's Dominion*. It is well documented with good charts and is available on both cassette tape and disk for those with Extended BASIC.

Program name:	Wizard's Dominion
Author:	Thomas Johnson
Program type:	Graphic Adventure
Language:	Extended BASIC
Distributor:	American Software Design & Distribution Co. P. O. Box 46, Cottage Grove, MN 55016-0046
Price:	$19.95 cassette, $21.95 disk

99'er HALL OF FAME



NAME: Joe Dyleski
GAME: CAR WARS
SCORE: 20,700 (4th screen with 2 computer crash cars)

NAME: Walter Dollard Jr.
GAME: TI-INVADERS
SCORE: 31,215 (level one, 14th board, blue flashers)

The following was submitted, but without verification. So we cannot induct this submittee into the Hall of Fame at this time.

Dear Sir:

After reading the article in the May/June issue on "The ATTACK", I sat down and started a game at Level 1. I used only the keyboard. I played for about 4 hours. The results were: I scored 1,022,300 points, and at one point had a ship supply of 32. One problem I found was the stardate counter would not go past 99. I played for 4 more stardates to see if it would go past 99. After four hours I had to quit, I was going "alien" happy. I hope other users have as much luck.

Joseph W. Kalinski
Buffalo, NY

FINALLY!

ASSEMBLY LANGUAGE GAME For Mini Memory Module

(or Editor Assembler)

DEFEND THE CITIES II—All the power and speed of 99/4 and 4A unleashed. Arcade action, graphics, sound effects, high score, music, multi-levels. Rotate, accelerate, fire at alien ship and bombs $19.95

EXTENDED BASIC PROGRAMS

DEFEND THE CITIES—Same game as above. (Cassette only) $19.95

MISSION: BATTLESTAR—Navigate space searching for deadly battlestar. (Cassette only) $14.95

METEOR STORM—Maneuver and fire to destroy all meteors in your path. . $14.95

GALACTIC GUNFIGHT—Fire lasers at squads of alien creatures.. ...... $14.95

All prices for cassette. (add $2.00 for disk.)

Intersoft
5407 Salem Hill
Austin, TX 78745

EASYWARE

27087 Brettonwoods
Madison Hts. MI 48071
(313) 541-2031

COSMIC COUP

A two-player board game of cosmic combat. Send your fleet into battle to win six planets. Use lasers, bombs, and shields but watch your energy. $ 20

DEFLECTORS

Search a grid for the hidden deflectors by shooting beams into it, and analyzing where they exit. A great game of deductive logic for 1-6. $ 15

TI-BASIC • cassette • TI-99/4A
complete instructions included
SEND for our FREE CATALOG

TASKFORCE

What happens when a detachment of the British Fleet meets an Argentinian Raiding Flotilla? The fight is equal-only skill will decide the victor.

AN EXCITING GAME

With full color graphics, animation and sound. Play the computer on 3 levels of difficulty or play a friend.

Available only for TI-99/4A
with Extended BASIC

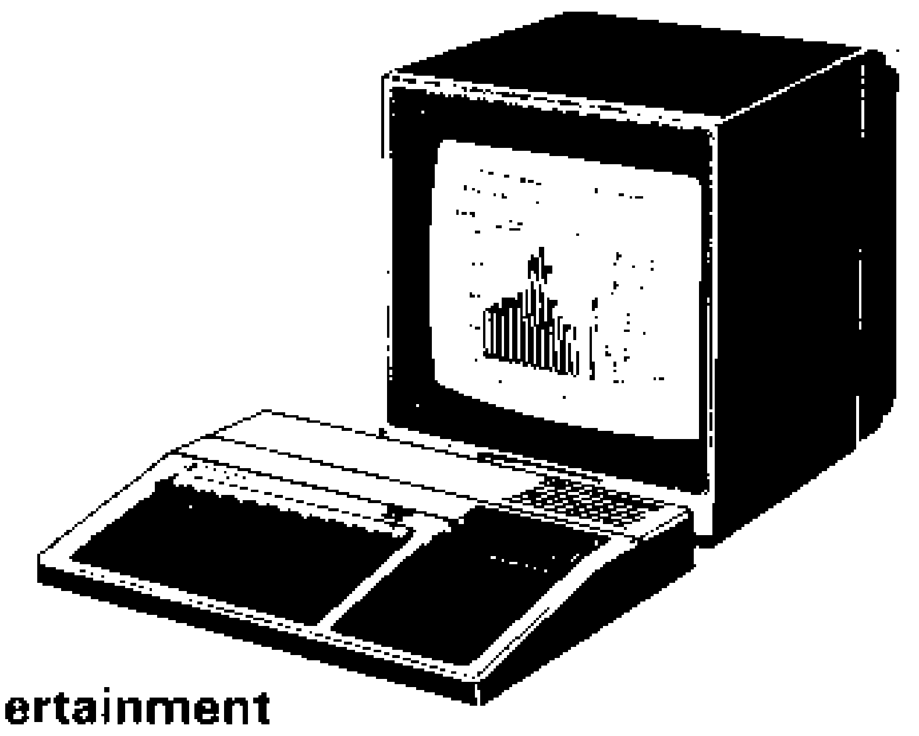
LIMITED ONE TIME OFFER

Cassette
$6.59 in Canada Post Paid
Ontario Residents add 7% sales tax.
or
$5.90 in USA Post Paid

Wentworth Supplies
Box 1005, Station A
Hamilton, Ontario
CANADA L8N 3R1

SAVE

Texas Instruments TI-99/4A



CONSOLE

TI-99/4A Home Computer with RF Modulator	339.00

PERIPHERALS

Peripheral Expansion Box	186.25
RS-232 Card	129.89
Disk Controller Card	186.25
Expansion Box Disk Drive Card	298.00
Memory Expansion Card (RAM)	223.00
P-Code Card	186.25
Solid State Speech™ Synthesizer	109.95
Telephone Coupler (Modem)	164.95
RS-232 Accessories Interface	164.95
Disk Drive Controller	219.95
Disk Memory Drive	367.95
Solid State Printer	296.95
Memory Expansion (32K RAM)	287.95
VCR Controller	519.49
P-Code Peripheral	298.00
R.F. Modulator (TV Adapter)	37.50
10" Color Monitor	324.95

OPTIONAL ACCESSORIES

Wired Remote Controllers (Pair)	26.95
Thermal Paper (2 Pack)	7.95
Dual Cassette Cable	11.95

APPLICATION PROGRAMS
Command Modules

Diagnostic	24.95
Statistics	35.95
Extended BASIC	74.95
Terminal Emulator II	39.50
Editor Assembler	74.95
Mini-Memory	74.95

Home Management/Personal Finance Command Modules

Home Financial Decisions	24.95
Household Budget Mgmt.	33.50
Securities Analysis	45.95
Personal Record Keeping	39.50
Tax/Investment Record Keeping	58.95
Personal Real Estate	58.95
Personal Report Generator	42.45

Education/Personal Enrichment Command Module

Early Learning Fun	24.95
Beginning Grammar	24.95
Number Magic	16.50
Video Graphs	16.50
Video Chess	54.50
Physical Fitness	21.95
Early Reading	45.95
Music Maker	33.50
Weight Control & Nutrition	44.95
Addition & Subtraction I	33.50
Addition & Subtraction II	33.50
Multiplication I	33.50
TI LOGO	86.60
Reading Fun	45.95

Entertainment Command Module

Football	24.50
Video Games I	24.50
Hunt the Wumpus	20.95
Indoor Soccer	24.50
Mind Challengers	20.95
A-Maze-Ing	20.95
The Attack	29.95
Blasto	20.95
Blackjack & Poker	20.95
Hustle	20.95
ZeroZap	16.50
Hangman	16.50
Connect Four	16.50
Yahtzee	20.95
Adventure (Pirate Adventure Diskette Game Included)	39.95
Adventure (Pirate Adventure Cassette Game Included	39.95
Tombstone City: 21st Century	32.95
TI Invaders	32.95
Car Wars	32.95
Munch Man	32.95

Cassette

Mystery Melody	7.95
Oldies But Goodies—Games I	12.50
Oldies But Goodies—Games II	16.50
Saturday Night Bingo	20.95
Draw Poker	16.50

Adventure Series

Pirate Adventure	24.95
Adventureland	24.95
Mission Impossible	24.95
Voodoo Castle	24.95
The Count	24.95
Strange Odyssey	24.95
Mystery Fun House	24.95
Pyramid of Doom	24.95
Ghost Town	24.95
Savage Island I&II	32.95
Golden Voyage	24.95

DOT MATRIX PRINTERS

Epson MX-80	488.00
Epson MX-80G with Graftrax (not installed) #	499.00
Epson MX-80FT	612.00
Epson MX-100	794.00
Epson 8141 ... Serial Interface	61.00
Epson 8145 Serial Interface w/2K Buffer	124.00
Epson Graftrax II#	65.00
Serial Interface Cable	25.00

#Graftrax requires Epson 8145 Interface

Texas Instruments 840 RO Basic	895.00
Texas Instruments 810 RO Basic	1480.00
Texas Instruments 820 RO Basic	1795.00
Texas Instruments 825 RO Basic	1408.00

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

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

N-VADER . . . from p. 35

```
100 REM *****************
110 REM *    N-VADER    *
120 REM *****************
130 REM
140 REM BY J. R. DEW
150 REM 99'ER VERSION 1.6.1XB
160 REM
170 CALL CHAR(104,"FF99FFFFA5A5A5A5")
180 CALL CLEAR
190 FOR X=1 TO 20
200 FOR Y=1 TO 19 STEP 9
210 DISPLAY AT(X,Y)SIZE(8):"N-VADER"
220 NEXT Y
230 NEXT X
240 CALL SPRITE(#1,104,11,1,1,0,20)
250 DISPLAY AT(22,2):"PRESS ENTER TO PLAY"
260 CALL SPRITE(#2,104,13,65,256,0,-20)
270 CALL SPRITE(#3,104,9,73,1,0,100)
280 CALL SPRITE(#4,104,4,81,256,0,-99)
290 ACCEPT AT(22,22)BEEP:X$
300 CALL DELSPRITE(ALL)
310 INPUT "INSTRUCTIONS (Y/N)?":X$
320 IF X$<>"Y" THEN 490
330 CALL CLEAR
340 PRINT "ALIEN CREATURES ARE":"ATTACKING THE
    EARTH!"
350 PRINT "YOU COMMAND THE ONLY DEFENSESHIPS.
    ONBOARD COMPUTERS    CONTROL THE
    LASERS WHICH CAN DESTROY THE INVADERS."
360 PRINT "THE INVADERS WILL NOT ATTACKYOU,
    ONLY THE EARTH."
370 CALL SPRITE(#1,104,11,1,1,0,20)
380 PRINT
390 INPUT "HIT ENTER WHEN READY":X$
400 CALL CLEAR
410 PRINT "IN THIS GAME, YOU CONTROL
    THE NUMBER OF INVADERS,     THEIR SPEED,
    YOUR SPEED &   THE LASER RANGE OF YOUR
    SHIPS."
420 PRINT "SUGGESTED VALUES ARE:
    INVADERS=6,SPEED=8":"YOUR SPEED=3,RANGE=2 5."
430 PRINT "A SMALLER RANGE MAKES THE
    GAME HARDER."
440 PRINT "YOU ALSO CONTROL THE LENGTH OF THE
    GAME. WHEN ASKED     'END OF GAME', ENTER THE"
450 PRINT "NUMBER OF TIMES THE ALIENS  HIT
    EARTH FOR THE GAME TO BE OVER."
460 PRINT
470 INPUT "ENTER WHEN READY":X$
480 CALL DELSPRITE(#1)
490 HIT,ZAP1,ZAP2=0
500 INPUT "NUMBER OF PLAYERS? ":NP
510 IF NP<0 OR NP>2 THEN 500
520 NP=INT(NP)
530 INPUT "PLAYER 1 NAME? ":P1$
540 IF NP=1 THEN 560
550 INPUT "PLAYER 2 NAME? ":P2$
560 PRINT "NUMBER OF INVADERS?"
570 INPUT INV
580 IF INV<1 OR INV>8 THEN 560
590 PRINT "INVADER SPEED?"
600 INPUT IS
610 IF IS<1 THEN 590
620 PRINT "DEFENDER SPEED(1-9)?"
630 INPUT SPD
640 IF SPD<=0 THEN 620
650 PRINT "DEFENSE RANGE?"
660 INPUT RNG
670 IF RNG<1 OR RNG>200 THEN 650
680 PRINT "END OF GAME?"
690 INPUT WIN
700 INPUT "JOYSTICKS (Y/N)? ":X$
710 IF SEG$(X$,1,1)="Y" THEN JS=1
720 IF WIN<1 THEN 680
730 CALL CHAR(100,"FFFFFFFFFFFFFFFF")
740 CALL CHAR(96,"00080B1C7F1C0808")
750 CALL SCREEN(2)
760 CALL CLEAR
770 CALL COLOR(9,16,16)
780 CALL COLOR(3,2,3)
790 CALL COLOR(4,2,3)
800 FOR X=22 TO 24
810 CALL HCHAR(X,1,100,32)
820 NEXT X
830 FOR X=1 TO INV
840 CALL SPRITE(#X,104,3+X,1,INT(RND*256)+1,
    INT(RND*IS)+1,INT(RND*IS)-IS/2)
850 NEXT X
860 IF NP=1 THEN CALL SPRITE(#9,96,16,100,128)
    ELSE CALL SPRITE(#9,96,16,100,56)
870 IF NP=2 THEN CALL SPRITE(#10,96,15,100,200)
```

EXPLANATION OF THE PROGRAM
N-VADER

Line Nos.	
170	Define invader character.
190-290	Display title screen with sprites.
330-490	Instructions.
500-720	Get parameters.
800-820	Draw Earth.
830-850	Draw invader sprites.
860-870	Draw player sprites. Note that positioning changes for Player #1 depending upon number of players.
880-1010	Check for player scoring.
1020-1070	Check for invaders at Earth.
1080-1120	Adjust player motion.
1130-1330	Process end of game.
1360-1380	Subroutine to introduce new invader during game.
1390-1510	Subprogram to simulate joysticks on keyboard.

```
880 FOR X=1 TO INV
890 CALL COINC(#X,#9,RNG,B)
900 IF NP=2 THEN CALL COINC(#X,#10,RNG,B2)
910 IF B>=0 AND B2>=0 THEN 1020
920 CALL PATTERN(#X,100)
930 CALL SOUND(-500,-3,0)
940 GOSUB 1360
950 IF B>=0 THEN 980
960 ZAP1=ZAP1+1
970 DISPLAY AT(23,3)SIZE(4):ZAP1
980 IF B2>=0 THEN 1080
990 ZAP2=ZAP2+1
1000 DISPLAY AT(23,23)SIZE(4):ZAP2
1010 GOTO 1080
1020 CALL POSITION(#X,V(X),H(X))
1030 IF V(X)<158 THEN 1080
1040 GOSUB 1360
1050 CALL SOUND(-50,-2,0)
1060 HIT=HIT+1
1070 DISPLAY AT(23,14)SIZE(4):HIT
1080 IF JS=1 THEN CALL JOYST(1,JX,JY)ELSE CALL
     KEYST(1,JX,JY)
1090 CALL MOTION(#9,-JY*SPD,JX*SPD)
1100 IF NP=1 THEN 1130
1110 IF JS=1 THEN CALL JOYST(2,JX,JY)ELSE CALL
     KEYST(2,JX,JY)
1120 CALL MOTION(#10,-JY*SPD,JX*SPD)
1130 IF HIT<WIN THEN 1340
1140 CALL DELSPRITE(ALL)
1150 CALL SCREEN(16)
1160 CALL CLEAR
1170 CALL COLOR(3,2,1)
1180 CALL COLOR(4,2,1)
1190 CALL SPRITE(#1,104,7,1,1,0,25)
1200 PRINT "GAME OVER"
1210 PRINT "EARTH HITS";TAB(18);HIT
1220 PRINT P1$;" DESTOYED";ZAP1;" ALIENS"
1230 PRINT TAB(10);INT(100*ZAP1/
.    (HIT+ZAP1+ZAP2));" PER CENT"
1240 IF NP=2 THEN PRINT P2$;" DESTROYED";
     ZAP2;" ALIENS"
1250 IF NP=2 THEN PRINT TAB(10);INT(100*ZAP2/
     (HIT+ZAP1+ZAP2));" PER CENT"
1260 FOR X=1 TO 100
1270 CALL SOUND(50,440,0)
1280 CALL SOUND(99,880,0)
1290 NEXT X
1300 CALL DELSPRITE(#1)
1310 CALL SCREEN(8)
1320 JS=0
1330 GOTO 170
1340 NEXT X
1350 GOTO 880
1360 CALL DELSPRITE(#X)
1370 CALL SPRITE(#X,104,15-X,1,INT(RND*256)+1,
     INT(RND*IS)+1,INT(RND*IS)-IS/2)
1380 RETURN
1390 SUB KEYST(N,X,Y)
1400 CALL KEY(N,K,S)
1410 IF S=0 THEN X,Y=0 :: SUBEXIT
1420 IF K=2 THEN X=-4 :: Y=0
1430 IF K=4 THEN X=-4 :: Y=4
1440 IF K=5 THEN X=0 :: Y=4
1450 IF K=6 THEN X=4 :: Y=4
1460 IF K=12 THEN X,Y=4
1470 IF K=3 THEN X=4 :: Y=0
1480 IF K=14 THEN X=4 :: Y=-4
1490 IF K=0 THEN X=0 :: Y=-4
1500 IF K=15 THEN X=-4 :: Y=-4
1510 SUBEND
```

Derby . . . from p. 35

```
100 REM  **********************
110 REM  * COUNTY FAIR DERBY *
120 REM  **********************
130 REM      BY JOHN GUNTER
135 REM    99'ER VERSION 1.6.1
140 CALL CLEAR
150 CALL COLOR(2,2,14)
160 FOR I=3 TO 8
170 CALL COLOR(I,2,12)
180 NEXT I
190 CALL HCHAR(24,2,42,29)
200 PRINT
210 PRINT TAB(8);"COUNTY FAIR DERBY"::
220 PRINT TAB(8);"A FIVE HORSE RACE"::
230 PRINT TAB(4);"YOU CAN BET FOUR WAYS:"::
240 PRINT "<1> WIN      PAYS 4 TO 1"::
250 PRINT "<2> PLACE    PAYS 3 TO 2"::
260 PRINT "<3> SHOW     PAYS 2 TO 3"::
270 PRINT "<4> PARLAY   PAYS 15 TO 1"::
280 CALL HCHAR(24,9,42,14)
290 PRINT ::
300 PRINT "PARLAY<PICK 1ST;  AND 2ND;>":::
310 PRINT "EACH PLAYER IS GIVEN $200"::
320 CALL HCHAR(24,2,42,29)
330 CALL VCHAR(1,2,42,24)
340 CALL VCHAR(1,30,42,24)
350 RESTORE 4370
360 READ DU,NO
370 IF DU=0 THEN 400
380 CALL SOUND(300*DU,NO,5)
390 GOTO 360
400 PRINT "PRESS ANY KEY"
410 CALL KEY(0,KEY,STAT)
420 IF STAT=0 THEN 410
430 CALL CLEAR
440 PRINT TAB(6);"***HANG  ON***"::::
450 PRINT TAB(7);"GOTTA  GET THE":::
460 PRINT TAB(11);"HORSES":::::
470 DIM H$(50)
480 H$(1)="000000004020100F"
490 H$(2)="000008080F1F30F0"
500 H$(3)="0F0F102040000000"
510 H$(4)="F0F0080402000000"
520 H$(5)="000000000000007F"
530 H$(6)="0000000601E3EF0"
540 H$(7)="0F0F080402000000"
550 H$(8)="F0F0102040000000"
560 H$(9)="0000000103070101"
570 H$(10)="00000F1F30000007"
580 H$(11)="00001F3F31000003"
590 H$(12)="000000000103060C"
600 H$(13)="0000070706060707"
610 H$(14)="0000808080808080"
620 H$(15)="0000C0E0606060E0"
630 H$(16)="0000C0E0F07070E0"
640 H$(17)="000060E0E0606060"
650 H$(18)="0000F0F00000C0F0"
660 H$(19)="0101010107070000"
670 H$(20)="1F1830303F3F0000"
680 H$(21)="030000313F1F0000"
690 H$(22)="0F0F000000000000"
700 H$(23)="0000060703010000"
710 H$(24)="B0B0B080E0E00000"
720 H$(25)="80000000E0E00000"
730 H$(26)="E07070F0E0C0000
740 H$(27)="F1F160606060000
750 H$(28)="7111111F0E00000
760 H$(29)="00003030303031
770 H$(30)="00000C0C0C0C8C8C
780 H$(31)="00003C3C181818
790 H$(32)="000030303C3E373
800 H$(33)="00000C0C0C0C0C8C
810 H$(34)="00003F3F303030
820 H$(35)="0000FCFC0C0000FC
830 H$(36)="1F1F1F1E1C1C000
840 H$(37)="F8F8F8F838380000
850 H$(38)="181818183C3C000
860 H$(39)="31303030303000
870 H$(40)="CCEC7C3C1C0C000
880 H$(41)="3F0000303F3F000
890 H$(42)="FC0C0C0CFCFC000
900 D=120
910 K=1
920 FOR D=D TO D+7
930 CALL CHAR(D,H$(K))
940 K=K+1
950 NEXT D
960 IF D>152 THEN 980
970 GOTO 910
980 CALL CLEAR
990 CALL COLOR(11,15,6)
1000 CALL COLOR(12,14,11)
1010 CALL COLOR(13,13,11)
1020 CALL COLOR(14,2,11)
1030 CALL COLOR(15,7,11)
1040 CALL COLOR(16,5,11)
1050 CALL COLOR(2,2,12)
1060 CALL CLEAR
```

Battle Star . . . from p. 35

```
100 REM ***************
110 REM *             *
120 REM * BATTLE STAR *
130 REM *             *
140 REM ***************
150 REM BY W.K. BALTHROP
160 REM 99'ER VERSION 1.6.1XB
170 RANDOMIZE
180 DIR=1 :: CALL CLEAR
190 CALL COLOR(9,7,1):: CALL COLOR(10,6,1)::
    CALL SCREEN(2)
200 CALL CHAR(96,"00000000000070707"):: CALL CHAR
    (97,"1818183C7EFFDB99")
210 CALL CHAR(98,"0000000000E0E0E0"):: CALL CHAR
    (99,"070E1CFFFF1C0E07")
220 CALL CHAR(104,"18423C99993C4218"):: CALL
    CHAR(101,"E07038FFFF3870E0"):: CALL CHAR
    (102,"070707")
230 CALL CHAR(107,"104628240A923044")
240 CALL CHAR(103,"99DBFF7E3C18181B"):: CALL
    CHAR(100,"E0E0E0")
250 CALL CHAR(112,"307B7C477C7830"):: CALL CHAR
    (113,"1010386CEEEE7C")
260 CALL CHAR(114,"0C1E3EE23E1E0C"):: CALL CHAR
    (115,"007CEEEE6C381010")
270 CALL CHAR(116,"101038FE381010"):: CALL CHAR
    (117,"0000183CFF7E2442")
280 CALL CHAR(105,"1818181818181818"):: CALL
    CHAR(106,"000000FFFF")
290 FOR COL=1 TO 12 :: CALL COLOR(COL,16,1)::
    NEXT COL
300 L=100 :: S=5 :: SC=0 :: SA1,SB1,SA2,SB2,SA3,
    SB3,SA4,SB4=0 :: T=0
310 GOSUB 350
320 GOSUB 390 :: GOSUB 650
330 L=L-.5 :: IF L<1 THEN L=1
340 DISPLAY AT(24,3):SC :: GOTO 320
350 CALL SPRITE(#10,96,16,81,113,0,0,#11,97,16,
    81,121,0,0,#12,98,16,81,129,0,0)
360 CALL SPRITE(#13,99,16,89,113,0,0,#14,104,7,
    89,121,0,0,#15,101,16,89,129,0,0)
370 CALL SPRITE(#16,102,16,97,113.0,0,#17,103,
    16,97,121,0,0,#18,100,16,97,129,0, 0)
380 RETURN
390 CALL KEY(0,K,S):: IF S=0 THEN RETURN
400 IF K=69 THEN 450
410 IF K=83 THEN 500
420 IF K=88 THEN 550
430 IF K=68 THEN 600
440 RETURN
450 IF SA1=0 AND SB1=0 THEN CALL VCHAR(1,16,105,10):
    : CALL SOUND(10,800,0):: CAL L VCHAR(1,16,32,10):
    : SC=SC-10 :: RETURN
460 IF SB1=0 THEN CALL VCHAR(2,16,105,9):
    : CALL SOUND(500,110,2,-5,2):: CALL
    VCHAR(2,16,32,9):: SC=SC+50 :: SA1=0 :: RETURN
470 CALL POSITION(#1,P1,P2):: IF P1>76 THEN 840
480 P1=INT(P1/8)+1 :: CALL VCHAR(P1,16,105,10-P1):
    : CALL SOUND(200,110,10,-5,8):
    : CALL VCHAR(P1,16,32,10-P1)
490 CALL DELSPRITE(#1):: SC=SC+20 :: SB1=0 :: RETURN
500 IF SA2=0 AND SB2=0 THEN CALL HCHAR(12,1,106,14):
    : CALL SOUND(10,800,0):: CAL L HCHAR(12,1,32,14):
    : SC=SC-10 :: RETURN
510 IF SB2=0 THEN CALL HCHAR(12,3,106,12):
    : CALL SOUND(500,110,2,-5,2):: CALL HC HAR(12,3,
    32,12):: SC=SC+50 :: SA2=0 :: RETURN
520 CALL POSITION(#2,P1,P2):: IF P2>86 THEN 840
530 P2=INT(P2/8)+1 :: CALL HCHAR(12,P2,106,15-P2):
    : CALL SOUND(200,110,10,-5,8):
    : CALL HCHAR(12,P2,32,15-P2)
540 CALL DELSPRITE(#2):: SC=SC+20 :: SB2=0 :: RETURN
550 IF SA3=0 AND SB3=0 THEN CALL VCHAR(14,16,105,10):
    : CALL SOUND(10,800,0):: CALL VCHAR(14,16,32,10):
    : SC=SC-10 :: RETURN
560 IF SB3=0 THEN CALL VCHAR(14,16,105,10):
    : CALL SOUND(500,110,2,-5,2):: CALL V
    CHAR(14,16,32,10):: SC=SC+50 :: SA3=0 :: RETURN
570 CALL POSITION(#3,P1,P2):: IF P1<110 AND P1>0
    THEN 840
580 P1=INT(P1/8)+1 :: CALL VCHAR(14,16,105,P1-14):
    : CALL SOUND(200,110,10,-5,8):
    : CALL VCHAR(14,16,32,P1-14)
590 CALL DELSPRITE(#3):: SC=SC+20 :: SB3=0 :: RETURN
600 IF SA4=0 AND SB4=0 THEN CALL HCHAR(12,18,106,14):
    : CALL SOUND(10,800,0):: CALL HCHAR(12,18,32,14):
    : SC=SC-10 :: RETURN
610 IF SB4=0 THEN CALL HCHAR(12,18,106,13):
    : CALL SOUND(500,110,2,-5,2):: CALL HCHAR(12,18,
    32,13):: SC=SC+50 :: SA4=0 :: RETURN
620 CALL POSITION(#4,P1,P2):: IF P8<142 AND P8>0
    THEN 840
630 P2=INT(P2/8):: CALL HCHAR(12,18,106,P2-15):
    : CALL SOUND(200,110,10,-5,8):
    : CALL HCHAR(12,18,32,P2-15)
640 CALL DELSPRITE(#4):: SC=SC+20 :: SB4=0 :
    : RETURN
650 IF SB1=0 THEN P1,P2=0 :: GOTO 660
    ELSE CALL POSITION(#1,P1,P2)
660 IF SB2=0 THEN P3,P4=0 :: GOTO 670
    ELSE CALL POSITION(#2,P3,P4)
670 IF SB3=0 THEN P5,P6=0 :: GOTO 680
    ELSE CALL POSITION(#3,P5,P6)
680 IF SB4=0 THEN P7,P8=0 :: GOTO 690
    ELSE CALL POSITION(#4,P7,P8)
690 IF P1>76 OR P4>86 OR(P5<110 AND P5>0)OR
    (P8<142 AND P8>0)THEN 840
700 NS=INT(RND*L):: IF NS>10 THEN RETURN
710 NS=INT(RND*4)+1 :: ON NS GOTO 730,
    760,790,820
720 IF SA1=1 AND SB1=1 THEN RETURN
730 CALL HCHAR(2,16,115):: SA1=1 :: IF L<80 AND
    SB1=0 THEN CALL SPRITE(#1,116,7,
    17,120,11-(L/10),0):: SB1=1
740 RETURN
750 IF SA2=1 AND SB2=1 THEN RETURN
760 CALL HCHAR(12,3,112):: SA2=1 :: IF L<80 AND
    SB2=0 THEN CALL SPRITE(#2,116,7,
    88,17,0,11-(L/10)):: SB2=1
770 RETURN
780 IF SA3=1 AND SB3=1 THEN RETURN
790 CALL HCHAR(23,16,113):: SA3=1 :: IF L<80 AND
    SB3=0 THEN CALL SPRITE(#3,116,7,175,120,
    -11+(L/10),0):: SB3=1
800 RETURN
810 IF SA4=1 AND SB4=1 THEN RETURN
820 CALL HCHAR(12,30,114):: SA4=1 :: IF L<80 AND
    SB4=0 THEN CALL SPRITE(#4,116,7,88,216,0,
    -11+(L/10)):: SB4=1
830 RETURN
840 CALL DELSPRITE(#1,#2,#3,#4):
    : CALL SOUND(2000,110,2,220,2,1000,30,-4,2)
850 FOR BUB=10 TO 18 :: CALL MOTION(#BUB,
    INT(RND*40)-20,INT(RND*40)-20):
    : CALL PATTERN(#BUB,107):: NEXT BUB
860 CALL SOUND(1000,110,2,220,2,110,2,-5,2):
    : CALL SOUND(1,40000,30)
870 CALL DELSPRITE(ALL):: CALL CLEAR
880 DISPLAY AT(12,7):"YOUR SCORE IS":TAB(10);SC
890 CALL DELSPRITE(ALL)
900 DISPLAY AT(22,1):"DO YOU WISH TO PLAY AGAIN?
    (Y/N)."
910 ACCEPT AT(23,8)VALIDATE("YN"):ANS$ :
    : IF ANS$="N" THEN 950
920 CALL CLEAR :: GOSUB 350 :: SC=0 :: L=100
930 SB1,SB2,SB3,SB4,P1,P2,P3,P4,P5,P6,P7,P8=0
940 RETURN
950 END
```

EXPLANATION OF THE PROGRAM
Battle Star

Line Nos.	
170-290	Initialize colors and characters.
300	Initialize variables.
310	Jump to subroutine to create Battle Star.
320-340	Main program loop.
350-380	Setup sprites to create the Battle Star.
390-440	Read keyboard; branch to fire laser cannon.
450-490	Fire laser up.
500-540	Fire laser left.
550-590	Fire laser down.
600-640	Fire laser right.
650-690	Check position of missiles, and branch off if Battle Star hit.
700	Checks the chance of another ship appearing.
710	Decides which ship will appear, and branches to subroutine.
720-740	Place top ship on screen—with missile if game progressed.
750-770	Place left ship on screen—with missile if game progressed.
780-800	Place bottom ship on screen—with missile if game progressed.
810-830	Place right ship on screen—with missile if game progressed.
840-870	Battle Star is hit and destroyed.
880-910	Display score. Play again? Accept answer.
920-940	Re-initialize variables.
950	End.

```
1070 X=1
1080 CALL CLEAR
1090 PRINT "TYPE PLAYER'S NAME ?":::
1100 PRINT "AFTER THE LAST PLAYERS NAME"::
1110 PRINT "HAS BEEN ENTERED TYPE LAST":::
1120 INPUT "NAME ?":NAME$(X)
1130 IF NAME$(X)="LAST" THEN 1570
1140 IF X>=9 THEN 1160
1150 GOTO 1190
1160 PRINT "EIGHT IS THE MAX.NUM.OF PLAYERS"
1170 PRINT "TYPE LAST TO CONTINUE"
1180 GOTO 1120
1190 TOT(X)=200
1200 CALL CLEAR
1210 GOSUB 1230
1220 GOTO 1080
1230 PRINT :"O.K. ";NAME$(X);" PICK A HORSE ?"::
1240 INPUT "HORSE ? ":HO(X)
1250 IF HO(X)>5 THEN 1270
1260 GOTO 1310
1270 GOSUB 1290
1280 GOTO 1240
1290 PRINT :"NUM. TOO BIG TRY AGAIN"::
1300 RETURN
1310 PRINT :"WHAT KIND OF BET ?<I TO 4>"::
1320 PRINT "<1>=  WIN"::
1330 PRINT "<2>=  PLACE"::
1340 PRINT "<3>=  SHOW"::
1350 PRINT "<4>=  PARLAY"::
1360 INPUT "KIND ? ":KI(X)
1370 IF KI(X)>4 THEN 1400
1380 IF KI(X)=4 THEN 1500
1390 GOTO 1420
1400 GOSUB 1290
1410 GO TO 1360
1420 PRINT :"HOW MUCH DO YOU BET ?
     <$1 TO $200>"::
1430 INPUT "BET ? ":BET(X)
1440 IF BET(X)>200 THEN 1460
1450 GOTO 1480
1460 GOSUB 1290
1470 GOTO 1430
1480 X=X+1
1490 RETURN
1500 PRINT :"YOU PICKED NO.";HO(X);
     "  TO WIN"::
1510 PRINT "WHICH HORSE TO PLACE ?"::
1520 INPUT "PLACE ?":PA2(X)
1530 IF PA2(X)>5 THEN 1550
1540 GOTO 1420
1550 GOSUB 1290
1560 GOTO 1520
1570 CALL CLEAR
1580 PRINT "PRESS S TO START"
1590 CALL COLOR(2,11,11)
1600 FOR X=1 TO 22
1610 PRINT
1620 NEXT X
1630 CALL CHAR(119,"81C366181866C381")
1640 CALL HCHAR(9,1,119,30)
1650 CALL HCHAR(20,1,119,30)
1660 X=10
1670 Y=2
1680 FOR A=1 TO 10
1690 CALL HCHAR(X,Y,42,29)
1700 X=X+1
1710 NEXT A
1720 RESTORE 4350
1730 READ DU,NO
1740 IF DU=0 THEN 1770
1750 CALL SOUND(200*DU,
     NO,5)
1760 GOTO 1730
1770 CALL HCHAR(10,2,49)
1780 CALL HCHAR(12,2,50)
1790 CALL HCHAR(14,2,51)
1800 CALL HCHAR(16,2,52)
1810 CALL HCHAR(18,2,53)
1820 Z=1
1830 D=120
1840 R=10
1850 V=3
1860 GOSUB 2490
1870 D=128
1880 R=12
1890 V=3
1900 GOSUB 2490
1910 D=136
1920 R=14
1930 V=3
1940 GOSUB 2490
1950 D=144
1960 R=16
1970 V=3
1980 GOSUB 2490
1990 D=152
2000 R=18
2010 V=3
2020 GOSUB 2490
2030 Z=0
```

Continued on p. 50

Derby . . . from p. 49

```
2040 A=10
2050 B=4
2060 I=16
2070 J=4
2080 E=12
2090 F=4
2100 O=18
2110 P=4
2120 G=14
2130 H=4
2140 K=0
2150 S=0
2160 CALL KEY(0,KEY,STATUS)
2170 IF STATUS=0 THEN 2160
2180 IF KEY=83 THEN 2200
2190 GOTO 2160
2200 RANDOMIZE
2210 N=INT(5*RND)+1
2220 ON N GOTO 2230,2280,
     2330,2380,2430
2230 R=A
2240 V=B
2250 IF B=0 THEN 2200
2260 D=120
2270 GOTO 2470
2280 R=E
2290 V=F
2300 IF F=0 THEN 2200
2310 D=128
2320 GOTO 2470
2330 R=G
2340 V=H
2350 IF H=0 THEN 2200
2360 D=136
2370 GOTO 2470
2380 R=I
2390 V=J
2400 IF J=0 THEN 2200
2410 D=144
2420 GOTO 2470
2430 R=O
2440 V=P
2450 D=152
2460 IF P=0 THEN 2200
2470 CALL HCHAR(R,V-1,42)
2480 CALL HCHAR(R+1,V-1,42)
2490 CALL HCHAR(R,V,D)
2500 CALL HCHAR(R,V+1,D+1)
2510 CALL HCHAR(R+1,V,D+2)
2520 CALL HCHAR(R+1,V+1,D+3)
2530 CALL SOUND(5,700,2)
2540 IF Z=0 THEN 2560
2550 RETURN
2560 IF Q=1 THEN 2610
2570 Q=1
2580 V=V+1
2590 D=D+4
2600 GOTO 2470
2610 D=D-4
2620 Q=0
2630 IF V>28 THEN 2840
2640 V=V+1
2650 IF D=120 THEN 2720
2660 IF D=128 THEN 2750
2670 IF D=144 THEN 2780
2680 IF D=152 THEN 2810
2690 G=R
2700 H=V
2710 GOTO 2200
2720 A=R
2730 B=V
2740 GOTO 2200
2750 E=R
2760 F=V
2770 GOTO 2200
2780 I=R
2790 J=V
2800 GOTO 2200
2810 O=R
2820 P=V
2830 GOTO 2200
2840 D=(D-112)/8
2850 IF S<>0 THEN 3000
2860 S=D
2870 ON S GOSUB 2890,
     2910,2930,2950,2970
2880 GOTO 2990
2890 CALL COLOR(9,2,14)
2900 RETURN
2910 CALL COLOR(9,15,13)
2920 RETURN
2930 CALL COLOR(9,15,2)
2940 RETURN
2950 CALL COLOR(9,2,7)
2960 RETURN
2970 CALL COLOR(9,2,5)
2980 RETURN
2990 ON S GOTO 3030,3050,
     3070,3090,3110
3000 IF K<>0 THEN 3130
3010 K=D
3020 ON K GOTO 3030,3050,
     3070,3090,3110
3030 B=0
3040 GOTO 2200
3050 F=0
3060 GOTO 2200
3070 H=0
3080 GOTO 2200
3090 J=0
3100 GOTO 2200
```

99'er Program Bug

Extended BASIC Bugs

In the newest version of Extended BASIC, recursive sub-program calls are *not* immediately diagnosed as errors. *Numeric* parameters work O.K., but *strings* will crash the system. Within a sub-program, a user-defined definition won't always work: it will work if the definition of the function only involves a "formal" definition—i.e., if it doesn't involve any global variables or calls to other user-definitions.

As a result of these bugs, our *Spriter* program, as published in Vol. 1, No. 5, will not work with the latest release of Extended BASIC; it will, however, work

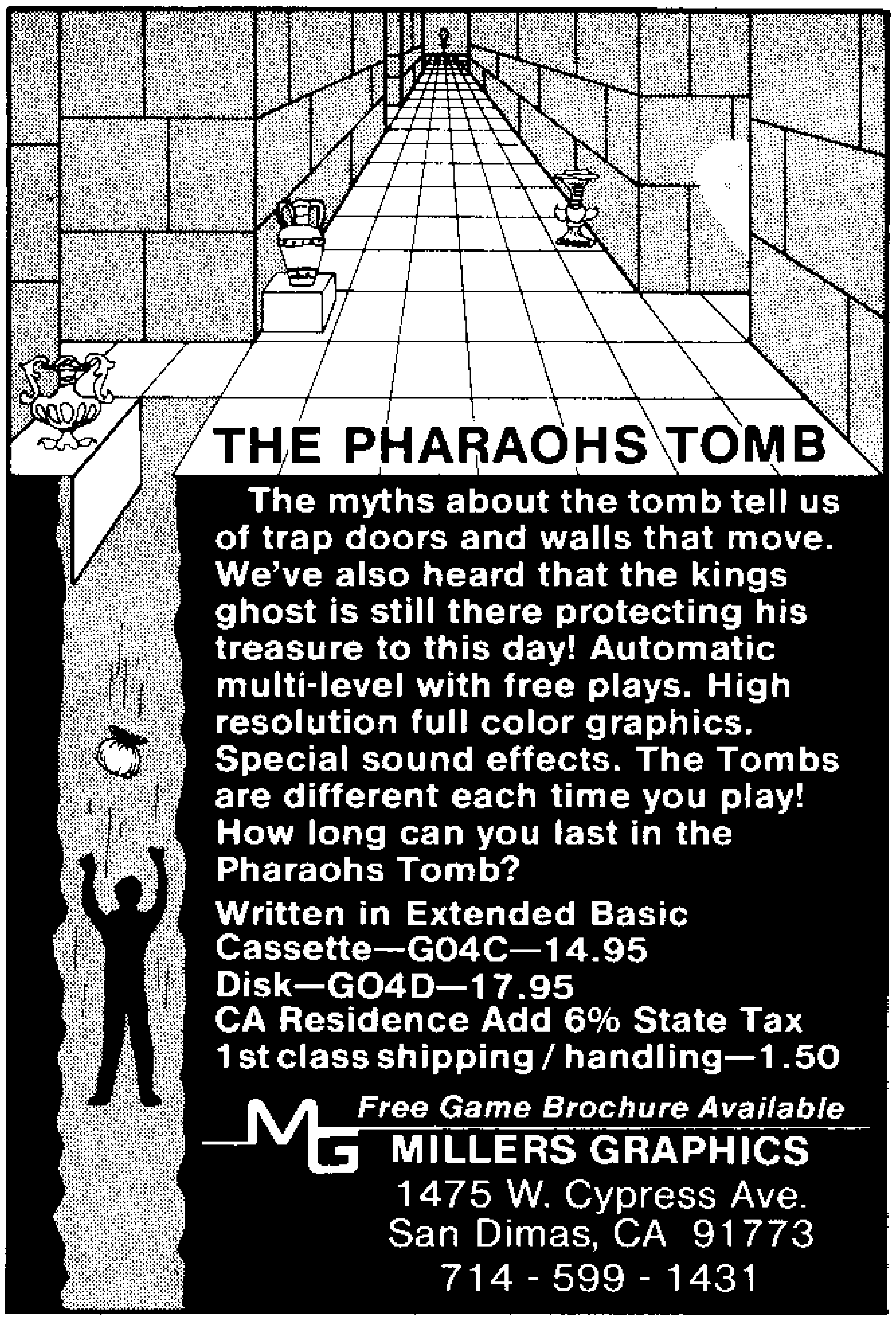
```
3110 P=0
3120 GOTO 2200
3130 R=22
3140 V=10
3150 X=S+8
3160 FOR Y=1 TO 4
3170 CALL CHAR((95+Y),
     H$(X))
3180 X=X+5
3190 NEXT Y
3200 FOR Y=1 TO 2
3210 CALL HCHAR(R,V,
     95+Y)
3220 V=V+1
3230 NEXT Y
3240 V=V-2
3250 R=R+1
3260 FOR Y=3 TO 4
3270 CALL HCHAR(R,V,95+Y)
3280 V=V+1
3290 NEXT Y
3300 CALL COLOR(10,15,6)
3310 CALL COLOR(11,15,6)
3320 R=22
3330 V=13
3340 Q=1
3350 Y=1
3360 FOR Y=Y TO Y+6
3370 CALL CHAR(103+Y,
     H$(28+Y))
3380 CALL HCHAR(R,V,
     103+Y)
3390 V=V+1
3400 NEXT Y
3410 IF Q=0 THEN 3470
3420 R=23
3430 V=13
3440 Q=0
3450 Y=8
3460 GOTO 3360
3470 PRINT :TAB(7);K;
     "PLACES"
3480 PRINT :TAB(7);D;
     "SHOWS"::
3490 PRINT
     "PRESS ANY KEY"
3500 CALL KEY(0,KEY,
     STATUS)
3510 IF STATUS=0
     THEN 3500
3520 CALL COLOR(2,2,12)
3530 CALL CLEAR
3540 X=1
3550 IF NAME$(X)="LAST" THEN 4130
3560 ON KI(X)GOTO 3570,3640,
     3720,3810
3570 IF HO(X)=S THEN 3600
3580 GOSUB 3970
3590 GOTO 3880
3600 BET(X)=BET(X)*4
3610 BET(X)=INT(BET(X)*100+.5)/100
3620 GOSUB 4090
3630 GOTO 3880
3640 IF HO(X)=S THEN 3680
3650 IF HO(X)=K THEN 3680
3660 GOSUB 3970
3670 GOTO 3880
3680 BET(X)=BET(X)*3/2
3690 BET(X)=INT(BET(X)*100+.5)/100
3700 GOSUB 4090
3710 GOTO 3880
3720 IF HO(X)=S THEN 3770
3730 IF HO(X)=K THEN 3770
3740 IF HO(X)=D THEN 3770
3750 GOSUB 3970
3760 GOTO 3880
3770 BET(X)=BET(X)*2/3
3780 BET(X)=INT(BET(X)*100+.5)/100
3790 GOSUB 4090
3800 GOTO 3880
3810 IF HO(X)<>S THEN 3830
3820 IF PA2(X)=K THEN 3850
3830 GOSUB 3970
3840 GOTO 3880
3850 BET(X)=BET(X)*15
3860 BET(X)=INT(BET(X)*100+.5)/100
3870 GOSUB 4090
3880 X=X+1
3890 IF X>5 THEN 3550
3900 IF X>4 THEN 3920
3910 GOTO 3550
3920 GOTO 4130
3930 CALL CLEAR
3940 GOTO 3550
3950 IF X<=8 THEN 3550
3960 GOTO 3930
3970 IF TOT(X)<BET(X)THEN 4020
3980 PRINT "SO SORRY ";NAME$(X);
     " YOU LOSE $";BET(X)::
3990 TOT(X)=TOT(X)-BET(X)
4000 PRINT "YOU NOW HAVE $";
     TOT(X)::
4010 RETURN
4020 TTOT(X)=TOT(X)*-1
4030 PRINT "HEY*";NAME$(X);
     " YOU  LOSE AGAIN *"::
4040 TOT(X)=TOT(X)-BET(X)
4050 TTOT(X)=TOT(X)*-1
4060 PRINT "YOU OWE THE TRACK $";
     TTOT(X)::
4070 PRINT "WE HOPE YOUR CREDIT
     IS GOOD"::
4080 RETURN
```

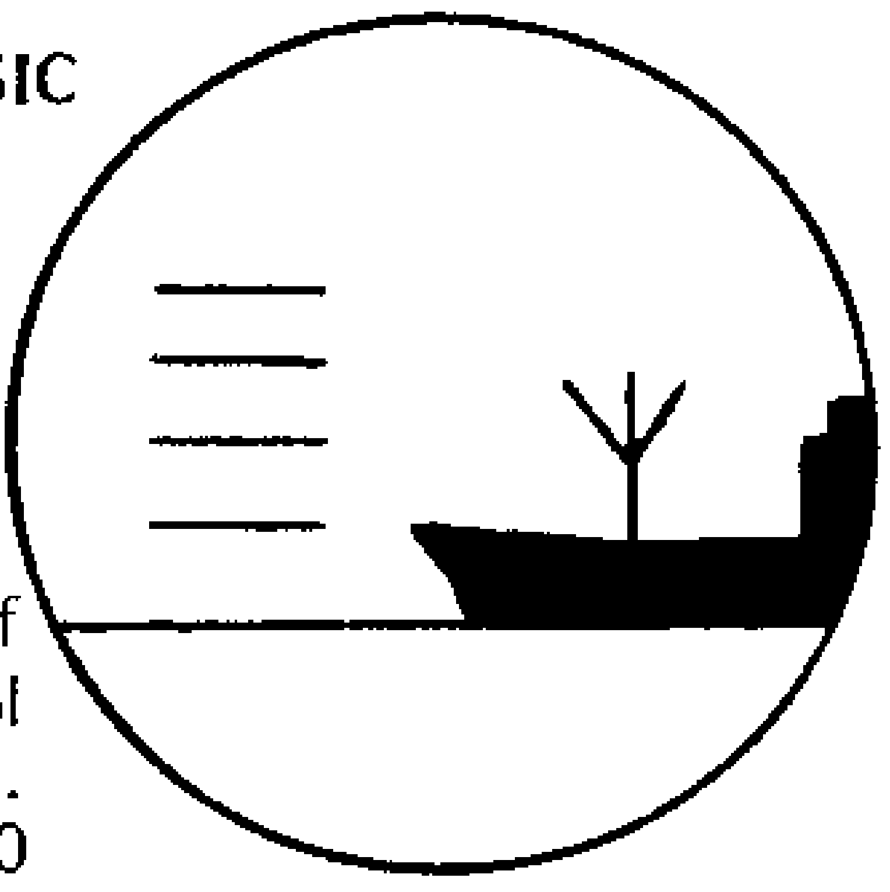
DeBUGS ON DisPLAY

with prior releases. To modify the program so it will work, do the following:

Delete line 980
Add line 1035 ZZ=INT(NH/2 Y0)−2* INT(NH/(2 (Y0+1)))
Change line 1040 IF ZZ=0 AND N=1 THEN NH=NH+ Y0
Change line 1050 IF ZZ=1 AND N=0 THEN NH=NH−2 Y0

Due to this problem with the latest version of Extended BASIC, Texas Instruments has informed us that a new corrected release of the Command Module will be forthcoming.





Spotlight . . . from p. 45

the time, because now and then he hooks the ball or aims at the wrong angles. I therefore threw in some very, very slight random deviations so that a player indeed can become a very good bowler, but will have difficulty playing a 300 game. You may achieve a 300 game, but it is going to be difficult to do it.

The very nature of bowling suggests a multi-player game, so I thought in terms of a league. Up to 8 different players are allowed. The screen actually shows each little player walking up to the lane with the ball in hand and throwing the ball. You can tell which player is up by the color shirt he wears—with 8 different players and 8 different color shirts, and the name of each is displayed on the screen at the time that player is on the lane.

GMK: *Did this particular program present any unusual difficulties with sprites or lead you into areas of programming and simulation where you had not ventured before?*

CME: Yes to both questions. In the sprite area, the obvious difficulty was that you cannot display more than *four* sprites in one horizontal row without losing a sprite. And if you think about the way the pins are set in a bowling alley, you have *five* pins in the back row! Well, I solved that problem—I'd rather not tell how—and in addition, I had to put in a couple *invisible* sprites (one in each gutter) so I could determine when the ball hits a gutter. I therefore really have *seven* sprites in a row even though two are invisible. But having *five visible sprites* was definitely a challenge.

GMK: *You seem to have mastered the use of sprites in simulations. Now that an assembly language is available on the TI-99/4A, do you see this as your next area of challenge?*

CME: Without a doubt. As a matter of fact, it is not only the next area in which we will venture, but is the next area in which we *must* venture. I think that assembler is going to be the language that most of the new games are written in. This is the only way that you can compete with the arcades, so I think Extended BASIC programs and BASIC programs are going to be in much less demand as assembly language programs become more available.

GMK: *But this doesn't preclude the place of these high-level languages in gaming does it?*

CME: Oh, no. Definitely not. The basic system—that is console, monitor, and cassette recorder—is going to be the bread and butter of the industry for a long time. I think there is going to be more of this configuration than any other. But then also, there is going to be a large percentage of more affluent or sophisticated users who are going to demand the assembly language programs and who will purchase the additional peripherals required. We must meet this demand if we want to stay in business.

The choice of language, however, depends on the design of the game. There are some games that are extremely well suited to BASIC. *SAM Defense* is a good example. Other games are perfectly well written in Extended BASIC. And any form of BASIC is certainly a much faster, easier language to master and to develop software in than assembly language.

GMK: *When Futura does its final play-testing—polishing of the game in actual use—do you handle that yourself or release the game to others for critism?*

CME: I don't release the game until I am satisfied with it, but then I have some key individuals—usually the leaders of users' groups throughout the country—who I send a copy to for their criticism and their advice. I have received some very good input many times, and have changed and modified games based on their recommendations. Until I feel the game is fool-proof, bug-free, crash-free, and entertaining, I won't release it.

GMK: *How do you feel about letting the game go after this final play-testing and polishing–putting it out there on the market for all the world to see?*

CME: I feel exactly as I think the expectant mother does peering in the nursery window and wondering how the baby will turn out. Hoping for the best, and full of doubt. You are always thinking that you have the most perfect game until you release it. Then as soon as you start shipping, you start wondering if you could have done better.

GMK: *In the evolution of a game, there's a point beyond which any improvements are only marginal and not even worth the delay in implementing them. How do you know when you reach this point?*

CME: That is really hard to determine. I guess when you're satisfied with the game . . . when you really can't find anything else to do with it that would improve the playability or the entertainment value. It's hard to justify any delay especially when you find people calling you or telling you that the game is great . . . that they enjoy it . . . that it is super the way it is, and you no longer receive any suggestions for improvement. At that time, I think you can put that game to rest and then go on to the next one.

GMK: *After the game is out there in the market, do you ever modify it and release or produce any updates to the game?*

CME: Yes, we do that sometimes. The update for *SAM Defense* comes to mind. We were getting comments from our customers that the game was a little too challenging, and so we decided to offer different levels of challenge. The game was therefore modified as I previously mentioned. We have a standing offer that any user or previous purchaser can send us back their cassette, and for our cost of reproduction and shipping, we will send them the revised version. This has been well received because now people can enjoy a level where they can actually beat the game fairly easily, then advance to a more challenging level, and finally advance to an almost unbeatable game.

GMK: *That appears to be a very sound business practice. Speaking about business, do you have extensive experience with businesses of your own, or was this your first small business venture?*

CME: Well, I did have one other small business venture of my very own. I used to own an ice cream shop. You can learn a lot about human nature . . . learn what people like and what they dislike . . . what bores them, and what they like in flavors. I picked up a lot of insight from that. It might seem hard to believe, but I can apply a lot of that experience to the games I design.

GMK: *I can see your next software advertisement now: "31 Flavors from Futura . . ."*

CME: That's right, because you can't sell vanilla to the people who like strawberry. If they like strawberry, you have to provide strawberry. So we need to provide games for all tastes. "31 Flavor Games"–this is going to be our new motto . . .

GMK: *Do any other family members participate in your business?*

CME: Yes. We have definite team work: I produce the software, and that is my sole responsibility. And then there's Glenda, my wife, a very astute business woman who handles all the business aspects. Finally, of course, our 10 year old daughter is the shipping department. So if any of our customers have found that their order is late or has been bagged or shipped incorrectly, please notify us immediately, and Sybil will get one week's allowance knocked off her salary . . .

Mapping . . . from p. 27

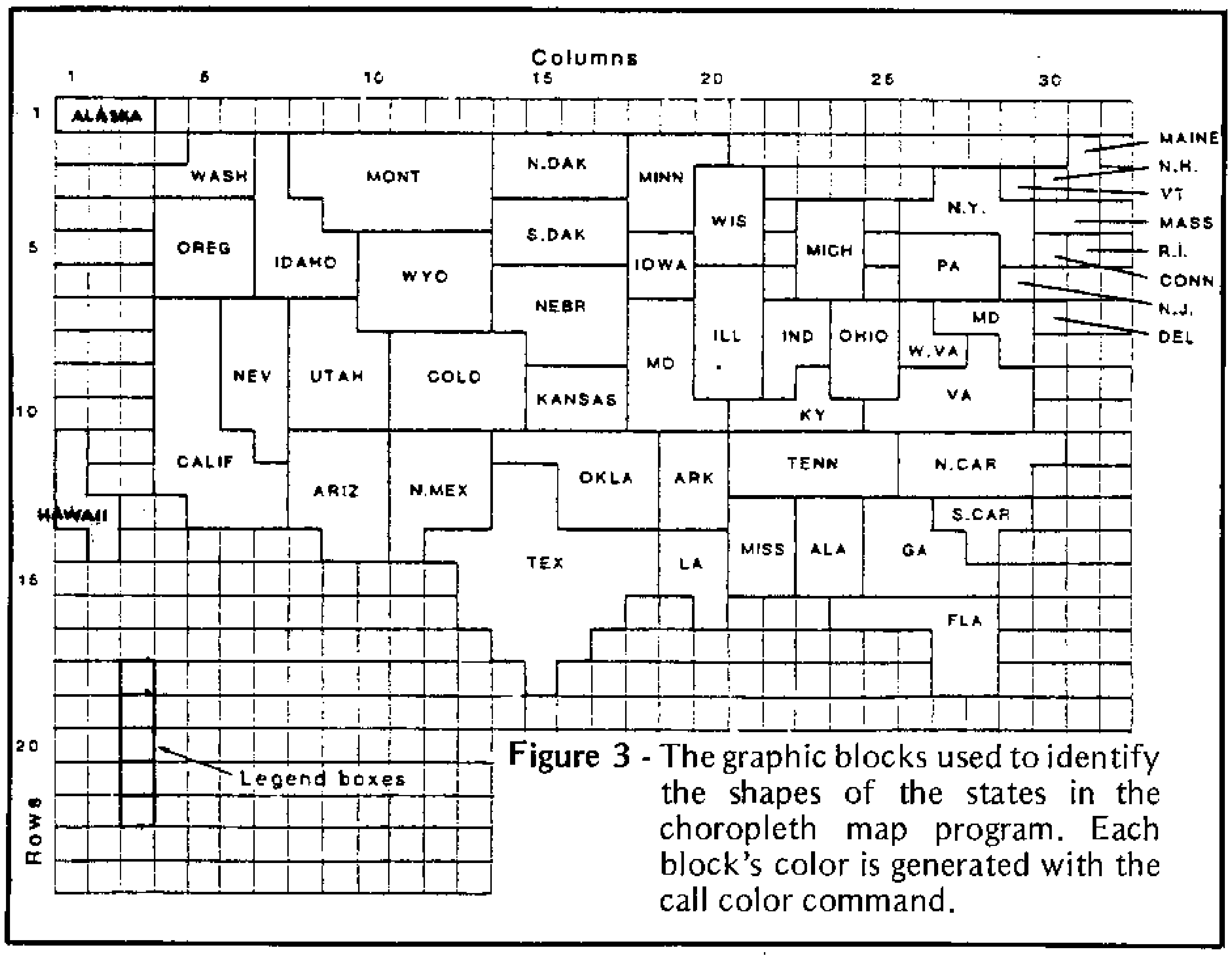


Figure 3 - The graphic blocks used to identify the shapes of the states in the choropleth map program. Each block's color is generated with the call color command.

only utilizes the 32 X 24 resolution screen, and does not develop "refinements" of the shapes of the states that are possible with the CALL CHAR command. The blocks used to identify the states are illustrated in Figure 3. Although only an approximation is achieved with this resolution, the shapes chosen fairly well resemble the individual states, and relative area is similar to real geographical patterns. Other users may wish to modify these if deemed desirable (although I suspect that the 16K RAM will be taxed).

The Choice of Color Symbolization

One standard, acceptable way to symbolize the areas on choropleth maps is to vary the lightness or darkness of one color (as mentioned earlier), in accordance with the values they represent. Classes having higher values are rendered darker, and the lower-valued classes rendered lighter. For this program, the highest class is black, the lowest class white, and the three intermediate classes are in three shades of green or three shades of blue. The TI-99/4 can display 15 different colors, and fortunately there are three different greens and blues—each ranging from light to dark. Symbolizing the color classes in this manner better shows the *total form* of the distribution over the map. The map reader gets a better idea of the continuously changing nature of the *spatial* attributes of the data.

Program Enhancements

A user can make any number of useful changes to this program. You may wish to provide alternate ways of classing the data (i.e., quartiles, equal steps, standard deviations, or others), add new subroutines, or enter your own classes. Another change could be in the color symbolization—with a *different* color for each class. The variable C (1-5) need only be changed to conform to the other color code options used by TI BASIC. As mentioned earlier, with small changes, files containing data sets can be input rather than from the keyboard. This would be especially useful in classroom settings, where census or other data from previous years (and other geographical data) can be compared with present patterns.

Computer-aided instruction (CAI) could also be added to this program—with inquiry questions generated by the spatial distribution seen on the screen. Geographical concepts could be brought out in this manner, and students could easily test hypotheses.

One most intriguing enhancement would be to introduce animation (dynamic cartography) to the program. Various data sets could be read (from files) and displayed in fairly fast sequence to produce a dynamic, *changing* image of the geographical distribution. For example, population density from 1850 to 1980 would show the steady drift of our population from east to west; or a temporal set of sales (or income) performance data would be of interest to marketing analyses. A chief advantage of all computer mapping is in its potential of showing the dynamic qualities of geographical data. This capability is possible on the TI-99/4, extending its versatility even further.

Bibliography

Robinson, Arthur, Sale, Randall, and Morrison, Joel, *Elements of Cartography*, 4th ed., New York, John Wiley and Sons, 1978.

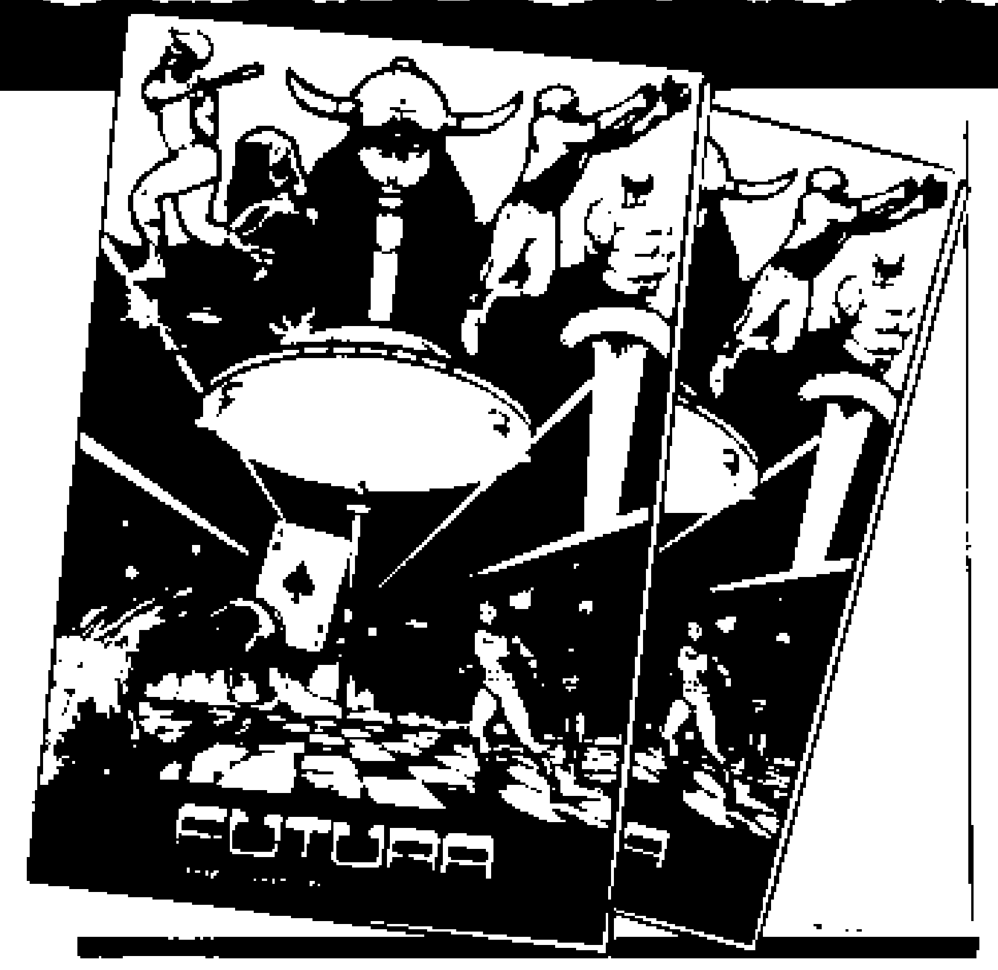
Lawrence, G.R.P., *Cartographic Methods*, 2nd ed., New York, Methuen and Co., Ltd., 1979.

Laboratory for Computer Graphics and Spatial Analysis, Harvard Graduate School of Design, various publications, Cambridge, Massachusetts 02138.

```
100 REM  ******************
110 REM  * CHOROPLETH MAP *
120 REM  ******************
130 REM  BY BORDEN D. DENT
140 REM  REVISED BY C. WHITELAW
150 REM  99'ER VERSION 1.6.1
160 DIM SN$(50),V(50),VV(50)
170 CALL CLEAR
180 CALL SCREEN(12)
190 PRINT TAB(12);"UNITED"
200 PRINT TAB(12);"STATES"
210 PRINT ::::TAB(8);
    "CHOROPLETH MAP"::::::::::
220 CALL COLOR(9,5,5)
230 CALL VCHAR(3,4,96,17)
240 CALL HCHAR(3,5,96,25)
250 CALL HCHAR(19;5,96,25)
260 CALL VCHAR(4,29,96,15)
270 RESTORE 280
280 DATA ALABAMA,ALASKA,ARIZONA,
    ARKANSAS,CALIFORNIA,COLORADO,
    CONNECTICUT,DELAWARE.FLORIDA,
    GEORGIA,HAWAII,IDAHO
290 DATA ILLINOIS,INDIANA.IOWA,
    KANSAS,KENTUCKY,LOUISIANA,
    MAINE,MARYLAND,MASSACHUSETTS,
    MICHIGAN,MINNESOTA
300 DATA MISSISSIPPI,MISSOURI,
    MONTANA,NEBRASKA,NEVADA,NEW
    HAMPSHIRE,NEW JERSEY,N EW
    MEXICO,NEW YORK
310 DATA NORTH CAROLINA,NORTH
    DAKOTA,OHIO,OKLAHOMA,OREGON,
    PENNSYLVANIA,RHODE ISL AND,
    SOUTH CAROLINA,SOUTH DAKOTA
320 DATA TENNESSEE,TEXAS.UTAH,
    VERMONT,VIRGINIA,WASHINGTON,
    WEST VIRGINIA.WISCONSI N,WYOMING
330 FOR I=1 TO 50
340 READ SN$(I)
350 NEXT I
360 CALL CLEAR
370 PRINT TAB(6);
    "PROGRAM INSTRUCTIONS"
380 PRINT TAB(14);"***"
390 PRINT :"CHOOSE MAP
    BACKGROUND COLOR"::
400 PRINT :TAB(8);"1-MEDIUM RED"
410 PRINT :TAB(8);"2-LIGHT RED"
420 PRINT :TAB(8);"3-DARK YELLOW"
430 PRINT :TAB(8);"4-LIGHT YELLOW"
440 PRINT :TAB(8);"5-GREY"::::
450 CALL KEY(0,KEY,ST)
460 IF (KEY<49)+(KEY>53)=-1 THEN 450
470 IF KEY<>53 THEN 500
480 SC=15
490 GOTO 510
500 SC=KEY-40
510 CALL CLEAR
520 PRINT "CHOOSE MAP COLOR"
530 PRINT ::"1-BLUE"::"2-GREEN"::::
540 CALL KEY(0,KEY,ST)
550 IF KEY=49 THEN 610
560 IF KEY<>50 THEN 540
570 C(2)=4
580 C(3)=3
590 C(4)=13.
600 GOTO 640
610 C(2)=8
620 C(3)=6
630 C(4)=5
640 C(1)=16
650 C(5)=2
660 CALL CLEAR
670 PRINT "DATA ENTRY INSTRUCTIONS"
680 PRINT :"PLEASE ENTER THE VALUES"
690 PRINT "FOR EACH STATE."
700 PRINT :"YOU MAY ENTER A VALUE"
    :"UP TO 8 DIGITS."
710 PRINT "DO NOT USE COMMAS."::
720 FOR I=1 TO 50
730 INPUT SN$(I)&" = ":V(I)
740 VV(I)=V(I)
750 PRINT
760 NEXT I
770 CALL CLEAR
780 PRINT TAB(6);"WHAT IS THE TITLE"
790 PRINT TAB(6);"OF YOUR MAP?"
800 PRINT
810 PRINT TAB(6);"BECAUSE OF SPACE"
820 PRINT TAB(6);"LIMITATIONS, KEEP"
830 PRINT TAB(6);"YOUR TITLE TO LESS"
840 PRINT TAB(6);"THAN 14 SPACES."::
850 INPUT "TITLE ":TT$
860 CALL CLEAR
870 PRINT TAB(6);"ONE MOMENT PLEASE"
880 PRINT :TAB(9);"* SORTING *"::
890 REM  SORT SUBROUTINE
900 FOR N=1 TO 49
910 FOR LT=N+1 TO 50
920 IF VV(N)<=VV(LT)THEN 960
930 LET W=VV(N)
940 LET VV(N)=VV(LT)
950 LET VV(LT)=W
960 NEXT LT
970 NEXT N
980 REM  CLASS LIMITS SUBROUTINE
990 X1=VV(10)+(VV(11)-VV(10))/2
1000 X2=VV(20)+(VV(21)-VV(20))/2
1010 X3=VV(30)+(VV(31)-VV(30))/2
1020 X4=VV(40)+(VV(41)-VV(40))/2
1030 REM  PRINT LEGEND,TITLE
1040 CALL CLEAR
1050 PRINT TAB(2);VV(1);"-";X1
1060 PRINT TAB(2);X1;"-";X2
1070 PRINT TAB(2);X2;"-";X3;TT$
1080 PRINT TAB(2);X3;"-";X4
1090 PRINT TAB(2);X4;"-";VV(50)
1100 REM  ALL STATE PLOTS
1110 K(1)=9
1120 K(2)=10
1130 K(3)=11
1140 K(4)=12
1150 K(5)=13
1160 S(1)=96
1170 S(2)=104
1180 S(3)=112
1190 S(4)=120
1200 S(5)=128
1210 FOR T=1 TO 5
1220 CALL COLOR(K(T),C(T),C(T))
1230 NEXT T
1240 CALL SCREEN(SC)
1250 CALL HCHAR(19,4,96)
1260 CALL HCHAR(20,4,104)
1270 CALL HCHAR(21,4,112)
1280 CALL HCHAR(22,4,120)
1290 CALL HCHAR(23,4,128)
1300 REM  ALABAMA
1310 NN=1
1320 GO SUB 3960
1330 CALL VCHAR(13,23,S(T),3)
1340 CALL VCHAR(13,24,S(T),3)
1350 REM  ALASKA
1360 NN=2
1370 GOSUB 3960
1380 CALL HCHAR(1,1,S(T),3)
1390 REM  ARIZ
1400 NN=3
1410 GOSUB 3960
1420 CALL VCHAR(11,8,S(T),3)
1430 CALL VCHAR(11,9,S(T),4)
1440 CALL VCHAR(11,10,S(T),4)
1450 REM  ARK
1460 NN=4
1470 GOSUB 3960
1480 CALL VCHAR(11,19,S(T),3)
1490 CALL VCHAR(11,20,S(T),3)
1500 REM  CALIF
1510 NN=5
1520 GOSUB 3960
1530 CALL VCHAR(7,4,S(T),6)
1540 CALL VCHAR(7,5,S(T),7)
1550 CALL VCHAR(11,6,S(T),3)
1560 CALL VCHAR(12,7,S(T),2)
1570 REM  COLO
1580 NN=6
1590 GOSUB 3960
1600 CALL HCHAR(8,11,S(T),4)
1610 CALL HCHAR(9,11,S(T),4)
1620 CALL HCHAR(10,11,S(T),4)
1630 REM  CONN
1640 NN=7
1650 GOSUB 3960
1660 CALL HCHAR(5,30,S(T))
1670 REM  DEL
1680 NN=8
1690 GOSUB 3960
1700 CALL HCHAR(7,30,S(T))
1710 REM  FLA
1720 NN=9
1730 GOSUB 3960
1740 CALL HCHAR(16,24,S(T),5)
1750 CALL HCHAR(17,27,S(T),2)
1760 CALL HCHAR(18,27,S(T),2)
1770 REM  GA
1780 NN=10
1790 GOSUB 3960
1800 CALL HCHAR(13,25,S(T),2)
1810 CALL HCHAR(14,25,S(T),3)
1820 CALL HCHAR(15,25,S(T),3)
1830 REM  HAWAII
1840 NN=11
1850 GOSUB 3960
1860 CALL VCHAR(11,1,S(T),3)
1870 CALL VCHAR(13,2,S(T),3)
1880 REM  IDAHO
1890 NN=12
1900 GOSUB 3960
1910 CALL VCHAR(2,7,S(T),5)
1920 CALL VCHAR(4,8,S(T),3)
1930 CALL VCHAR(5,9,S(T),2)
1940 REM  ILL
1950 NN=13
1960 GOSUB 3960
1970 CALL VCHAR(6,20,S(T),4)
1980 CALL VCHAR(6,21,S(T),4)
1990 REM  IND
2000 NN=14
2010 GOSUB 3960
2020 CALL VCHAR(7,22,S(T),3)
2030 CALL VCHAR(7,23,S(T),2)
2040 REM  IOWA
2050 NN=15
2060 GOSUB 3960
2070 CALL HCHAR(5,18,S(T),2)
2080 CALL HCHAR(6,18,S(T),2)
2090 REM  KAN
2100 NN=16
2110 GOSUB 3960
2120 CALL HCHAR(9,15,S(T),3)
2130 CALL HCHAR(10,15,S(T),3)
2140 REM  KY
2150 NN=17
2160 GO SUB 3960
2170 CALL HCHAR(9,23,S(T))
2180 CALL HCHAR(10,21,S(T),4)
2190 REM  LA
2200 NN=18
2210 GO SUB 3960
2220 CALL VCHAR(14,19,S(T),2)
2230 CALL VCHAR(14,20,S(T),3)
2240 REM  MAINE
2250 NN=19
2260 GOSUB 3960
2270 CALL VCHAR(2,31,S(T),2)
2280 REM  MD
2290 NN=20
2300 GOSUB 3960
2310 CALL HCHAR(7,27,S(T),3)
2320 CALL HCHAR(8,29,S(T))
2330 REM  MASS
2340 NN=21
2350 GOSUB 3960
2360 CALL HCHAR(4,30,S(T),2)
2370 REM  MICH
2380 NN=22
2390 GOSUB 3960
2400 CALL VCHAR(4,23,S(T),3)
2410 CALL VCHAR(4,24,S(T),3)
2420 REM  MINN
2430 NN=23
2440 GOSUB 3960
2450 CALL HCHAR(2,18,S(T),3)
2460 CALL HCHAR(3,18,S(T),2)
2470 CALL HCHAR(4,18,S(T),2)
2480 REM  MISS
2490 NN=24
2500 GOSUB 3960
```

Continued on p. 56

See article this mag, pg 3



Color Mapping

```
2510 CALL VCHAR(13,21,S(T),3)
2520 CALL VCHAR(13,22,S(T),3)
2530 REM  MD
2540 NN=25
2550 GOSUB 3960
2560 CALL VCHAR(7,18,S(T),4)
2570 CALL VCHAR(7,19,S(T),4)
2580 CALL VCHAR(10,20,S(T))
2590 REM  MONT
2600 NN=26
2610 GOSUB 3960
2620 CALL HCHAR(2,8,S(T),6)
2630 CALL HCHAR(3,8,S(T),6)
2640 CALL HCHAR(4,9,S(T),6)
2650 REM  NEBR
2660 NN=27
2670 GOSUB 3960
2680 CALL HCHAR(6,14,S(T),4)
2690 CALL HCHAR(7,14,S(T),4)
2700 CALL HCHAR(8,15,S(T),3)
2710 REM  NEV
2720 NN=28
2730 GOSUB 3960
2740 CALL VCHAR(7,6,S(T),4)
2750 CALL VCHAR(7,7,S(T),5)
2760 REM  NH
2770 NN=29
2780 GOSUB 3960
2790 CALL HCHAR(3,29,S(T))
2800 REM  NJ
2810 NN=30
2820 GOSUB 3960
2830 CALL HCHAR(6,29,S(T))
2840 REM  N MEX
2850 NN=31
2860 GOSUB 3960
2870 CALL VCHAR(11,11,S(T),4)
2880 CALL VCHAR(11,12,S(T),3)
2890 CALL VCHAR(11,13,S(T),3)
2900 REM  N YORK
2910 NN=32
2920 GOSUB 3960
2930 CALL HCHAR(3,27,S(T),2)
2940 CALL HCHAR(4,26,S(T),4)
2950 CALL HCHAR(5,29,S(T))
2960 REM  NC
2970 NN=33
2980 GOSUB 3960
2990 CALL HCHAR(11,26,S(T),5)
3000 CALL HCHAR(12,26,S(T),4)
3010 REM  N DAK
3020 NN=34
3030 GOSUB 3960
3040 CALL HCHAR(2,14,S(T),4)
3050 CALL HCHAR(3,14,S(T),4)
3060 REM  OHIO
3070 NN=35
3080 GOSUB 3960
3090 CALL VCHAR(7,24,S(T),3)
3100 CALL VCHAR(7,25,S(T),3)
3110 REM  OKLA
3120 NN=36
3130 GOSUB 3960
3140 CALL HCHAR(11,14,S(T),5)
3150 CALL HCHAR(12,16,S(T),3)
3160 CALL HCHAR(13,16,S(T),3)
3170 REM  ORE
3180 NN=37
3190 GOSUB 3960
3200 CALL HCHAR(4,4,S(T),3)
3210 CALL HCHAR(5,4,S(T),3)
3220 CALL HCHAR(6,4,S(T),3)
3230 REM  PA
3240 NN=38
3250 GOSUB 3960
3260 CALL HCHAR(5,26,S(T),3)
3270 CALL HCHAR(6,26,S(T),3)
3280 REM  RI
3290 NN=39
3300 GOSUB 3960
3310 CALL HCHAR(5,31,S(T))
3320 REM  S CAR
3330 NN=40
3340 GO SUB 3960
3350 CALL HCHAR(13,27,S(T),3)
3360 CALL HCHAR(14,28,S(T))
3370 REM  S DAK
3380 NN=41
3390 GOSUB 3960
3400 CALL HCHAR(4,14,S(T),4)
3410 CALL HCHAR(5,14,S(T),4)
3420 REM  TENN
3430 NN=42
3440 GOSUB 3960
3450 CALL HCHAR(11,21,S(T),5)
3460 CALL HCHAR(12,21,S(T),5)
3470 REM  TEX
3480 NN=43
3490 GOSUB 3960
3500 CALL HCHAR(12,14,S(T),2)
3510 CALL HCHAR(13,14,S(T),2)
3520 CALL HCHAR(14,12,S(T),7)
3530 CALL HCHAR(15,13,S(T),6)
3540 CALL HCHAR(16,13,S(T),5)
3550 CALL HCHAR(17,14,S(T),3)
3560 CALL HCHAR(18,15,S(T))
3570 REM  UTAH
3580 NN=44
3590 GOSUB 3960
3600 CALL VCHAR(7,8,S(T),4)
3610 CALL VCHAR(7,9,S(T),4)
3620 CALL VCHAR(8,10,S(T),3)
3630 REM  VERMONT
3640 NN=45
3650 GOSUB 3960
3660 CALL HCHAR(3,30,S(T))
3670 REM  VA
3680 NN=46
3690 GOSUB 3960
3700 CALL HCHAR(8,28,S(T))
3710 CALL HCHAR(9,26,S(T),4)
3720 CALL HCHAR(10,25,S(T),5)
3730 REM  WASH
3740 NN=47
3750 GOSUB 3960
3760 CALL HCHAR(2,5,S(T),2)
3770 CALL HCHAR(3,4,S(T),3)
3780 REM  W VA
3790 NN=48
3800 GO SUB 3960
3810 CALL HCHAR(7,26,S(T))
3820 CALL HCHAR(8,26,S(T),2)
3830 REM  WISC
3840 NN=49
3850 GOSUB 3960
3860 CALL VCHAR(3,20,S(T),3)
3870 CALL VCHAR(3,21,S(T),3)
3880 REM  WYO
3890 NN=50
3900 GOSUB 3960
3910 CALL HCHAR(5,10,S(T),4)
3920 CALL HCHAR(6,10,S(T),4)
3930 CALL HCHAR(7,10,S(T),4)
3940 GOTO 3940
3950 REM  CLASS CHECK
3960 IF V(NN)<=X1 THEN 4020
3970 IF V(NN)<=X2 THEN 4040
3980 IF V(NN)<=X3 THEN 4060
3990 IF V(NN)<=X4 THEN 4080
4000 T=5
4010 RETURN
4020 T=1
4030 RETURN
4040 T=2
4050 RETURN
4060 T=3
4070 RETURN
4080 T=4
4090 RETURN
4100 END
```

Verbose . . . from p. 28

Listing 1 continued

```
440 DATA 218,196,26,103,157,119,235,83,133,156
450 DATA 233,220,113,110,117,170,88,51,77,58
460 DATA 238,169,211,240,100,207,186,167,201,69
470 DATA 196,162,42,205,46,245,41,179,68,87
480 DATA 97,51,24,105,146,233,22,0,64,1
490 DATA 93,121,60
```

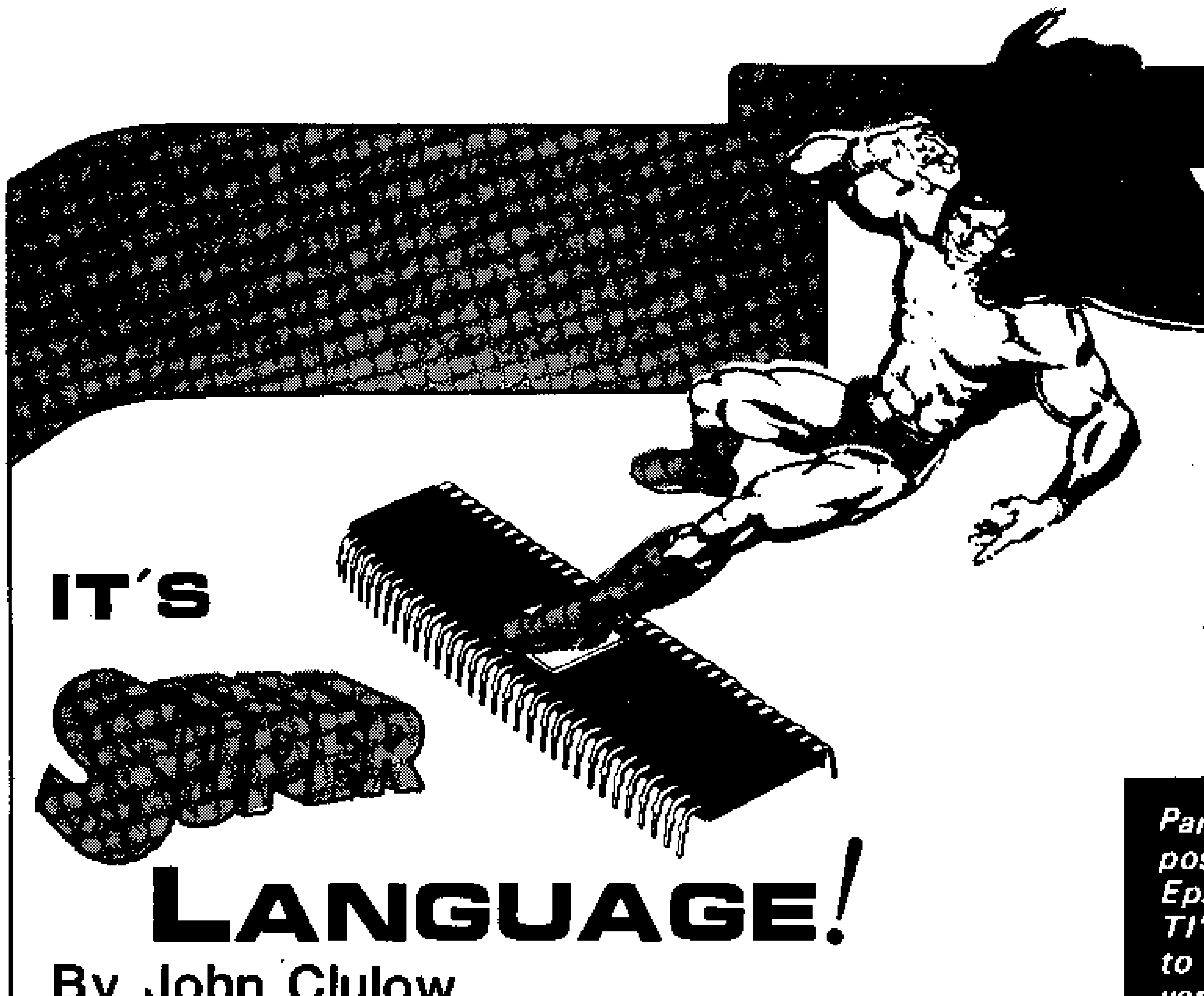
Listing 2

```
THE WORD IS ** REWRITE **
LENGTH = 133 BYTES

DATA 96,0,42,161,19,49,92,60,149,149
DATA 78,86,51,117,147,223,26,61,196,197
DATA 69,253,170,93,103,231,176,108,167,10
DATA 158,83,211,151,156,188,40,21,157,106
DATA 180,178,42,89,125,96,0,85,162,101
DATA 33,221,57,28,139,154,142,144,176,116
DATA 172,106,58,92,162,67,137,105,248,82
DATA 142,49,39,169,209,7,179,84,220,175
DATA 218,196,26,103,157,119,235,83,133,156
DATA 233,220,113,110,117,170,88,51,77,58
DATA 238,169,211,240,100,207,186,167,201,69
DATA 196,162,42,205,46,245,41,179,68,87
DATA 97,51,24,105,146,233,22,0,64,1
DATA 93,121,60
```

Listing 3

```
100 REM +++++++++++++++++++++++++
110 REM +  SPELLING  TEST  GAME +
120 REM +           BY          +
130 REM +     DAVID G. BRADER   +
140 REM +   99'ER VERSION 1.6.1 +
150 REM +++++++++++++++++++++++++
160 REM  USES "DSK1.WORDS" FILE AS SOURCE OF WORDS TO
170 REM      GUESS IN GAME.
180 DIM WORD$(20),F$(20)
190 CALL CLEAR
200 PRINT "PUT DISK WITH ""WORDS"" FILE   IN DRIVE ONE"
210 INPUT "PRESS ENTER WHEN READY ":X$
220 OPEN #1:"DSK1.WORDS",INTERNAL,INPUT ,
    VARIABLE 254
230 FOR I=1 TO 20
240 IF EOF(1)<>0 THEN 300
250 INPUT #1:WORD$(I)
260 INPUT #1:F$(I)
270 NEXT I
280 LAST=I
290 GOTO 310
300 LAST=I-1
310 CLOSE #1
320 REM
330 CALL CLEAR
340 SCORE=0
350 PRINT "THERE ARE ";LAST;"WORDS": : :
360 PRINT "SEE IF YOU CAN SPELL THEM
    ALL CORRECTLY.  GOOD LUCK!"
370 FOR M=1 TO 700
380 NEXT M
390 FOR J=1 TO LAST
```

Continued on p. 86



MAGIC CRAYON

Learning Assembly Language The Hard Way

Part 2 of Patricia Swift's article, A Screen Printing Utility, *has been postponed until next issue because of the CES announcement that the Epson MX-80 printer (with dot-addressable graphics installed) would be TI's new 99/4 matrix printer. Extra article development time is needed to ensure screen-dump compatibility with both the MX-100 and TI's version of the MX-80. We think you'll learn quite a lot from this issue's substitute article in the interim.*

By John Clulow
Technical Editor

Like many other 99'ers, I was anxious to receive the long awaited Editor/Assembler package. When it finally arrived, I remember the excitement of unwrapping the 470 page manual—and the sinking feeling when I read, "This manual assumes that you already know a programming language, preferably an assembly language."

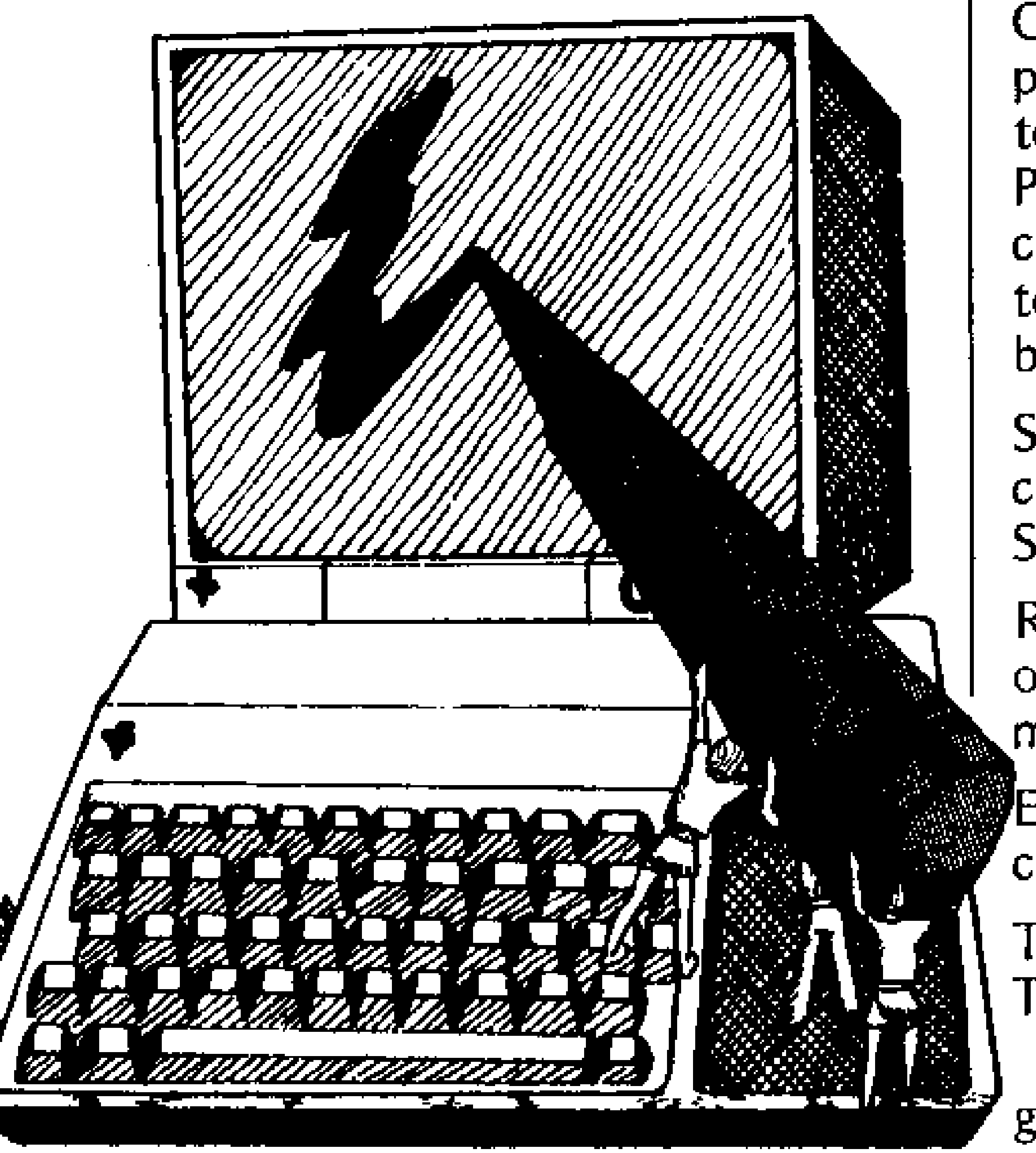
My anxiety grew as I thumbed through it—there were no pictures, cartoons, or fill-in-the-blank examples. It did say, "There are many fine books available which teach the basics of assembly language." So I called the local computer stores. The only books they were aware of, however, also assumed familiarity with the basics.

I guess I had some fuzzy ideas about assembly language in the back of my mind; it was qualitatively different from higher level languages, requiring an in-depth knowledge of digital electronics and a capacity for the most detailed sort of logico-mathematical thought. In short—nothing seemed more difficult . . .

But my experience thus far seemed to confirm my worst fear. Learning assembly language presumed a prior knowledge of assembly language; it was not merely difficult, it was *impossible*. After running *Tombstone City* a few times and typing in Pat Swift's *Life* program (in Vol. 1, No. 4), I put the Editor/Assembler on a shelf thinking maybe I'd learn about it gradually over the next year or two.

It would still be there gathering dust were it not for a back injury that kept me flat on the floor, unable to do anything *except* read the manual. I was surprised to discover that writing an assembly language program is similar to, and in some respects simpler than writing a program in BASIC. A new programming context or conceptual model is required. But to get started, I found that this picture could be primitive, containing many over-simplifications and approximations.

The picture I developed enabled me to successfully formulate and execute a simple programming objective. The program and associated underlying concepts are presented here to facilitate the learning process for others who, like me, find it hard to overcome preconceived notions about how difficult assembly language is. The program should not be taken as a model of exemplary programming technique; at this point my conception of "good programming" is programming that works . . . period. You will undoubtedly be able to find ways to improve this one—to make it work faster and utilize memory more efficiently—and in so doing, further develop the concepts presented.



In assembly language, four video display modes are available: Graphics (or Pattern) Mode, Text Mode, Bit-Map Mode (99/4A only), and Multicolor Mode. In Multicolor Mode, the screen is divided into a grid of 64 x 48, with each box measuring 4 pixels on a side. Each box can have a color assigned to it. The program allows use of a joystick to move a flashing cursor on the screen. Whenever the fire button is depressed, the cursor leaves a trail of small, colored boxes. The following single key commands are available:

C – *Change Color*. Displays a color palette and pointer. Move the pointer to the desired color with the joystick. Press the fire button to make that the color of the boxes, or press the C key to make it the color of the screen background.

S – *Save Screen*. Saves the current contents of the screen as "DSK1. SCREEN".

R – *Recall Screen*. Loads the contents of "DSK1.SCREEN" for subsequent modification.

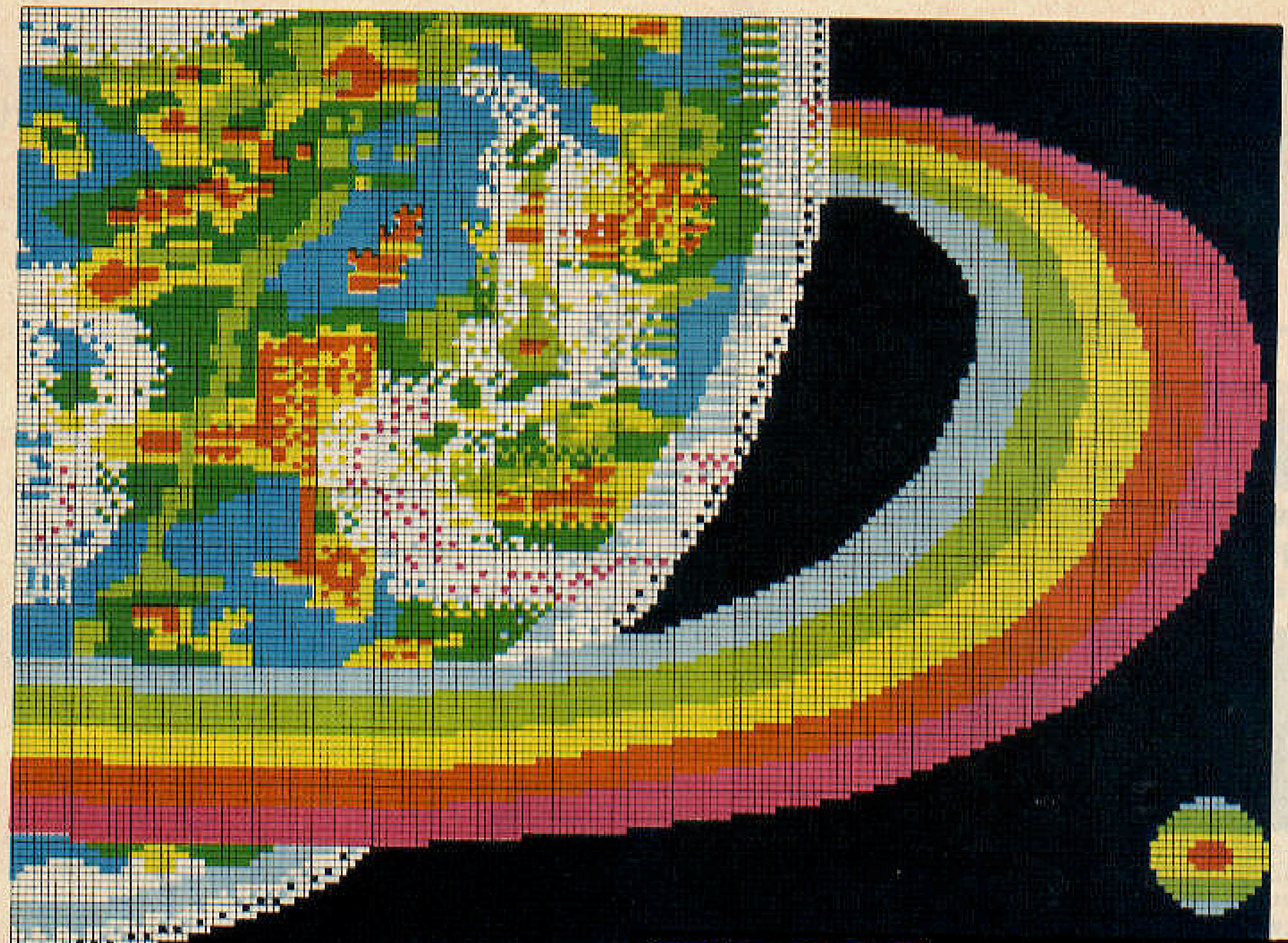
E – *Erase Screen*. Erases the screen contents.

T – *Terminate*. Returns to the Master Title Screen.

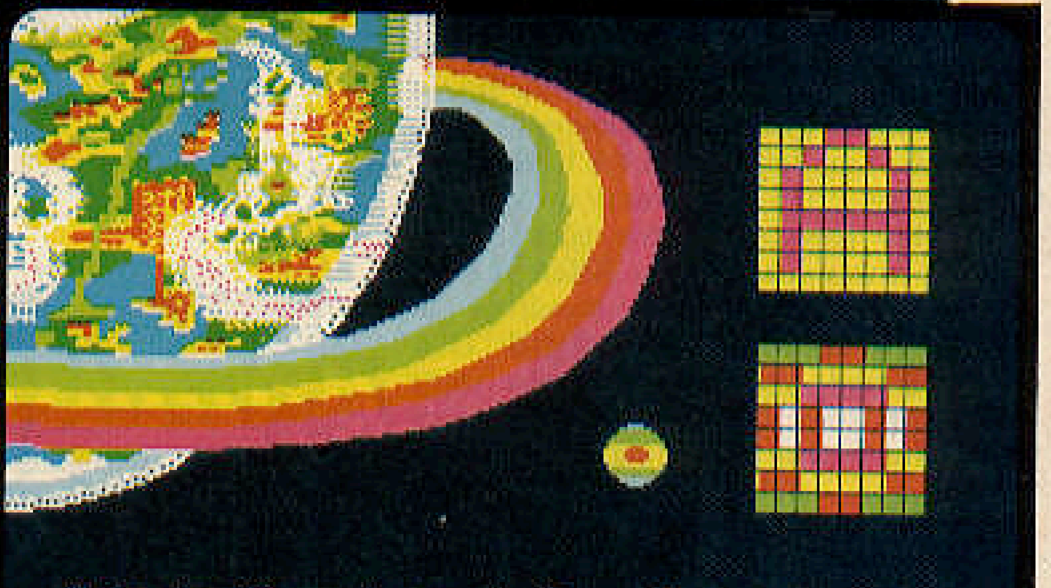
In order to understand how the program works, it will be helpful to differentiate two systems. You probably know that the Central Processing Unit (CPU) in the Home Computer is the TMS9900. It has three built-in, 16-bit "hardware" registers (the Program Counter, Workspace Pointer, and Status Register) and makes use of sixteen workspace registers located in read-write memory. Because these 16-bit workspace registers are not located on the chip, they are called "software" registers. The CPU can directly address the read-write memory (RAM) in the Memory Expansion Unit and CPU scratch pad, as well as ROM in the console, Command Modules, and various peripherals. However, it cannot directly address the 16K of RAM built into the console.

That 16K RAM block is addressed by another microprocessor—the TMS9918

Continued on p. 60



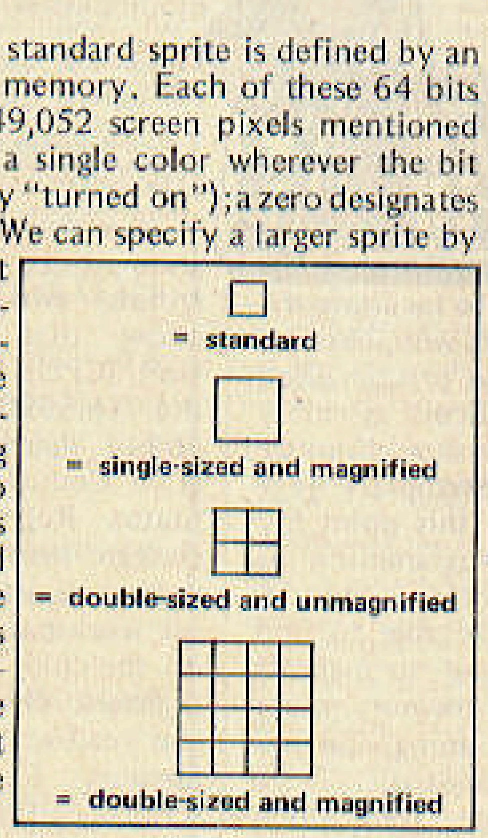
This high-resolution scene is shown over-sized to illustrate the color combination possibilities in Bit-Map Mode. Notice the color blending and shadings that the eye perceives when viewing the *same* scene on the screen-size display of the lower figure.



Notice the difference between the Pattern Mode in the top 8 x 8 pixel square which is limited to just 2 colors (foreground and background), and the Bit-Map Mode of the bottom 8 x 8 pixel square which is allowed 2 unique colors in *each* horizontal 8-pixel row.

3-D . . . from p. 39

The shape of a regular or standard sprite is defined by an 8 x 8 bit pattern stored in memory. Each of these 64 bits corresponds to one of the 49,052 screen pixels mentioned previously—with each being a single color wherever the bit pattern contains a 1 (is thereby "turned on"); a zero designates transparency ("turned off"). We can specify a larger sprite by either (a) using a 16 x 16 bit pattern ("a double-sized" unmagnified sprite), (b) magnifying the existing sprite by a factor of four, ("single-sized magnified"), or (c) using both techniques together to create a sprite sixteen times normal size ("double-sized and magnified"). This size feature allows screen objects to grow and shrink at will—with virtually *none* of the programming effort that would be required in more conventional VDP systems.


= standard
= single-sized and magnified
= double-sized and unmagnified
= double-sized and magnified

Each sprite carries four attributes: the first two specify its horizontal and vertical position; the third defines its shape "name" (according to the bit-pattern concept described above); and the fourth specifies its color. Moving a sprite is simply a matter of changing its position indicators; it will continue moving smoothly on its own. The high-speed, smooth motion of a sprite compared with a conventional moving-graphic element is due to the smaller, more precise "steps" (higher resolution) that the sprite can take while moving. Animated secondary motion—for example, rotating wheels or an asteroid tumbling through space—is achieved by defining ("naming") several similar looking sprites in different secondary postions (e.g., states of rotation), and then swapping the sprite names as what appears to be a *single* sprite smoothly moves across the screen.

The TMS9918A VDP chip has four modes of operation: (1) Graphics 1 or Pattern Mode, (2) Graphics 2 or Bit-Map Mode, (3) Text Mode, and (4) Multicolor Mode. The Pattern Mode consists of a 32-column by 24-row grid of 8 x 8 pixels in each 2-color grid square. Bit-Map Mode (shown in the above figures) allows *each* of the 8 horizontal rows within an

Pros . . . from p. 43

```
60 REM   **    CHUCK-A-LUCK    **
70 REM   *       TI BASIC       *
80 REM   *     .BY SAM PINCUS   *
90 REM  * 99'ER VERSION 1.6.1 *
100 DIM DICE_VALUE(3),
    PLAYER_NAME$(4),
    PLAYER_CASH(4),PLAYER_BET(4),
    PLAYER_DICE(4)
110 DIM DICE_PIP(9,9),LOC_X(27),
    LOC_Y(27)
160 GOSUB 20000
170 REM  BETTING LOOP
200 REM  GET BET
210 GOSUB 1200
220 REM  THROW DICE
230 GOSUB 2000
240 REM  UPDATE CASH BALANCE
250 FOR I=1 TO PLAYERS
260 IF PLAYER_CASH(I)=0 THEN 760
280 PRINT "":PLAYER_NAME$(I);",
    YOU BET ON";PLAYER_DICE(I);
    "FOR";PLAYER_BET(I);"DOLLAR";
290 IF PLAYER_BET(I)<2 THEN 310
300 PRINT "S";
310 PRINT "."
520 WIN=0
530 FOR J=1 TO 3
540 IF PLAYER_DICE(I)<>DICE_VALUE(J)
    THEN 560
550 WIN=WIN+1
560 NEXT J
570 IF WIN=0 THEN 690
580 WIN=WIN*PLAYER_BET(I)
590 PRINT "YOU ";"WIN";WIN;"DOLLAR";
600 IF WIN<2 THEN 620
610 PRINT "S";
620 PRINT "."
630 PLAYER_CASH(I)
    =PLAYER_CASH(I)+WIN
640 PRINT "YOU NOW HAVE";
    PLAYER_CASH(I);"DOLLAR";
650 IF PLAYER_CASH(I)<2 THEN 670
660 PRINT "S";
670 PRINT "."
680 GOTO 760
690 PRINT "YOU LOST";PLAYER_BET(I);
    "DOLLAR";
700 IF PLAYER_BET(I)<2 THEN 720
710 PRINT "S";
720 PRINT "."
730 PLAYER_CASH(I)=PLAYER_CASH
    (I)-PLAYER_BET(I)
740 IF PLAYER_CASH(I)>0 THEN 640
750 PRINT "YOU ARE BANKRUPT!"
760 NEXT I
770 REM  CHECK FOR END OF GAME
780 GOSUB 5000
790 IF NO_LEFT>1 THEN 970
800 INPUT "WANT TO PLAY AGAIN
    (Y/N)?":A$
810 A$=SEG$(A$,1,1)
820 IF A$<>"Y" THEN 850
830 GOSUB 22000
840 GOTO 200
850 IF A$<>"N" THEN 880
860 PRINT "THANK YOU FOR PLAYING.":
    "":""
870 STOP
880 PRINT PL$
890 GOTO 800
970 FOR I=1 TO 600
980 NEXT I
990 GOTO 200
1200 CALL CLEAR
1210 FOR I=1 TO PLAYERS
1220 IF PLAYER_CASH(I)=0 THEN 1500
1230 ON INT(RND*4+1)GOTO 1240,1260,
     1280,1300
1240 PRINT "NOW, ";
1250 GOTO 1350
1260 PRINT "OK, ";
1270 GOTO 1350
1280 PRINT "ALRIGHT, ";
1290 GOTO 1350
1300 PRINT "YOUR TURN, ";
1350 PRINT PLAYER_NAME$(I);","
1360 PRINT "YOU HAVE";PLAYER_CASH(I);
     "DOLLAR";
1370 IF PLAYER_CASH(I)<2 THEN 1390
1380 PRINT "S";
1390 PRINT ".":"WHAT'S YOUR BET? "
1400 INPUT PLAYER_BET(I)
1410 IF PLAYER_BET(I)<1 THEN 1450
1420 IF PLAYER_BET(I)>PLAYER_CASH(I)
     THEN 1450
1430 IF PLAYER_BET(I)>50 THEN 1450
1440 IF INT(PLAYER_BET(I))
     =PLAYER_BET(I)THEN 1470
1450 PRINT "THAT'S NOT POSSIBLE."
1460 GOTO 1230
1470 PRINT "WHAT NUMBER WILL YOU
     BET ON?"
1480 INPUT PLAYER_DICE(I)
1490 IF INT(PLAYER_DICE(I))
     <>PLAYER_DICE(I)THEN 1520
1500 IF PLAYER_DICE(I)<1 THEN 1520
1510 IF PLAYER_DICE(I)<7 THEN 1540
1520 PRINT "TRY AGAIN."
1530 GOTO 1470
1540 NEXT I
1550 RETURN
2000 REM
2010 CALL CLEAR
2020 CALL SCREEN(10)
2030 FOR I=1 TO PLAYERS
2035 GOSUB 28000
2040 ROW=(I-1)*5+1
2050 COL=15
2060 MSG$=PLAYER_NAME$(I)
2070 GOSUB 4900
2100 ROW=ROW+1
2110 COL=15
2120 MSG$="BET"
2130 GOSUB 4900
2150 COL=20
2160 MSG$="$"&STR$(PLAYER_BET(I))
2170 GOSUB 4900
2200 ROW=ROW+1
2210 COL=15
2220 MSG$="CASH"
2230 GOSUB 4900
2250 COL=20
2260 MSG$="$"&STR$(PLAYER_CASH(I))
2270 GOSUB 4900
2300 ROW=ROW+1
2310 COL=15
2320 MSG$="DIE-"
2330 GOSUB 4900
2340 COL=21
2350 MSG$=STR$(PLAYER_DICE(I))
2360 GOSUB 4900
2370 NEXT I
2500 FOR I=1 TO 3
2510 DICE_VALUE(I)=INT(RND*6)+1
2520 NEXT I
2600 REM  DISPLAY DICE
2610 FOR I=1 TO 3
2620 CHAR_NO=DICE_VALUE(I)
2630 IF CHAR_NO=1 THEN 2740
2640 IF CHAR_NO=4 THEN 2740
2650 IF CHAR_NO=5 THEN 2740
2660 IF RND<.5 THEN 2740
2670 IF CHAR_NO<>2 THEN 2700
2680 CHAR_NO=7
2690 GOTO 2740
2700 IF CHAR_NO=6 THEN 2730
2710 CHAR_NO=8
2720 GOTO 2740
2730 CHAR_NO=9
2740 REM  DISPLAY A DIE
2750 FOR J=1 TO 9
2760 K=(I-1)*9+J
2780 CALL HCHAR(LOC_X(K),LOC_Y(K),
     96+DICE_PIP(CHAR_NO,J))
2790 NEXT J
2800 CALL SOUND(20,1111,0,1166,0,
     1221,1)
2990 NEXT I
3800 FOR I=1 TO 400
3810 NEXT I
3900 CALL SCREEN(4)
3910 CALL CLEAR
3920 RETURN
4900 FOR Z=1 TO LEN(MSG$)
4910 CALL HCHAR(ROW,COL+Z+1,
     ASC(SEG$(MSG$,Z,1)))
4920 NEXT Z
4930 RETURN
4990 REM  CHECK FOR A WINNER
5000 NO_LEFT=0
5010 FOR I=1 TO PLAYERS
5020 IF PLAYER_CASH(I)=0 THEN 5050
5030 NO_LEFT=NO_LEFT+1
5040 LAST_PLAYER=I
5050 NEXT I
5060 IF NO_LEFT>0 THEN 5200
5100 PRINT "NO ONE IS LEFT.":
     "THE GAME ENDS IN A TIE."
5110 GOTO 5400
5200 IF NO_LEFT>1 THEN 5400
5300 PRINT PLAYER_NAME$(LAST_PLAYER);
     " WINS!"
5400 RETURN
20000 PL$="PLEASE ANSWER THE QUESTION"
20010 CALL CHAR(96,"0000000000000000")
20020 CALL CHAR(97,"0000001818000000")
20030 CALL COLOR(9,2,16)
20050 CALL CLEAR
20090 ROW=12
20100 FOR I=1 TO 9
20110 FOR J=1 TO 9
20120 READ DICE_PIP(I,J)
20130 NEXT J
20140 IF INT(I/2)=I/2 THEN 20170
20150 MSG$="CHUCK-A-LUCK"
20160 GOTO 20180
20170 MSG$="            "
20180 COL=10
20190 GOSUB 4900
20200 NEXT I
20300 CNT=0
20310 FOR I=1 TO 3
20320 FOR J=1 TO 3
20330 FOR K=1 TO 3
20340 CNT=CNT+1
20350 LOC_Y(CNT)=J+I*4
20370 LOC_X(CNT)=K+2
20400 NEXT K
20410 NEXT J
20420 NEXT I
21000 CALL CLEAR
21010 INPUT
      "NEED INSTRUCTIONS (Y/N)? ":A$
21020 A$=SEG$(A$,1,1)
21030 IF A$="Y" THEN 21100
21040 IF A$="N" THEN 22000
21050 PRINT PL$
21060 GOTO 21010
21100 PRINT :"":"":"WELCOME TO THE
      GAME OF":"   CHUCK-A-LUCK!":""
21110 PRINT "THIS GAME CAN BE PLAYED
      BY":"1 TO 4 PLAYERS. EACH
      PLAYER STARTS OUT
      WITH $500. FOR"
21120 PRINT "EVERY TURN, EACH PLAYER
      BETSFROM $1 TO $50 ON A DICE
      VALUE FROM 1 TO 6. THREE"
21130 PRINT "DICE ARE THEN ROLLED.
      EACH  PLAYER WILL THEN RECEIVE
      AN AMOUNT EQUA L TO HIS BET"
21140 PRINT "MULTIPLIED BY THE
      NUMBER OF TIMES THE VALUE HE
      SELECTED CAME UP. IF
      NO DIE HAS THE"
21150 PRINT "VALUE SELECTED, THE
      PLAYER  LOOSES HIS BET.
      A PLAYER WHOGOES BANKRUPT IS
      OUT OF THE"
21160 PRINT "GAME. THE GAME IS OVER
      WHEN ONLY 1 PLAYER REMAINS. IF
      NOONE REMAINS, THERE IS NO"
21170 PRINT "WINNER. ":""
21500 FOR I=1 TO 1000
21510 NEXT I
22000 INPUT "HOW MANY PLAYERS
      (2-4)? ":PLAYERS
22010 IF PLAYERS<2 THEN 22060
22020 IF PLAYERS>4 THEN 22060
22030 IF INT(PLAYERS)=PLAYERS
      THEN 22100
22060 PRINT PL$
22070 GOTO 22000
22100 FOR I=1 TO PLAYERS
22110 PRINT "PLAYER NUMBER";
      I;"ENTER YOUR"
22120 INPUT "NAME-":
      PLAYER_NAME$(I)
22140 IF PLAYER_NAME$(I)<>""
      THEN 22250
22170 PRINT PL$
22180 GOTO 22110
22250 PLAYER_NAME$(I)=SEG$
      (PLAYER_NAME$(I),1,10)
22310 PLAYER_CASH(I)=500
22320 NEXT I
22330 RETURN
25000 DATA 0,0,0,0,1,0,0,0,0
25010 DATA 1,0,0,0,0,0,0,0,1
25020 DATA 1,0,0,0,1,0,0,0,1
25030 DATA 1,0,1,0,0,0,1,0,1
25040 DATA 1,0,1,0,1,0,1,0,1
25050 DATA 1,1,1,0,0,0,1,1,1
25060 DATA 0,0,1,0,0,0,1,0,0
25070 DATA 0,0,1,0,1,0,1,0,0
25080 DATA 1,0,1,1,0,1,1,0,1
25090 DATA X
28000 T2=700
28005 T=120
28010 CALL SOUND(T,392,1)
28020 CALL SOUND(T,523,1)
28030 CALL SOUND(T,659,1)
28040 CALL SOUND(T,784,1)
28050 CALL SOUND(T,784,1)
28060 CALL SOUND(T,784,1)
28070 CALL SOUND(T,659,1)
28080 CALL SOUND(T,659,1)
28090 CALL SOUND(T,659,1)
28100 CALL SOUND(T,523,1)
28110 CALL SOUND(T,659,1)
28120 CALL SOUND(T,523,1)
28130 CALL SOUND(T2,392,1)
28140 CALL SOUND(1,39999,30)
28150 CALL SOUND(T2,392,1)
28160 CALL SOUND(T,523,1)
28170 CALL SOUND(T,659,1)
28180 CALL SOUND(T,784,1)
28190 CALL SOUND(T,784,1)
28200 CALL SOUND(T,784,1)
28210 CALL SOUND(T,659,1)
28220 CALL SOUND(T,659,1)
28230 CALL SOUND(T,659,1)
28240 CALL SOUND(T,392,1)
28250 CALL SOUND(T,392,1)
28260 CALL SOUND(T,392,1)
28270 CALL SOUND(T2,523,1)
28280 RETURN
```

3-D . . . from p. 58

8 x 8 grid square to have *2 unique colors*. In Text Mode the screen is a 2-color *single* plane (so sprites aren't available) of 40 columns by 24 rows composed of 6 x 8 grid squares. This allows an ASCII character set with each character formed from a 5 x 7 pixel grid, with 2 pixels between characters and rows. Multicolor Mode [see also, *Super Crayon* in this issue] divides the pattern plane into an unrestricted 64-column by 48-row color-square display, with *each* 4 x 4 pixel square allowed to take on any of the 15 colors or be made transparent.

NOTICE:

has been indefinitely delayed in production.

Crayon . . . from p. 57

Table 1 **VDP RAM MEMORY** **—Editor/Assembler—**

Address of First Byte		Length of Block, Bytes	Contents
Decimal	Hex		
0	>0000	768	Screen Image Table
768	>0300	128	Sprite Attribute List
896	>0380	128	Color Table
1024	>0400	896	Sprite Descriptor Table
1920	>0780	128	Sprite Motion Table
2048	>0800	2048	Pattern Descriptor Table and Peripheral Access Blocks
4096	>1000	10199	More Peripheral Access Blocks and Buffers
14295	>37D7	2089	Reserved for Diskette Device Service Routines
16383	>3FFF	——	Last Address
Total 16384 Bytes			

(or 9918A if you have a 99/4A). This Video Display Processor (VDP) has eight 8-bit hardware registers and four 8-bit software registers. The software registers are located in read-write memory locations which can also be addressed by the CPU. The fact that these four bytes can be addressed by both the CPU and VDP makes it possible for the CPU and VDP systems to transfer data back and forth. The CPU addresses of the registers— > 8800, > 8802, > 8C00, > 8C02—are assigned respectively to the symbols VDPRD (VDP Read Data), VDPSTA (VDP Status), VDPWD (VDP Write Data), and VDPWA (VDP Write Address).

We don't have to be concerned with the details of moving data to and from VDP RAM and to VDP registers, however, thanks to some of the built-in programs called "utilities." The five utilities of use are identified by the symbols VSBW, VMBW, VSBR, VMBR, and VWTR. The respective functions of these programs are VDP RAM: Single Byte Write, Multiple Byte Write, Single Byte Read, Multiple Byte Read, and Write to Register. User workspace registers are used to pass parameters—e.g., the number of bytes to read or write—to the utility.

The standard utilization of VDP RAM in the Editor/Assembler is shown on Table 1. The blocks involved in the multicolor mode are the Screen Image and Pattern Descriptor Tables. Before entering multicolor mode, the Screen Image Table is initialized. The 768 bytes of the table are divided into six 128-byte sets. Each set is further subdivided into four 32-byte groups. To initialize the table, the numbers 0-31 are written in order into each of the four 32 byte groups in the first set: 0, 1, 2, . . . 31 four times. Then the numbers 32-61 are written four times into the next 128-byte set. This process is continued until the numbers 160-191 are written four times in the sixth 128-byte set. In my program, I didn't want this process to be visible on the screen, so I first put the display in Text Mode and made the foreground and background colors gray.

Once the Screen Image Table is initialized, color boxes are placed on the screen by means of the Pattern Descriptor Table. Each 4 x 4 pixel box on the screen corresponds to half a byte in the Pattern Descriptor Table. To place a colored box on the screen, the appropriate color code is written in the nybble (4 bits) in the Pattern Descriptor Table which corresponds to the desired screen position.

The first eight bytes of the Pattern Descriptor Table correspond to boxes in a column beginning in the upper left corner of the screen. The first four bits in byte #1 contain the color of the box in the extreme upper left corner and the last four bits the color of the box immediately to the right of the first box. Byte #2 contains the colors of the two boxes immediately under the first two, and so on for the first eight bytes.

The ninth byte in the table contains the colors for the pair of boxes in a new column beginning again at the top of the screen. Subsequent bytes follow this pattern corresponding to 32 columns of box pairs with eight pairs in each column. This group of 256 bytes thus takes care of the top sixth of the screen.

The 257th byte corresponds to the beginning of a new column of box pairs starting again on the left side of the screen. The six 256-byte groups thus correspond to the 3,072 possible boxes in multicolor mode. [Since the color of each box is indicated in a name table in memory, and the names are mapped onto the screen according to their position in the table, this multicolor mode is a *true* memory-mapped configuration. It does, however, trade off lower resolution for color memory-mapping capability, but the high-resolution sprites are still available. For an explanation of sprites and an introduction to the high-resolution bit-map mode, see *3-D Animation With the TMS9918A Video Chip* in this issue—Ed.]

In the program, a double-size sprite provides a reference point for determining where boxes will appear. The dot row and dot column of the sprite can be determined at any time by referring to the Sprite Attribute List in VDP RAM. Then, since boxes are supposed to appear in the center of the sprite, the screen location can be calculated by adding 8 to the dot row and dot column, which represent the sprite's upper left corner. But in order to find the corresponding location in the Pattern Descriptor Table, a few more calculations must be performed.

If we let R and C be the dot row and dot column desired for the box location, the number of complete 256-byte groups above that location is the integer quotient of R/32. Multiplying that number by 256 thus gives the first component of the offset in the Pattern Descriptor Table.

Similarly, the integer quotient of C/8 gives the number of complete 8-byte columns to the left of the location. So that number is multiplied by 8 and added to the offset. Dividing the remainder of R/32 by 4 gives the number of bytes above the location in the 8-byte column the location is in. Adding that to the offset gives the offset for the byte in the Pattern Descriptor Table.

But we still have to know if the desired location is the most or least significant nybble of the byte, and to determine that we can divide the remainder of C/8 by 4. If the integer quotient is 0, it's the left nybble; if 1, it's the right nybble. The appropriate color code then need only be placed in the correct nybble (leaving the other one unchanged) and the box appears just where it should.

Let's consider an example: Suppose the upper left corner of the sprite were at dot row 83 and dot column 147. The center of the sprite would then be at 91 and 155. The number of complete groups (32 columns with 8 bytes in each) above that location is 2--i.e., INT(91/32). So the initial component of the offset is 2 * 256 or 512 bytes. The number of 8-byte columns to the left of the location is INT(155/8) or 19. That makes the offset 531. Above the location, in its 8-byte column, there are 6 bytes—i.e., INT((remainder 91/32)/4)—giving an offset of 537. The remainder of 155/8 is 3, and INT(3/4) is 0, so the nybble of interest is the most significant (left) one of the 538th byte of the Pattern Descriptor Table.

Now let's take a brief look at the source listing. The first section consists of a number of assembler directives. The DEF directive makes the symbol MARKER available to other programs, and the REF directives make several utilities available for use in MARKER. Then there are a variety of other assembler directives. The simplest type is EQUate which assigns a constant to a symbol at assembly time. USRWS, for instance, will be assigned the value of >20BA(8378), and that value replaces the symbol wherever it appears in an operand; the label may subsequently be substituted for the number.

The mnemonic BSS stands for Block Starting with Symbol and this directive causes the assembler to advance its location counter without writing anything into the object program. It leaves an empty area (of the number of bytes specified in the operand) which can then be used as a storage space for data later on. The label is set equal to the memory location of the first byte in the block at the time the object program is loaded. (Since this program is relocatable, the place where the loader program decides to start loading it may change depending on what other programs have already been loaded.)

The DATA, BYTE, and TEXT directives are similar to BSS except that the contents of the buffer are explicitly

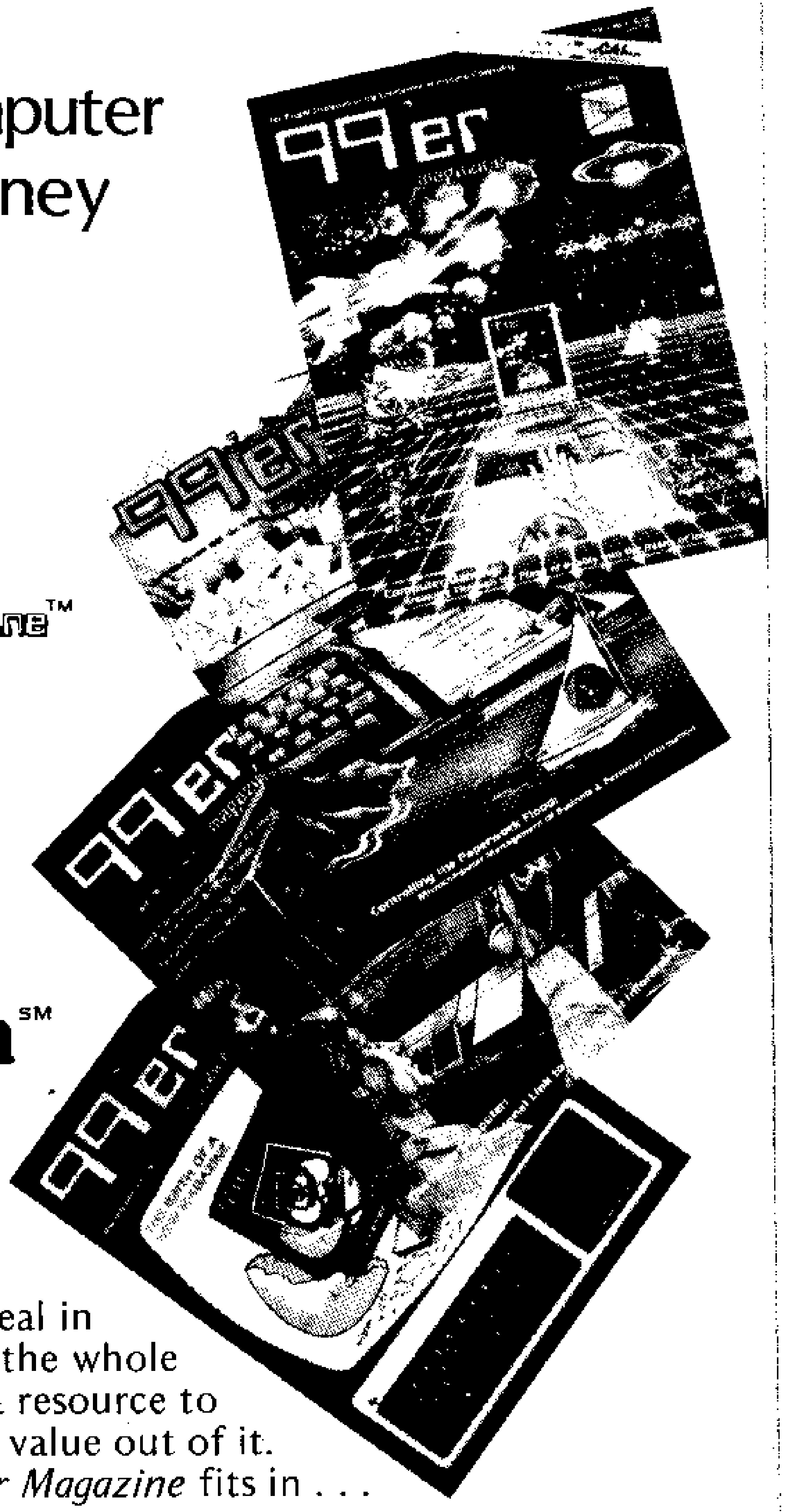
Continued on p. 83

WE CHALLENGE YOU TO COMPARE—

☞ Try To Find Any Other Personal Computer
That Gives You More For Your Money
Than Texas Instruments . . .

. . . Or Any Other Magazine
That Helps You Benefit More
From A Personal Computer Than 99'er magazine™

☞ There Are None !



That's Why The Experts Say:

Once You Compare— There's No Comparison℠

99'er magazine™ SUBSCRIPTION

□ YES—Please sign me up as a subscriber. Enclosed is my payment or credit card billing information.

Term	U.S.A.	Canada & Mexico	Foreign Surface	Foreign Air
1-yr (6 issues)	□ $18	□ $22	□ $28	□ $43
2-yr (12 issues)	□ $32	□ $42		
3-yr (18 issues)	□ $45	□ $60		

Sample Issue □ $3.95

Address shown is □ Business □ Home   □ Check enclosed   MUST BE IN U.S. FUNDS DRAWN ON A U.S. BANK

NAME ______________________ PLEASE

ADDRESS ______________________ PRINT

CITY ______________ STATE ______ ZIP ______

Bill my:

□ VISA   ▶ Expiration Date ______________

□ Master Charge   Signature ______________

Account No. □□□□□□□□□□□□□□□□

MC Only — List 4 digits above your name □□□□

Please Mail Your Order To:

99'er Magazine
P.O. Box 5537
Eugene, OR 97405

Credit card phone orders call: (503) 485-8796

Finding the best deal in a computer isn't the whole story. You also need a resource to help you get the most value out of it. That's where the *99'er Magazine* fits in . . .

As the *ONLY* magazine exclusively for the Texas Instruments personal computers—the TI-99/4A family—99'er Magazine is a MUST for all current users interested in entertainment, education, business, professional, and home applications. Also, it is a valuable information resource for those who want to know more about these friendly, value-packed personal computers in order to make an intelligent purchase decision in the future . . .

A 4-in-1 Resource

As a special bonus, 99'er Magazine includes the bound-in publications—*OnLoCAItion: The International Journal of Computer Assisted Instruction; LOGO Times: The Magazine of the LOGO Language;* and *Computer Gaming*, a treasure-trove of fun and excitement.

A Multi-Level Teacher and News Medium With Lots of FREE Software

Each BIG issue of 99'er Magazine contains tutorials for beginners, applications tips and advanced programming techniques that keep the pros coming back for more, as well as nearly *one dozen (!)* ready-to-run computer programs for EVERYONE. Additionally, there's in-depth descriptions and reviews of the latest hardware, software, and books —timely information to keep its readers well informed and help them buy products wisely.

99'er magazine™ A Resource for People Interested in the Enrichment of Personal Computing

Satisfaction Guaranteed

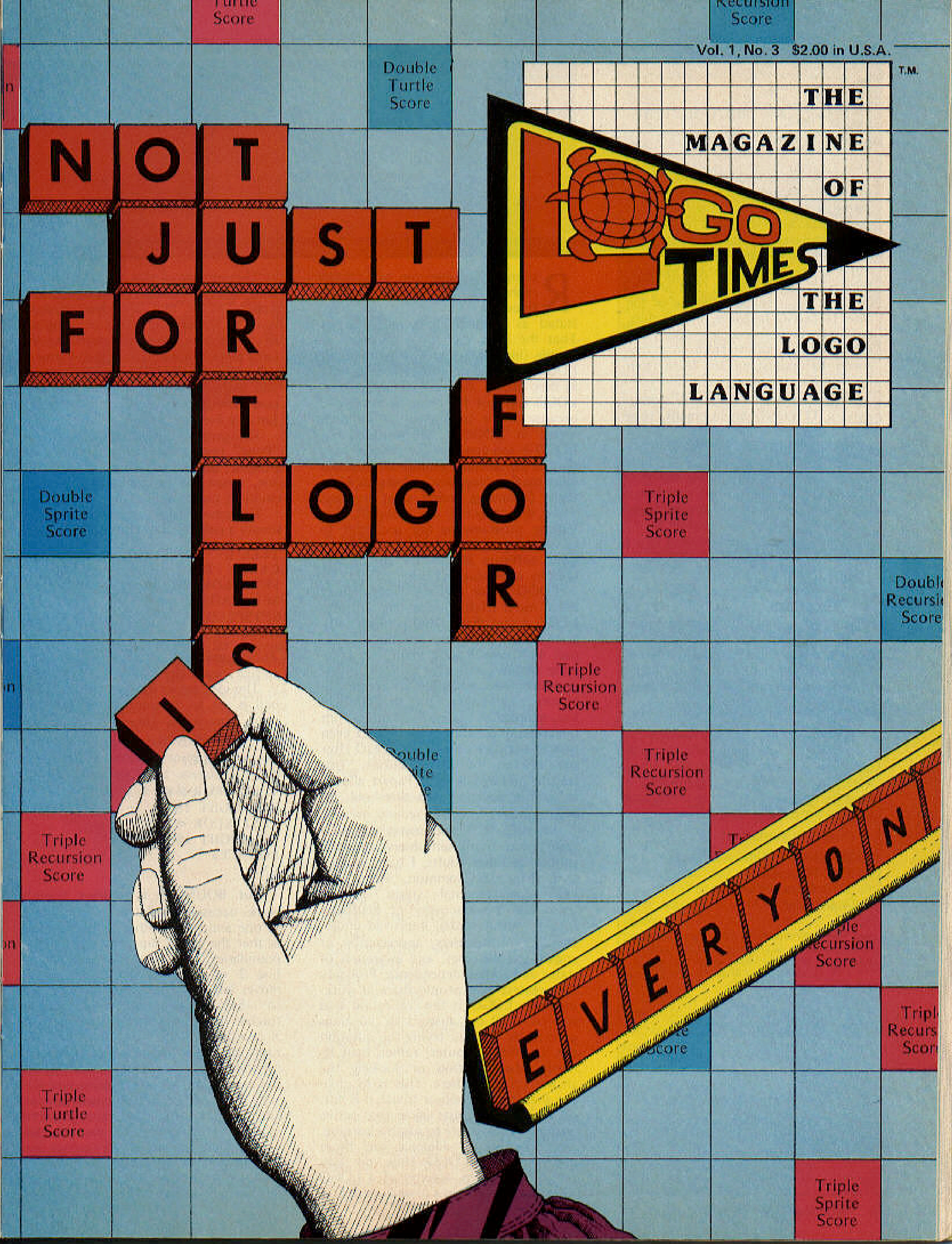
Or the unfilled portion of your subscription will be refunded

℠ A service mark of Emerald Valley Publishing Co.

Vol. 1, No. 3 $2.00 in U.S.A.

LOGO TIMES T.M.

THE MAGAZINE OF THE LOGO LANGUAGE



Introduction

LOGO Times is an information resource for anyone interested in participating in the creation of their own *personal* language – one that will easily allow them to communicate with a computer in a totally new audiovisual realm of applied imagination, exploration, and self-discovery. The articles on these pages concern the use of the new TI LOGO language, but readers, however, do *not* need any additional software or equipment (or even a computer) to understand and learn from the material presented here.

If readers want to actually *experience* a TI LOGO environment, thev will need either a TI-99/4 or TI-99/4A computer, the Expansion Memory peripheral, and the TI LOGO Command Module. A disk drive, although convenient to have, is *not* required; a user's work may alternately be saved on cassette tape, printed out on the TI Thermal Printer, or hand copied into a notebook (for later re-keyboarding).

In each issue, one or more of the articles may reference or build upon the topics discussed in a previous article. It is therefore recommended that for maximum benefit and understanding, new readers obtain the appropriate back issues of *99'er Magazine* in which the *LOGO Times* articles are contained.

Notice

LOGO Times is actively soliciting articles. Manuscripts should be typed double-spaced, and accompanied by a cassette tape or disk if containing any lengthy procedures or graphics.

Send all materials to:

LOGO Times Editorial Dept.
99'er Magazine
2715 Terrace View Drive
Eugene, OR 97405

All mail directed to the Letters-to-the Editor column (*Letters on LOGO*) will be published in accordance with the conditions set forth on *99'er Magazine*'s contents pages.

Our Contributing Editors

Henry Gorman, Jr.
Department of Psychology
Austin College
Box 1584
Sherman, TX 75090

Roger B. Kirchner
Department of Mathematics
Carleton College
Northfield, MN 55057

WHO IS LOGO FOR?

Recently the question of LOGO's relevance for children and its relevance for adults has been stated as an implicit either/or issue. That the issue ever arose, means that people (including me) who write about LOGO have not done their jobs as fully as they should. Perhaps the notion that LOGO was just for children developed because of the total attention children invest in LOGO. The position that LOGO is too complex for children may have arisen because published programs seem magic unless one actively explores them (including seeing what happens when the programs are changed). Presenting programs as *fait accompli* to be copied, run, stored, and used like any other software is contrary to the philosophy of education behind LOGO.

LOGO is for humans. When Papert asked me if I felt comfortable with my LOGO, I said that LOGO is like a hologram--when you grasp just the smallest part of it, you have a small, but complete picture; and later as your understanding grows, you still have a complete picture, albeit larger. From that perspective, people can always learn more from LOGO and do more with LOGO even though they are able to use LOGO after the briefest of introductions. This feature of Logo is what Papert alludes to in his slogan, "Low threshold, no ceiling."

The LOGO slogan invites empirical verification. In my self-observations and studies of other adults, I have noticed that there are common, identifiable LOGO-developmental stages. Among these are the discovery of heuristics (i.e., powerful ideas), improved understanding of numbers, appreciation of angles and headings, and awareness of states and state independence. Probably the greatest gain people share in working with LOGO is the realization that one can find out answers to questions quickly by actively trying the question out on the computer, rather than ignoring the question or looking the answer up somewhere. This is so obvious that it might appear trivial; it is not. All learning theorists agree that active learning is preferable to passive learning. This presents a dilemna for those writing about LOGO How do you capture the open, activity of a LOGO learning enterprise in a closed article?

The purpose of *this* article, howeve is to reflect the development of a LOG game, and in that development sho how an apparently complex progra is child's play, even for adults. At t same time, I hope that the developme will point to variations and will enti you into active exploration. The pr gram was initiated by a student in m January Term course last winter.

The program was supposed to be "Pong" type game. As you follow i growth, find the point, if there i one, where the program stops being chidren's program.

The game begins not as program, bu simply a collection of conditions.

```
TELL 0
CARRY :BALL
SETCOLOR :BLUE
SETHEADING 90
SETSPEED 15
HOME
```

These commands set a ball speeding lef to-right across the middle of the screen

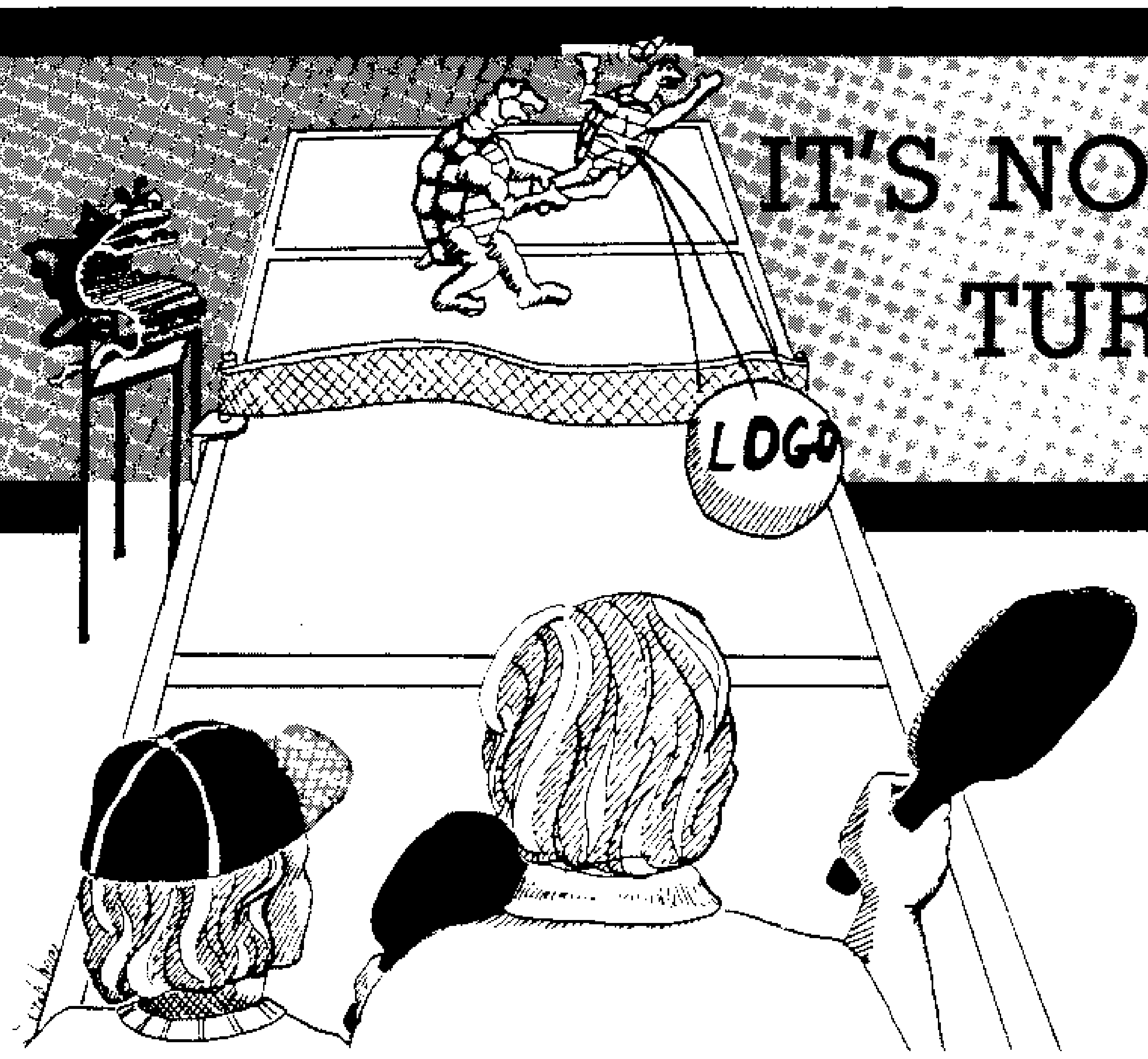
The idea grows into a program as t ball is set to "bouncing" off left an right boundaries. This is accomplishe any of several ways:

```
TO BOUNCE1
TELL 0
TEST XCOR > 85
IFT RIGHT 180
TEST XCOR < -85
IFT RIGHT 180
BOUNCE1
END
```

But BOUNCE1 sometimes doesn work--occasionally the sprite is "caught at one end or the other. What happe is that the sprite slips past one of t boundaries (e.g., the computer is line 2 of the program as the sprit moves left through X coordinate equ to --85); by the time the comput reaches line 4, the sprite is well left o X coordinate -85. Then the comput turns the sprite right 180 (a right 18 functions equivalent to a left 180 Before the sprite can move beyond th -85 X coordinate, the computer check line 4 again, turns the sprite 180 an sends it still further to the left. O course, when the computer reaches lin 4 a third time, the sprite is still left o --85; the poor sprite is stuck beyond th left-hand boundary! This bug could b

IT'S NOT JUST FOR TURTLES ANYMORE . . .

By Henry Gorman Jr.



eliminated with a second type of BOUNCE program:

```
TO BOUNCE2
TELL 0
TEST XCOR >85
IFT SETHEADING 270
TEST XCOR < -85
IFT SETHEADING 90
BOUNCE2
END
```

Now, regardless of how far beyond either boundary the sprite travels, the program will change the sprite's heading so that it will move back away from the boundary. A second bug could occur if one used BOUNCE2 without first typing in the setup commands, since BOUNCE2 requires sprite 0 to have a shape, heading, and speed. To avoid any problems, a better arrangement would be:

```
TO GAME
SETUP
BOUNCE2
END
```

and

```
TO SETUP
TELL 0
CARRY :BALL
SETCOLOR :BLUE
HOME
SETHEADING 90
SETSPEED 15
END
```

A ball bouncing between two boundaries is not much of a pong game. A closer approximation would result if there were a paddle for the ball to bounce off. This could be achieved by merely putting two sprites together as a team - stacked vertically on top of each other, with each carrying a box. Since the team of sprites is, like the sprite carrying the ball, part of the initial game setup, the team should be part of the SETUP program:

```
TO SETUP
TELL 0
CARRY :BALL
SETCOLOR :BLUE
HOME
SETHEADING 90
SETSPEED 15
TELL 1
CARRY :BOX
SXY 100 0
SETCOLOR :BLACK
SETHEADING 0
TELL 2
CARRY :BOX
SETCOLOR :BLACK
SETHEADING 0
SXY 100 16
END
```

Notice, however, that sprites 1 and 2 receive almost identical commands, so that a cleaner SETUP program can be written:

```
TO SETUP
TELL 0
CARRY :BALL
SETCOLOR :BLUE
HOME
SETHEADING 90
SETSPEED 15
TELL [ 1 2 ]
CARRY :BOX
SETHEADING 0
SETCOLOR :BLACK
SXY 100 0
TELL 2
SY 16
END
```

To make the game even more realistic, it is necessary to change the heading of the ball, to have the player able to move the paddle, and to keep a score. Obviously the ball should only bounce when it hits the paddle! These additions are complex, so one should apply a Papert "powerful idea" and work on the complexity in smaller, simpler parts.

The movement of the paddles can be accomplished by:

```
TO PADDLE
CALL RC "A
IF :A = "E TELL [ 1 2 ]
  FORWARD 16
IF :A = "X TELL [ 1 2 ] BACK
   16
PADDLE
END
```

and PADDLE is simply added to the GAME

```
TO GAME
SETUP
PADDLE
BOUNCE2
END
```

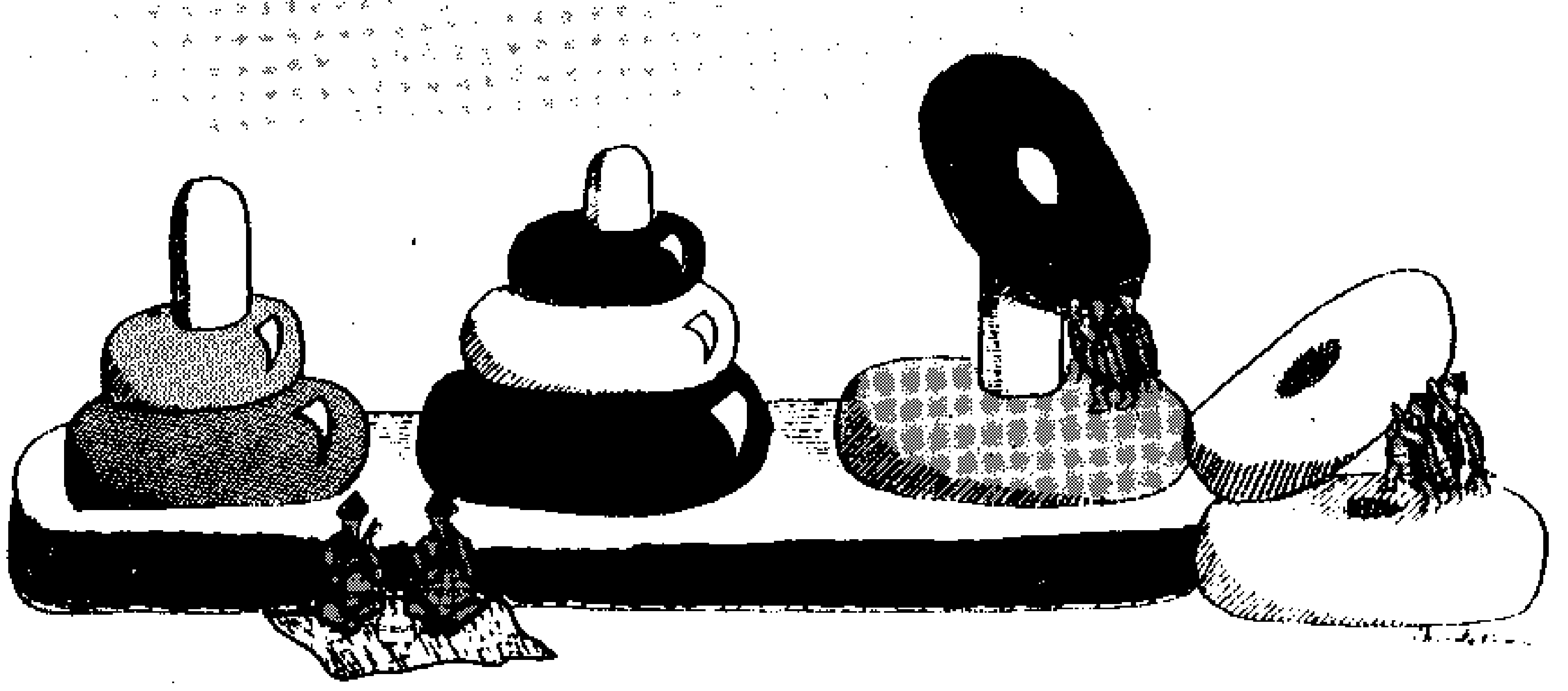
Ooops; there's a very bad bug in this—the ball never bounces because PADDLE is recursive without a stop rule, and the computer never reaches BOUNCE2. So the recursive line in PADDLE is removed

```
TO PADDLE
CALL RC "A
IF :A = "E TELL [ 1 2 ]
  FORWARD 16
IF :A = "X TELL [ 1 2 ] BACK
  16
END
```

But now, when GAME is run, there's another bad bug, the program sets up, allows for one paddle movement and then stays stuck in BOUNCE2. Once again the difficulty is that a subprocedure is recursive. As a general rule, when a recursive program is used as a building block for a more complex program, there can be a bug. The bug is common enough to deserve a name —the "Recursion Interface Bug." When the bug is corrected by removing the recursive line of BOUNCE2, a new bug appears.

```
TO BOUNCE2
TELL 0
TEST XCOR > 85
IFT SETHEADING 270
TEST XCOR < -85
IFT SETHEADING 90
END
```

Continued on p. 68



TOWER OF

The Tower of Hanoi is a puzzle that consists of a number of different sized rings and three pegs. The rings are initially on one of the pegs in order, with the largest at the bottom. The puzzle is to move the rings one at a time from one peg to another, but such that a larger ring is never moved on top of a smaller one. In the last issue, we showed how LOGO could be used to develop a solution to the puzzle. We now describe how the puzzle might be implemented graphically in LOGO.

Let A, B, and C be the three pegs. Since the state of the puzzle is given by knowing which rings are on which pegs, it seems natural to let A, B, and C be names for lists which tell which rings are on each peg. Our puzzle will have 8 rings. Let us number them 1 through 8 in order of increasing size. Then the beginning position, with all rings on peg A, is represented by :A = [1 2 3 4 5 6 7 8], :B = [ ], and :C = [ ]. Moving the top ring from A onto B results in the state :A = [2 3 4 5 6 7 8].:B = [1], :C = [ ]. A move essentially consists of removing a number from the beginning of one list and adding it to the beginning of another list. At the same time, of course, the ring must be erased and redisplayed in the correct position.

Let us first construct a procedure, HANOI, which will allow us to play with the puzzle and then, when we want, solve it automatically:

```
TO HANOI
INITIALIZE
SETUP
PLAY
SETUP
SOLVE 8 “A “B “C
END
```

INITIALIZE should set colors and define constants. SETUP should display the puzzle with all the rings on peg A. PLAY should allow us to pick rings up and put them down by simply pressing the names of the corresponding pegs. Play might continue until 'Q' is pressed. The puzzle should then be redisplayed and solved automatically, beginning with the rings on peg A, and ending with the rings on peg B. The procedure SOLVE was developed in the last issue. Procedures SETUP, PLAY, and SOLVE will depend on workhorse procedures GETRING and SETRING. The requirements for INITIALIZE will become apparent as we make choices about representation.

Assume that INITIALIZE assigns the value 8 to N and then :TOP is the number of the ring to be displayed. Then SETUP can be:

```
TO SETUP
MAKE “A [ ]
MAKE “B [ ]
MAKE “C [ ]
STAND “A
STAND “B
STAND “C
MAKE “TOP :N
REPEAT :N [ SETRING “A MAKE
  “TOP :TOP – 1 ]
END
```

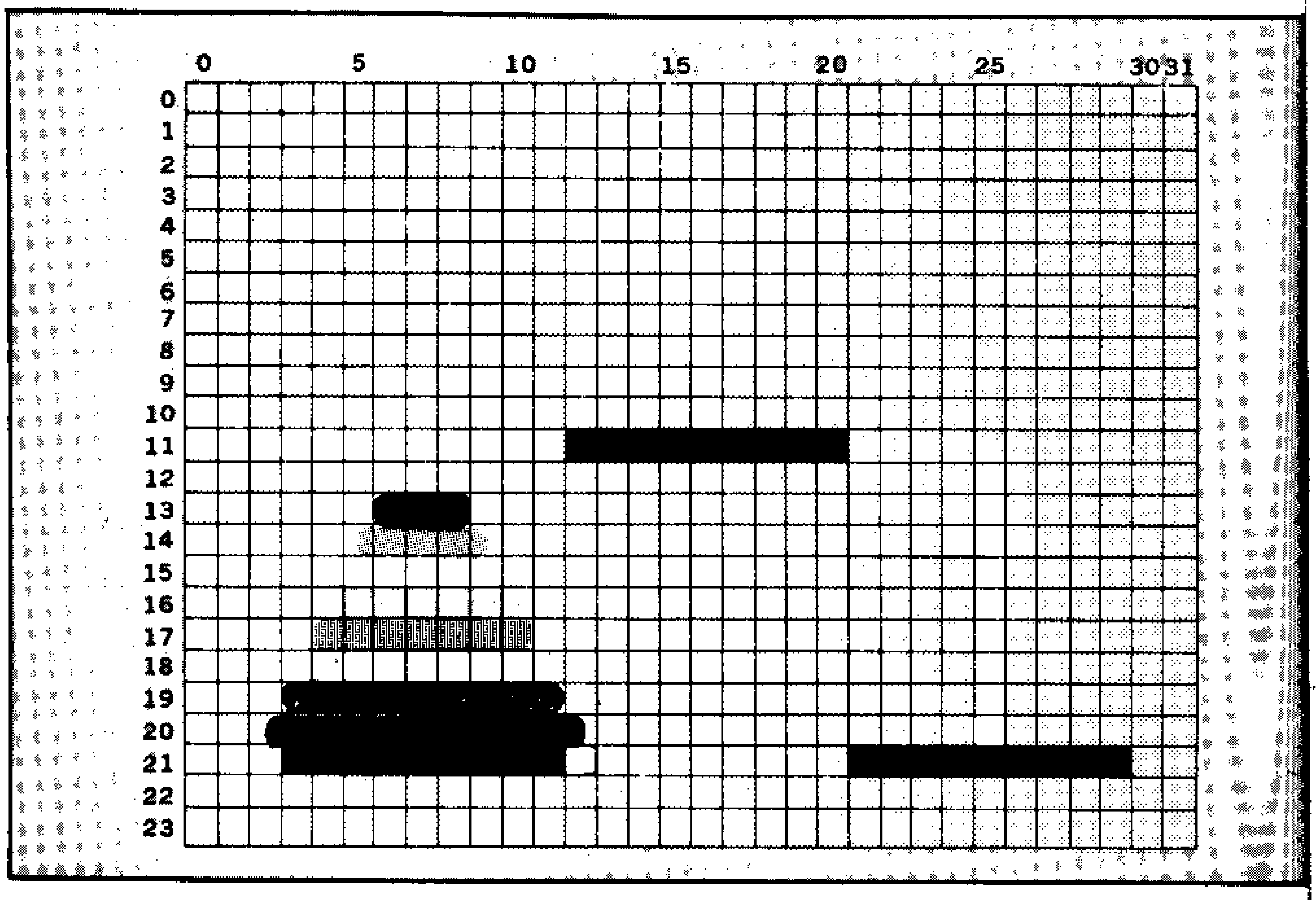
Using utilities MEMBER?, EMPTY?, and ALARM, we can write PLAY in such a way as to validate all inputs. We want to accept 'Q' to stop PLAY, but otherwise only the letters A, B, and C. (:VALID will be initialized to [A B C].) We also want to prevent an attempt to remove a ring from an empty peg. If an error is made, we will cause an alarm to be sounded. (See the listing for definitions of the utilities.)

```
TO PLAY
L1:
MAKE “X RC
IF :X = “Q THEN STOP
IF NOT MEMBER? :X :VALID
  THEN GO “L1
IF EMPTY? THING :X THEN
  ALARM GO “L1
GETRING :X
L2:
MAKE “X RC
IF NOT MEMBER? :X :VALID
  THEN ALARM GO “L2
SETRING :X
PLAY
END
```

In this procedure, note that the valu of X, :X, is the name of a peg, eithe A, B, or C. One might expect that th value of :X would be denoted : :X but this denotes the value of ':X'. Th primitive THING must be used. THING :X is the list named by :X.

In order to discuss GETRING an SETRING, we need to be specific abou how the graphics will be represented We could use the turtle, but we choos tiles because this allows the most colorful display. The LOGO screen is divide into 32 columns numbered 0 to 31 fro left to right, and 24 rows numbere 0 to 23 from top to bottom. We ca locate the rings by locating them relative to their pegstands. Let ABASE BBASE, and CBASE name the pairs o column and row coordinates for th centers of the pegstands. Reasonabl choices are :ABASE = [7 21] :BBASE = [25 21], and :CBASE = [16 11]. Suppose a ring is the to one on a given peg. Its center has it column the same column as the peg, an its row equal to the row of the bas minus as many rings as are on the peg. I we use TOP, COL, and ROW to name th number of the top ring and its colum and row coordinates, we are led to

```
TO GETRING P
MAKE “BCOORD THING WORD
  :P “BASE
MAKE “COL FIRST :BCOORD
MAKE “K COUNT THING :P
```



HANOI Part Two

By Roger B. Kirchner

```
MAKE "ROW (LAST :BCOORD)
   - :K
ERASERING
MAKE :P BF THING :P
END

TO SETRING P
MAKE "P SE :TOP THING :P
MAKE "BCOORD THING WORD
   :P "BASE
MAKE "COL FIRST :BOORD
MAKE "K COUNT THING :P
MAKE "ROW (LAST :BCOORD)
   - :K
DISPLAYRING
END
```

In using these procedures, :P is a letter (A, B, or C). Thus WORD :P "BASE will return the word ABASE, BBASE, or CBASE. Note how BF (BUTFIRST) and SE (SENTENCE) are used to change the value of :P (which will equal A, B, or C). By passing the name of the peg, we can change its value. This would not be the case if we passed the value of the peg to the procedure. (Computer scientists call this passing parameters "by reference" rather than "by value.")

We are left with the problems of actually displaying the pegs, and displaying and removing the rings. The work will be done by STAND, DISPLAYRING, and ERASERING. We need to choose the tiles and colors.

The bases will use tile 96 and be black. The pegs will use tiles 104 and 105, and be white. Tile 104 is square, and tile 105 is rounded at the top. Recall that the number of rings is :N, and the division in LOGO is integer division.

```
TO STAND P
MAKE "BCOORD THING WORD :P
   "BASE
MAKE "COL FIRST :BCOORD
MAKE "ROW LAST :BCOORD
MAKE "J :COL - :N/2

REPEAT 1+2*(N/2) [ PT 96 :J
   :ROW MAKE "J :J + 1 ]
MAKE "K :ROW - 1
REPEAT :N [ PT 104 :COL :K
   MAKE "K :K - 1 ]
PT 105 :COL :K
PT CHARNUM :P :COL :ROW + 1
END
```

Tiles and colors for the rings will be chosen as follows: The shapes for the tiles (see figure) are designed so that ring k appears to be k + 2 tiles wide, but it is actually 3 + 2*(k/2) tiles wide.

Ring	Tiles	Color	Tiles wide
1	112,113,114	Red	3
2	120,121,122	Orange	5
3	128,129,130	Yellow	5
4	136,137,138	Lime	7
5	144,145,146	Olive	7
6	152,153,154	Sky	9
7	160,161,162	Blue	9
8	168,169,170	Purple	11

Displaying a ring is accomplished by displaying the right number of tiles of the right shape and color. Erasing a ring is done by displaying blanks and the peg tile. For effect, the rings will be displayed from the center out, and erased from the outside in.

```
TO DISPLAYRING
MAKE "LT 104 + :TOP*8
MAKE "MID 105 + :TOP*8
MAKE "RT 106 + :TOP*8
PT :MID :COL :ROW
MAKE "J 1
REPEAT :TOP/2 [ PT :MID :COL
   - :J :ROW PT :MID :COL + :J
   :ROW MAKE "J :J+1 ]
PT :LT :COL - :TOP/2 - 1 :ROW
PT :RT :COL + :TOP/2 + 1 :ROW
END

TO ERASERING
MAKE "J 1 + :TOP/2
REPEAT 1 + :TOP/2 [ PT 32
   :COL - :J :ROW PT 32 :COL +
   :J :ROW ]
PT 104 :COL :ROW
END
```

We are almost ready to play with the puzzle. INITIALIZE (see listing) defines colors for the tiles, and assigns values to N, VALID, ABASE, BBASE, and CBASE. Before anything will happen, though, the tiles must be defined using MAKECHAR. (See figures.) Then, ENJOY! Recall that to manipulate the rings, you just need to press the letter of the peg from which you want to take, or to which you want to add a ring. Use the procedure HELP if you forget.

After you have had some fun with the puzzle, you might want to try a four peg variation. To implement a four peg version, do the following:

Change INITIALIZE to include:

```
MAKE "VALID [ A B C D ]
MAKE "ABASE [ 8 10 ]
MAKE "BBASE [ 24 10 ]
MAKE "CBASE [ 8 23 ]
MAKE "DBASE [ 24 23 ]
```

IN SETUP, add:

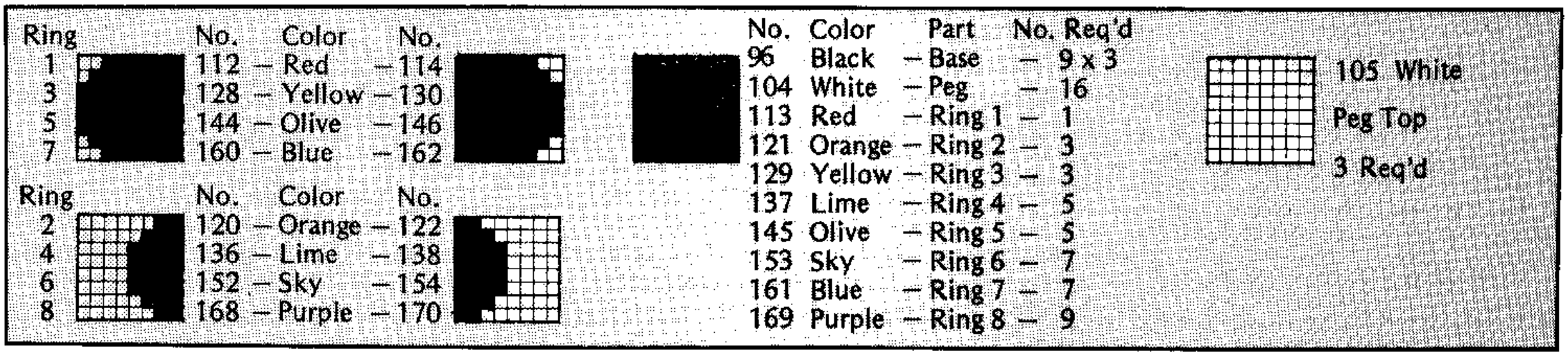
```
MAKE "D [ ]
STAND "D
```

The puzzle should then contain four pegs A, B, C, and D. It can be manipulated just like the three peg puzzle. The automatic solution will still use just three pegs. But as a worthy challenge, you might try to write a better version of SOLVE which takes advantage of the fact that there are two auxiliary pegs instead of just one. The puzzle should take fewer moves to solve. How many less than $2^n - 1$ moves are required if there are n rings and four pegs? I would be interested in any of your results. Then, can five pegs be fit on the screen . . . ?

But if you are looking for less of a challenge, or just want to experiment with a simpler puzzle, note that the number of rings is set in INITIALIZE, and can be changed. Try this: Enter INITALIZE, and then MAKE "N 5 (or some other integer). If you now enter SETUP, a puzzle with 5 rings will be displayed. Enter PLAY, and you can manipulate this puzzle until you press Q. Now enter SETUP again, and then SOLVE 4 "A "C "B. This will cause four rings to be moved automatically to peg C. Then enter PLAY, and you can complete the puzzle by yourself. With LOGO, the procedures are your own to do with or modify as you please. Use your imagination, make up other puzzles, or just go ahead and play with this issue's puzzle as is.

Whatever you choose,
You're sure to enjoy
Our new LOGO toy—
The classic and colorful
Game of Hanoi.

Listings p. 68



Ring	No.	Color	No.
1	112	– Red	–114
3	128	– Yellow	–130
5	144	– Olive	–146
7	160	– Blue	–162

Ring	No.	Color	No.
2	120	– Orange	–122
4	136	– Lime	–138
6	152	– Sky	–154
8	168	– Purple	–170

No.	Color	Part	No. Req'd
96	Black	– Base	– 9 x 3
104	White	– Peg	– 16
113	Red	– Ring 1	– 1
121	Orange	– Ring 2	– 3
129	Yellow	– Ring 3	– 3
137	Lime	– Ring 4	– 5
145	Olive	– Ring 5	– 5
153	Sky	– Ring 6	– 7
161	Blue	– Ring 7	– 7
169	Purple	– Ring 8	– 9

105 White
Peg Top
3 Req'd

Hanoi . . . from p. 67

Note: Due to an inherent bug in the LOGO Command Module, the auto-solve mode might sometimes "quit" in the middle of a sequence of moves. If this is the case, try to "toggle" the internal "gremlins" by typing in DEBUG (another undocumented command), then trying again. Then if it still doesn't work, turn the "debugger" off by typing in DEBUG, and typing again. If this still doesn't get it to run through without halting prematurely, try SAVEing to disk or tape, and RECALLing HANOI. This combination "fix" routine should help wake up the little "gremlin" who keeps falling asleep on the job—Ed.

```
TO ERASERING
MAKE "J 1 + :TOP / 2
REPEAT 1 + :TOP / 2 [PT 32 :COL
- :J :ROW PT 32 :COL + :J :ROW M
AKE "J :J - 1 ]
PT 104 :COL :ROW
END

TO SETUP
CS                      TO HANOI
STAND "A                INITIALIZE
STAND "B                SETUP
STAND "C                PLAY
MAKE "TOP :N            SETUP
MAKE "A [ ]             SOLVE 8 "A "B "C
MAKE "B [ ]             END
MAKE "C [ ]
REPEAT :N [SETRING "A MAKE "TOP
:TOP - 1 ]
END

TO STAND P
MAKE "BCOORD THING WORD :P "BASE
MAKE "COL FIRST :BCOORD
MAKE "ROW LAST :BCOORD
MAKE "J :COL - :N / 2
REPEAT 1 + 2 * ( :N / 2 ) [PT 96
 :J :ROW MAKE "J :J + 1 ]
MAKE "K :ROW - 1
REPEAT :N [PT 104 :COL :K MAKE "
K :K - 1 ]
PT 105 :COL :K
PT CHARNUM :P :COL :ROW + 1
END

TO DISPLAYRING
MAKE "LT 104 + :TOP * 8
MAKE "MID 105 + :TOP * 8
MAKE "RT 106 + :TOP * 8
PT :MID :COL :ROW
MAKE "J 1
REPEAT :TOP / 2 [PT :MID :COL -
:J :ROW PT :MID :COL + :J :ROW M
AKE "J :J + 1 ]
PT :LT :COL - :TOP / 2 - 1 :ROW
PT :RT :COL + :TOP / 2 + 1 :ROW
END

TO COUNT LIST
IF :LIST = [ ] THEN OUTPUT 0 ELS
E OUTPUT 1 + COUNT BF :LIST
END

TO GETRING P
MAKE "BCOORD THING WORD :P "BASE

MAKE "TOP FIRST THING :P
MAKE "COL FIRST :BCOORD
MAKE "K COUNT THING :P
MAKE "ROW ( LAST :BCOORD ) - :K
ERASERING
MAKE :P BF THING :P
END

TO EMPTY? LIST
IF :LIST = [ ] THEN OUTPUT "TRUE
 ELSE OUTPUT "FALSE
END

TO SOLVE N P1 P2 P3
IF :N = 1 THEN GETRING :P1 SETRI
NG :P2 STOP
SOLVE :N - 1 :P1 :P3 :P2
GETRING :P1 SETRING :P2
SOLVE :N - 1 :P3 :P2 :P1
END

TO SETRING P
MAKE :P SE :TOP THING :P
MAKE "BCOORD THING WORD :P "BASE
MAKE "COL FIRST :BCOORD
MAKE "K COUNT THING :P
MAKE "ROW ( LAST :BCOORD ) - :K
DISPLAYRING
END

TO HELP
CS
PRINT [TYPE HANOI TO BEGIN. ]
PRINT [ ]
PRINT [PUSH A, B, OR C TO REMOVE
 OR SET DOWN A RING. ]
PRINT [ ]
PRINT [TO QUIT, AND WATCH THE PU
ZZLE SOLVED AUTOMATICALLY, PUSH
Q. ]
END

TO INITIALIZE
TELL TILE 96 SC :BLACK
TELL TILE 104 SC :WHITE
TELL TILE 112 SC :RED
TELL TILE 120 SC :ORANGE
TELL TILE 128 SC :YELLOW
TELL TILE 136 SC :LIME
TELL TILE 144 SC :OLIVE
TELL TILE 152 SC :SKY
TELL TILE 160 SC :BLUE
TELL TILE 168 SC :PURPLE
MAKE "N 8
MAKE "VALID [A B C ]
MAKE "ABASE [7 21 ]
MAKE "BBASE [25 21 ]
MAKE "CBASE [16 11 ]
END

TO MEMBER? X LIST
IF :LIST = [ ] THEN OUTPUT "FALSE
IF :X = FIRST :LIST THEN OUTPUT
"TRUE
OUTPUT MEMBER? :X BF :LIST
END

TO PLAY
L1:
MAKE "X RC
IF :X = "Q THEN STOP
IF NOT MEMBER? :X :VALID THEN AL
ARM GO "L1
IF EMPTY? THING :X THEN ALARM GO
 "L1
GETRING :X
L2:                             TO ALARM
MAKE "X RC                      BEEP
IF NOT MEMBER? :X :VALID        WAIT 30
THEN ALARM GO "L2               NOBEEP
SETRING :X                      END
PLAY
END
```

Who . . . from p. 65

The ball doesn't bounce, or only bounces once, and the paddles only work once. This bug is killed by:

TO GAME
SETUP
PADDLE
BOUNCE2
GAME
END

With that fix, the paddles work, but a completely new SETUP happens at every execution of GAME. A better solution is to seperate those subprocedures which should be repeated from those which need to happen just once:

PADDLE   SETUP
BOUNCE2

and construct a new, superprocedure:

TO PLAY
SETUP
GAME
END

and alter GAME:

TO GAME
PADDLE
BOUNCE2
GAME
END

There is still a small bug left in PADDLE—the computer will wait at line 1 of GAME until a key is touched (to satisfy the CALL RC "A command it needs an RC). The computer needs to skip PADDLE if no key is touched. You can accomplish this using TEST and the operation RC? (RC? answers "TRUE when a key is touched and "FALSE if a key is not).

TO GAME
TEST RC?
IFT PADDLE
BOUNCE2
GAME
END

At last, the programs are all bug-free and working. The final tasks consist of linking the ball-bounce off the right to hit the paddle, keeping a score, and making the flight of the ball a little more eccentric. Again these are complex problems, so each should be tackled separately.

The BOUNCE2 program now reads:

TO BOUNCE2
TELL 0
TEST XCOR > 85
IFT SETHEADING 270
TEST XCOR < –85
IFT SETHEADING 90
END

The second line causes the bounce off the right-hand boundary. If that TEST were altered so that it answered "TRUE only when the ball is near the paddle or if a new program were designed to check the relationship of the ball to the paddle's Y coordinates when the ball was to the right of X coordinate 85, then the problem could be solved. The paddle is always at X coordinate 100 and at a variable Y coordinate; since the ball is in motion, the TEST at 85 is reasonable: when the ball passes through XCOR = 85 it will approach XCOR = 100 by the time the computer has completed all of the Y coordinate tests. The paddle begins the game (through SETUP) with the extremes of its Y coordinates between –16 and 16; each time the E key is typed, the paddle advances 16 along the Y coordinate, and each time that X is typed, it backs up 16 on the Y coordinate. Therefore, some PADDLETOUCH operation is needed which can compare the Y coordinate of the ball and that of the paddle:

```
TO PADDLETOUCH
TELL 0
TEST EITHER YCOR > :Y
  YCOR <( :Y – 32 )
IFT OUTPUT "FALSE
OUTPUT "TRUE
END
```

This program will answer "TRUE whenever the ball (carried by sprite 0) is between :Y and (:Y –32) on the Y coordinate and "FALSE when the ball is above or below that range on the Y coordinate. If the PADDLE program is altered to not just move the paddle, but also to keep track of the Y coordinates of the paddle through :Y, then PADDLETOUCH will function nicely:

```
TO PADDLE
CALL RC "A
IF :A = "E TELL [ 1 2 ]
  FORWARD 16 CALL :Y + 16 "Y
IF :A = "X TELL [ 1 2 ] BACK
  16 CALL :Y – 16 "Y
END
```

Unfortunately, this doesn't quite work as intended because it introduces a new bug: the CALL commands CALL :Y + 16 "Y and CALL :Y – 16 "Y will not work unless there is an initial value for :Y specified. Recall that the beginning value for the top of the paddle on the Y coordinate is 16 (as achieved in SETUP). Since this just happens once, it belongs in SETUP:

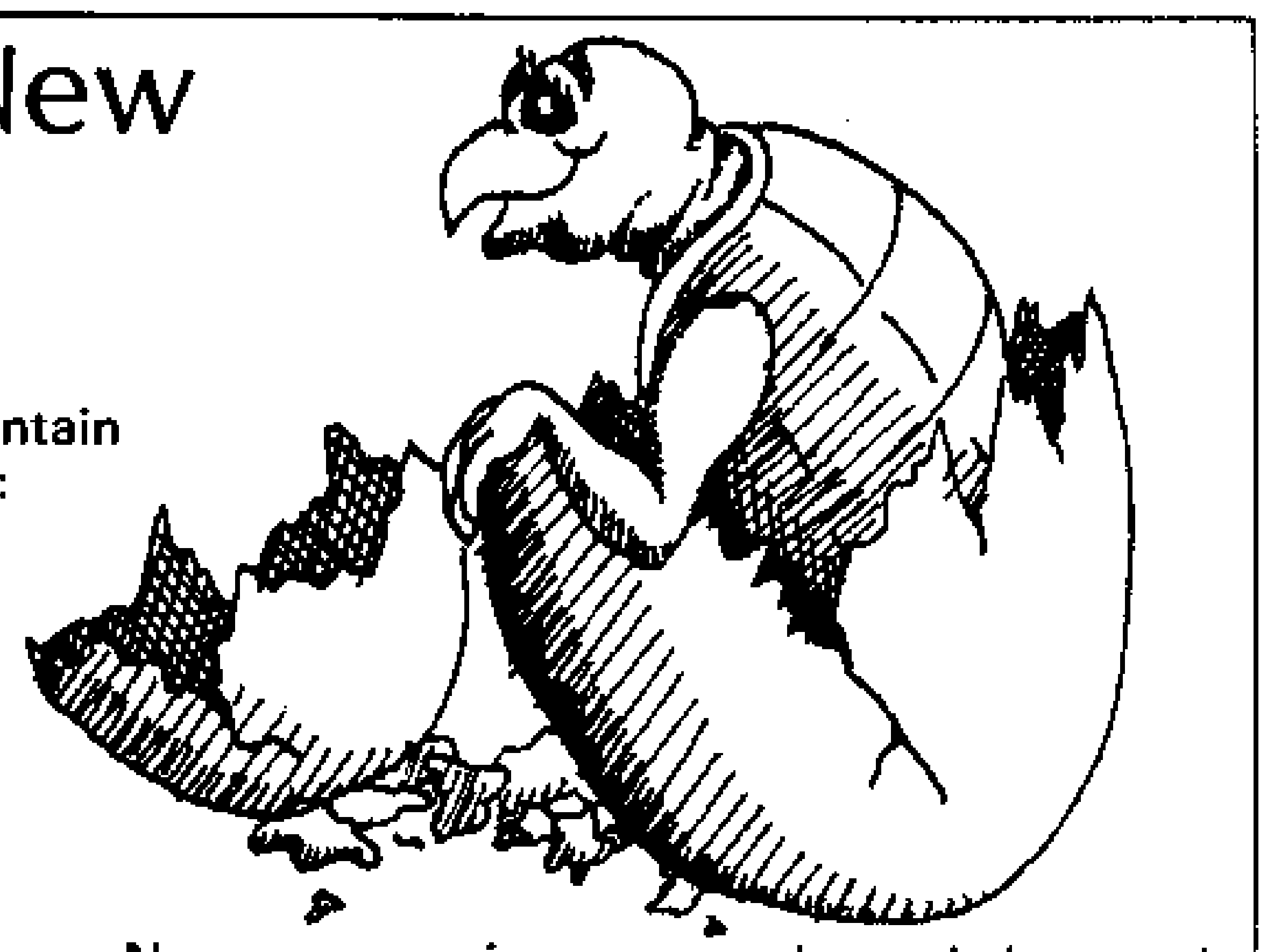
```
TO SETUP
TELL 0
CARRY :BALL
SETCOLOR :BLUE
HOME
SETHEADING 90
SETSPEED 15
TELL [ 1 2 ]
CARRY :BOX
SETHEADING 0
SETCOLOR :BLACK
SXY 100 0
TELL 2 SY 16
CALL 16 "Y
END
```

Next, it is trivial to both make the flight of the ball less predictable and to tie PADDLETOUCH into the GAME program. First to relate PADDLETOUCH, the BOUNCE2 program is altered:

```
TO BOUNCE2
TELL 0
TEST XCOR > 85
IFT CHECK
.
.
END
```

Continued on p. 81

The Birth Of A New LOGO



TI LOGO II, available this fall, will contain quite a few enhancements—most notably:

- **MUSIC**
  New commands include NOTE, DRUM, REST, SETVOICE, SETVOLUME, SETTEMPO, STACCATO, LEGATO, CHROMATIC, MAJOR, PLAYMUSIC, LOOPMUSIC, and PLAYNOTE.
- **Out-of-Space Warning**
- **Big Sprites of 32 x 32 Pixels**
- **Double the User-Memory**
- **Auto-Entry of Variable Name from Procedure Title Line**
- **RS232 Output Option**

New non-music commands and keywords associated with the enhanced features listed above include .HELP, BIG, SMALL, SIZE, REVERSE [list], ROTATE [list], LENGTH [list], TRUE, FALSE, .GC (for memory clean-up, .NODES (gives space remaining), .BAUD [value] (either 9600, 2400, 1200, or 300), and .COL [value] (for setting 32 to 132 columns in an RS232 device such as a printer.)

Letters on LOGO

Dear Sir:

Several years ago, a group of parents seeking enrichment for their children began exploring various extra-curricular options available to them both within and outside the community. Their different needs and expectations were brought to the East Aurora Community Resource Council. Parents, students, teachers, business and professional people, school board members and administrators collectively worked to develop a program of Saturday morning enrichment classes called East Oz. It was decided that Oz should be primarily for children, but also one in which all ages and interests could fulfill expression. A prime objective has been to achieve excellence in disciplines of high interest while never losing sight of the element of pure fun. The phrase "where anything can be" is linked to East Oz's name to allude to the special blend of excitement, enthusiasm and spontaneity that offerings have achieved.

Being an educator and computer enthusiast for nearly fourteen years, I thought East Oz would provide an ideal setting for TI LOGO. During the months of Febuary and March, a series of workshops were scheduled. Forty children grouped in two sections (twenty each—grades 2 through 5, grades 6 through 8) met for 90-minute sessions on five consecutive Saturday mornings. Seven TI-99/4A LOGO systems were utilized. Instruction was provided by myself. Assisting were Dr. David Farr, Director of the Microcomputer Educational Laboratory at State University of New York at Buffalo (UB) and David Padowski, TI Field Sales Engineer. Curriculum material was obtained from different sources including that provided by Dan Watt and the MIT LOGO Group.

Not knowing what to expect would be an understatement. My experience in teaching has been at the high school level, in particular 12th grade. All those involved were excited about the potential. Advanced publicity prepared the community for the coming of the TURTLE. Interest was noticeably visible. Registration period was to last one week, but both sections were filled to capacity through mail pre-registration . Needless to say, the participation was intense and electrifying. After learning primatives and how to define procedures, the children spent considerable time just discovering. My purpose for writing is not to describe various TI LOGO features nor expand upon its philosophical foundations. Others have done that quite well in previous 99'er Magazine articles. My observations of the Oz Workshops indicate that TI LOGO plus kids results in a positive educational experience. It is indeed suitable for children—a natural for our video culture.

Not only does TI LOGO provide young people with what Papert suggests, but it also serves as the means for teaching them computer literacy. Programming is the epitome of self-learning. Children have the opportunity to see their thoughts displayed on a T.V. or monitor screen (a picture is worth a thousand words). The Oz Workshops allowed students to work together, interactively with a computer and investigate various kinds of problems. So many computer educators have stated that programming is the best means to achieve problem-solving skills. Not only were *children* working together, but *parents* with sons and daughters—a rather refreshing experience. LOGO is suitable for home use; it is not limited to the classroom. It just might be more appropriate for the home, since educational funding restraints sometimes make computer equipment purchase restrictive (or impossible). It should be noted that during the period the Oz-LOGO program took place, an inservice computer course was taught to forty of the district's teachers. TI LOGO was used to introduce programming to the participants.

Because of its versatile features, the TI-99/4A has unlimited educational potential. It certainly is a learning device, an instructional aid, and a discovery machine. The East Oz LOGO project demonstrated that TI LOGO has something for all school-age children (because of its nature, BASIC doesn't). So much emphasis has been placed on a microcomputer's ability to provide children with entertainment. TI LOGO unveils a new dimension that can give kids so many different rewarding educational experiences.

Daniel R. Rozler
East Aurora, NY

Dynamic MANIPULATION OF Screen CHARACTER Graphics

By Fred Ellis
P.O. Box 777
Edinburg, TX 78539

Would you appreciate being able to write *shorter* programs that effectively do the same thing as longer ones? Or, would you enjoy watching the computer do a large amount of the tedious and boring designing, defining, and selecting of dozens of graphic characters—work that you would otherwise have to do yourself? If your answer to both of these questions is YES, read on fellow 99'er.

The scheme used in the TI-99/4A to represent screen character patterns with hexadecimal numbers is compact and convenient. Ingenious really. Compact because only 16 digits uniquely specify the on-off states of the 64 pixels in the 8 x 8 pixel character block. Such a system is certainly more satisfactory than display systems that only provide a small selection of predefined characters. It is convenient because the programming only requires simple statements of the form:

```
CALL CHAR(IJK,"0123456789AB
CDEF")
```

to define any 8 x 8 character imaginable. Likewise the statement

```
CALL HCHAR(ROW,COLUMN,IJK,
REPEAT)
```

will put character IJK anywhere on the screen. After a brief period, one is able to work intuitively with little conscious thought given to the format.

Yet, even with this system, there remains a considerable amount of tedious work to be done because every character we want on the screen (beyond the resident alphabet, etc.) must be defined and must be located. Doing this for many characters can mean lots of work as in Figure 1, where a graphic occupying less than half the screen contains 33 different characters. All 64 user-definable characters would use up 64 lines of code just to define; and if resident characters were redefined, we could end up having in memory a hundred or so program lines devoted to this one purpose.

In addition, there is the wear and tear on the programmer. He gets his ears burned if he leaves out one of those quote marks. Additional possibilities for errors include leaving out a comma or parenthesis or a pattern identifier string with more or less than 16 numbers, or a nonhexadecimal symbol accidentally typed in. Just type in four or five dozen CALL CHAR(IJK,"0123456789ABCD EF") statements and you will surely develop an acute case of boredom. Such static definition—with a program line for every new character and the resulting long list of CALL CHAR statements—is a lot of trouble and a source of errors.

It is also unnecessary. A little experimenting will show that we can define screen characters with data statements and a loop. Only a single CALL CHAR statement need be typed in and carried in memory. Such a method was used in the program which draws Figure 1. The program is given in Listing 1. The hexadecimal strings which define the screen characters to be used are in data statements starting at line 18. The loop starting at line 34 reads a data statement and puts the hexadecimal string it has picked up in a CALL CHAR statement. Thus the definition is sent off to graphic memory where it can be used later in the program as many times as needed. In this program, each data entry contains a comment to help one figure out what is happening on the screen. Each data entry contains three items: identification string, character number, and pattern-identifier string. On the next pass through the loop, another hexadecimal string is picked up and put in the CALL CHAR statement. Thus another defined screen character is sent off to memory.

After the program has cycled the last time through the loop, all the screen characters described in the data statements are in memory. They are now available using CALL HCHAR or CALL VCHAR statements just as if the program had run through dozens of CALL CHAR lines. Fewer program lines have been used, the possibility or errors reduced, and life has been made much easier for the programmer.

In a similar manner, characters are located on the screen beginning at line 65. For this application the data entries have the form: identification string, row number, column number, character number. The identification string serves only as documentation. The loop at line 85 puts this information in a CALL HCHAR statement which then sends it off to the video display processor. All characters will now appear on the screen at their assigned locations. Of course, the information we have in data statements could also be stored on a floppy disk.

Dynamically defining characters and putting them on the screen with data statements and loops (1) saves program lines and effort, (2) reduces errors, and (3) if documentation is added, can make a program easier to follow. No special attempt has been made to reduce the memory required for this program. The information in data statements could be packed tighter by omitting identification. Also, we could incorporate the number of repetitions in the data statements.

Another opportunity for making character definiton and placement a part of program dynamics occurs in plotting bar graphs. Bar graphs are a frequent application for computer graphics, and they look terrific on the color monitor.

On the TI-99/4A it is easy to plot a bar (Y characters high) by just using CALL VCHAR(ROW,COLUMN,IJK,Y). But the resolution will be very poor because we can only adjust the bar height in increments of one full character, which on the 13-inch monitor is about 3/8 of an inch. Ideally we would like the bar height to be continuously adjustable, but this infinite resolution cannot be realized with raster-scan systems. We can, however, get resolution equal to the pixel height. Toward this end we will define eight screen characters as shown in Figure 2. The first character has the bottom row of pixels turned on, the next one has the bottom two rows turned on, etc. The eighth character has all pixels turned on.

Figure 1.

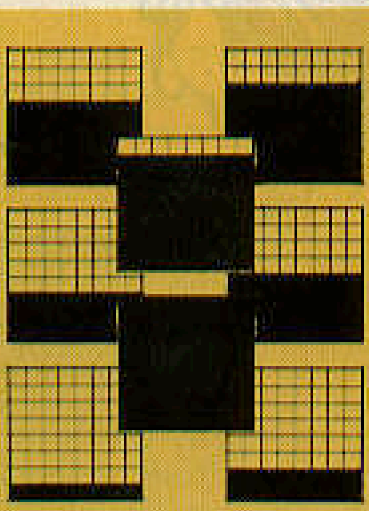


Figure 2.

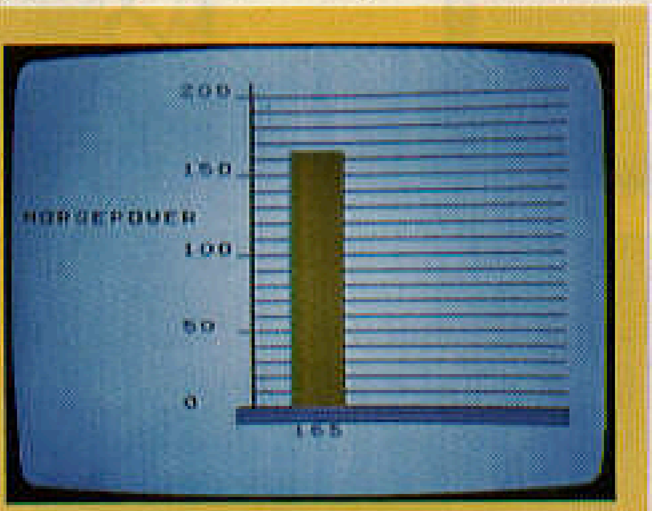


Figure 3.

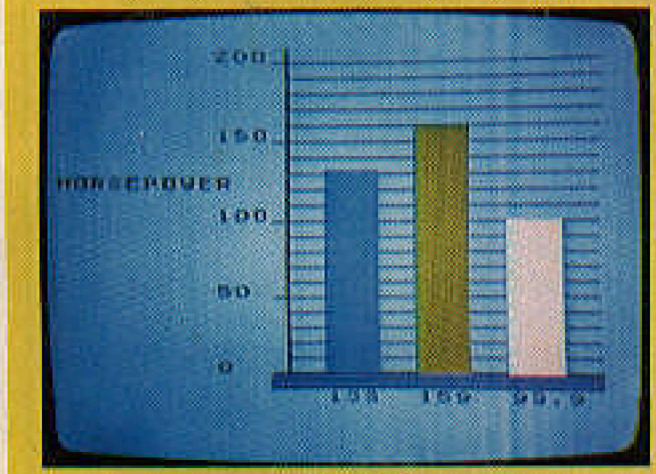


Figure 4.

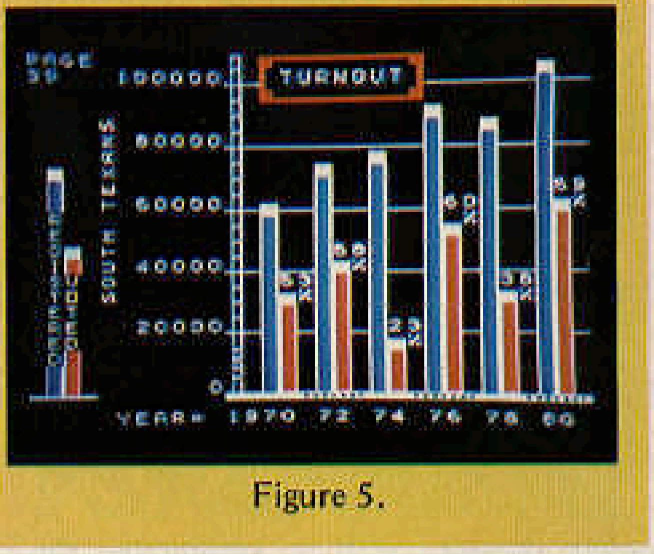


Figure 5.

Figure 1. Many different characters can mean lots of work for the programmer.
Figure 2. Screen characters used for one-pixel resolution in bar height.
Figure 3. Bar graph with one-pixel resolution.
Figure 4. Three variables plotted with one-pixel resolution.
Figure 5. An example of 99/4 graphics.

These characters are then used as bar tops. Stick the right one on top of your bar graph and you have resolution of one pixel (which is 1/8 of a character)—quite satisfactory with existing CRT's. On the 13-inch monitor this height increment is about 3/64 of an inch.

The bar graph in Figure 3 is plotted by the program in Listing 2 which uses this method. The characters available for use as bar tops are defined beginning at line 27. Scale of one character = 10 units is applied to the value entered at the keyboard starting at line 61. The integral value of Y is found and the remainder used to select the bar top character needed. The actual selection is done by the ON GOTO statement at line 69.

This program does work, but represents a brute force approach. If there is only one bar on the graph, then only *one* character will be used at the bar top. Yet *eight* bar-top characters have been defined and are sitting in memory. To take an extreme case, suppose we have four variables to be represented by four bars of different colors. Here, 32 characters must be defined and available for use as bar tops, yet only four bar-top characters will actually be used. Besides taking up memory, we have used half of the user-defined characters. This approach is wasteful. Why define characters that sit in memory but are never used?

Let's try a better idea by devising a program that defines bar-top characters *after* reading the data. Then it can define only characters that are needed. In other words, the data determine what bar-top characters are defined. To do this, we will have in the program a "master string" containing fourteen zeroes and sixteen F's. Segments exactly sixteen spaces long can be taken from this master string with a SEG$ statement. Next, the segment can be used as the pattern-identifier string and put in a CALL CHAR statement to define a bar top. Where will these 16-space segments start? Well, the data determine that matter. Thus, the data can cause a character with the first row of pixels turned on to be defined, or a character with the second row turned on, etc.

A possible coding to do this might be as follows:

```
110 MASTER$="00000000000000
     FFFFFFFFFFFFFFFF"
115 REMAINDER=BARHEIGHT-
     INT(BARHEIGHT)
120 TOPPATTERN=INT(REMAIN
     DER*8+.5)+1
130 STARTPOSITION=2*TOPPAT
     TERN-1
140 TOPPATTERN$=SEG$(MAST
     ERS,STARTPOSITION,16)
150 CALL CHAR(97,TOPPATTER
     NS)
160 CALL HCHAR(21-Y,16,97,3)
```

Here the 21 in 21−Y allows the bar to be up to 20 rows high.

Suppose, for example, the data calls for a bar top with the bottom two rows turned on. Then TOPPATTERN will be 2. Then STARTPOSITION = 3. Then the pattern-identifier string created in line 140 will be

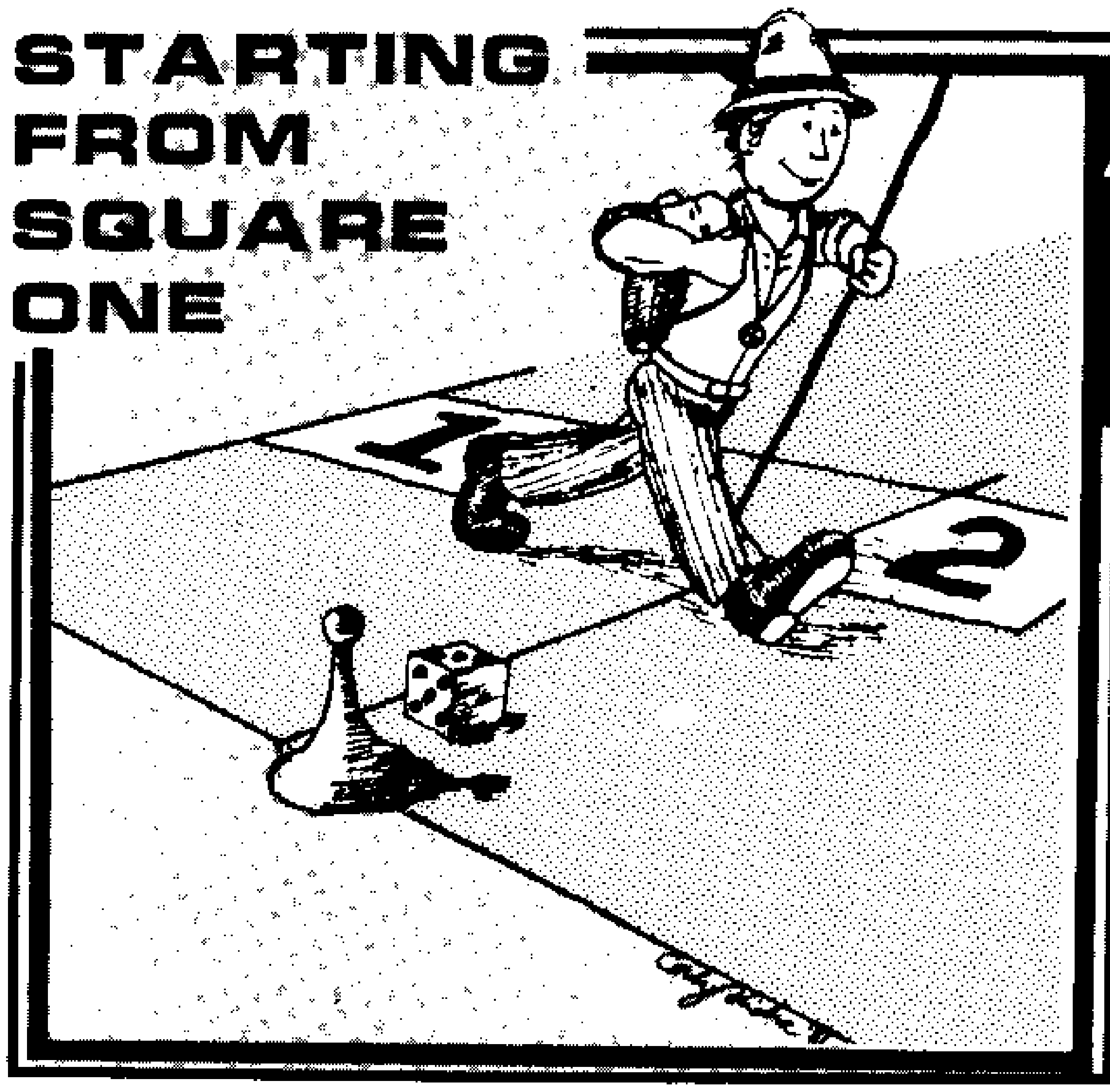
```
TOPPATTERN$="000000000000F
FFF"
```

(as you can see—if you will take the trouble to count this off—starting at the third position in the master string). The resulting screen character which is defined in line 150 will be one with the bottom two rows of pixels turned on. As the program runs, we want each datum to determine where the 16-space segment will begin. Thus we have used the remainder to calculate STARTPOSITION. By notching back and forth with STARTPOSITION, the routine will define any character needed to top off a bar.

With this particular routine there will be a little problem associated with rounding up to the next higher grid line on the next higher row. For instance, if the scale used is 1 character = 10 units, we would want 99.9 to appear on the graph as 100. Another problem (I didn't say this was too simple) involves the character to be used for the body of the bar. This character must have all pixels turned on, but the routine above will not create such a character for all values of the data set.

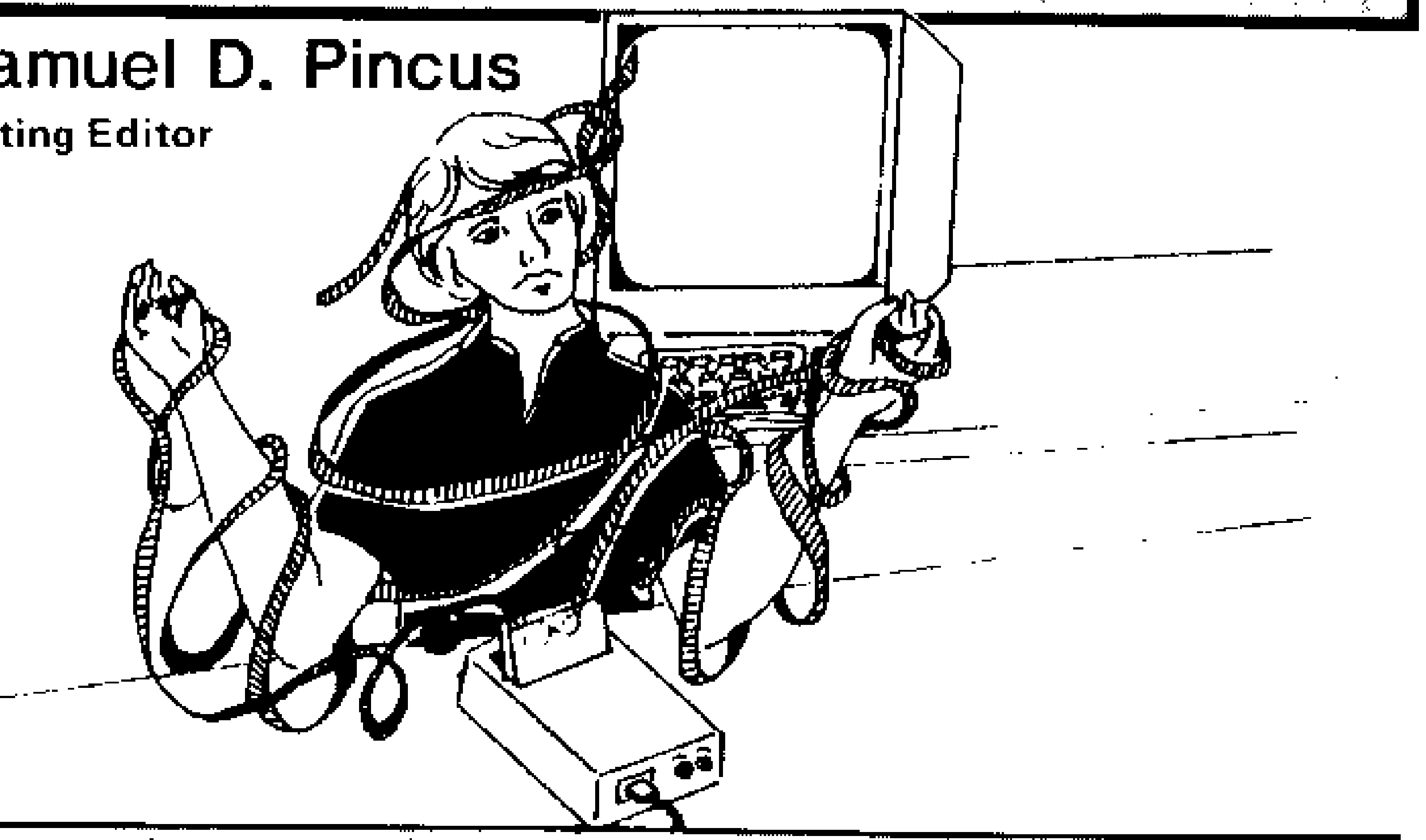
A program in which these problems are solved is given in Listing 3. A routine similar to the one above starts at line 66. Character 96, which is used for the body of the bar, is defined earlier in the program. Note this master string contains 18 F's. (If you try this program, you better count them *carefully*.) TOPPATTERN = 9 will pick up the extra F's at the 17th and 18th positions.

Continued on p. 77



A Beginner's Guide To Cassette Operation With A Home Computer

By Samuel D. Pincus
Contributing Editor



You bought your TI-99/4A Home Computer because the plug-in Command Modules looked like a quick and easy way to get started. You played the games and typed in the programs that you found in the Users Reference Guide. Now comes the moment of truth—*What to do next?* The answer, fellow 99'ers, is easy: Learn how to use a cassette tape recorder with your computer so that you can begin to build up a program library by recording and saving the many excellent software programs that appear printed in *99'er Magazine*.

In order to get started, you'll need a tape recorder cable, a cassette tape recorder, and some good quality cassette tapes. The cable assembly is specially manufactured for use with the TI-99/4A Home Computer (Dual Cassette Cable, PHA2000) and should be available from any TI retailer. Cables made for other computers will *not* work with your machine.

When it comes to locating a tape recorder, the matter isn't as straightforward. TI does not manufacture a special tape recorder for use with their computer and really doesn't recommend any one particular brand. There is a good reason for this, because finding a recorder that provides satisfactory results is not as easy as you'd think. To explain why, I will have to give you a quick background on how a computer talks to a tape recorder and vice versa.

The first thing that you must realize is that a computer is very, very dumb! While your brain can understand things like "yes," "no," and "maybe," a computer only understands "off" and "on." Everything that a computer does is based on the fact that it understands only these two things.

What the Recorder Records

In order to do the wonderful things your computer is capable of doing, the "offs" and "ons" have to be arranged into patterns that the computer can use. This is true not only for numbers, but for letters as well. For example, if you type in the letter "A" on the keyboard, your TI-99/4A really sees a pattern that looks like this: on-off-off-off-off-off-off-on. If we think of an "off" as a zero and an "on" as a one, the pattern looks like this: 10000001. Remember that everything your TI-99/4A does is based on groups of binary numbers like that. Each 1 or 0 is called a "bit." In addition, every pattern of ones and zeros has its own binary value.

Learning to count in binary is beyond the scope of this article, but there are a number of books or articles around that can teach it to you. What you should know for now is that each letter and character has its own pattern of zeros and ones and its own binary value. In order to make it easier to communicate with a computer, the bits are grouped into groups of four. Two groups of four bits is called a "byte." A byte can contain 256 different combinations of bits (or values) and has enough combinations to allow a unique pattern for each letter, number and character on your keyboard. For example, the 65th possible pattern (a byte value = 65) represents the letter "A" in the ASCII character coding system used by the TI-99/4A and most computers. This means that 65 is the ASCII value of the letter "A." That is why the computer will give you back an answer of 65 if you ask for the value of ASC("A").

To make a word, the letters are strung together (just like in English). When you type in a word as part of a TI BASIC program, the computer will convert each letter in the word into a byte and make sure that this particular grouping of bytes makes sense. For example, if you type in GOTO on the keyboard, the computer will read that as 71 79 84 79.

In order to read data from a tape recorder, your computer will have to be able to read in bytes of data. That means that it will have to understand "offs" and "ons" when listening to the tape. Unfortunately, there is no such thing as an "on" or an "off" to a tape recorder. Even when it is absolutely quiet to your ears, a tape contains some amount of noise. So we can't say that no noise = "off" and noise = "on." Instead, we need another way for a tape recorder to communicate an "off" or "on" to the computer. This is done by using two tones, each at a different frequency. The lower frequency tone can then mean "off" and the higher frequency tone will mean "on." If your tape recorder cannot record or playback the higher frequency, your computer would only hear "offs."

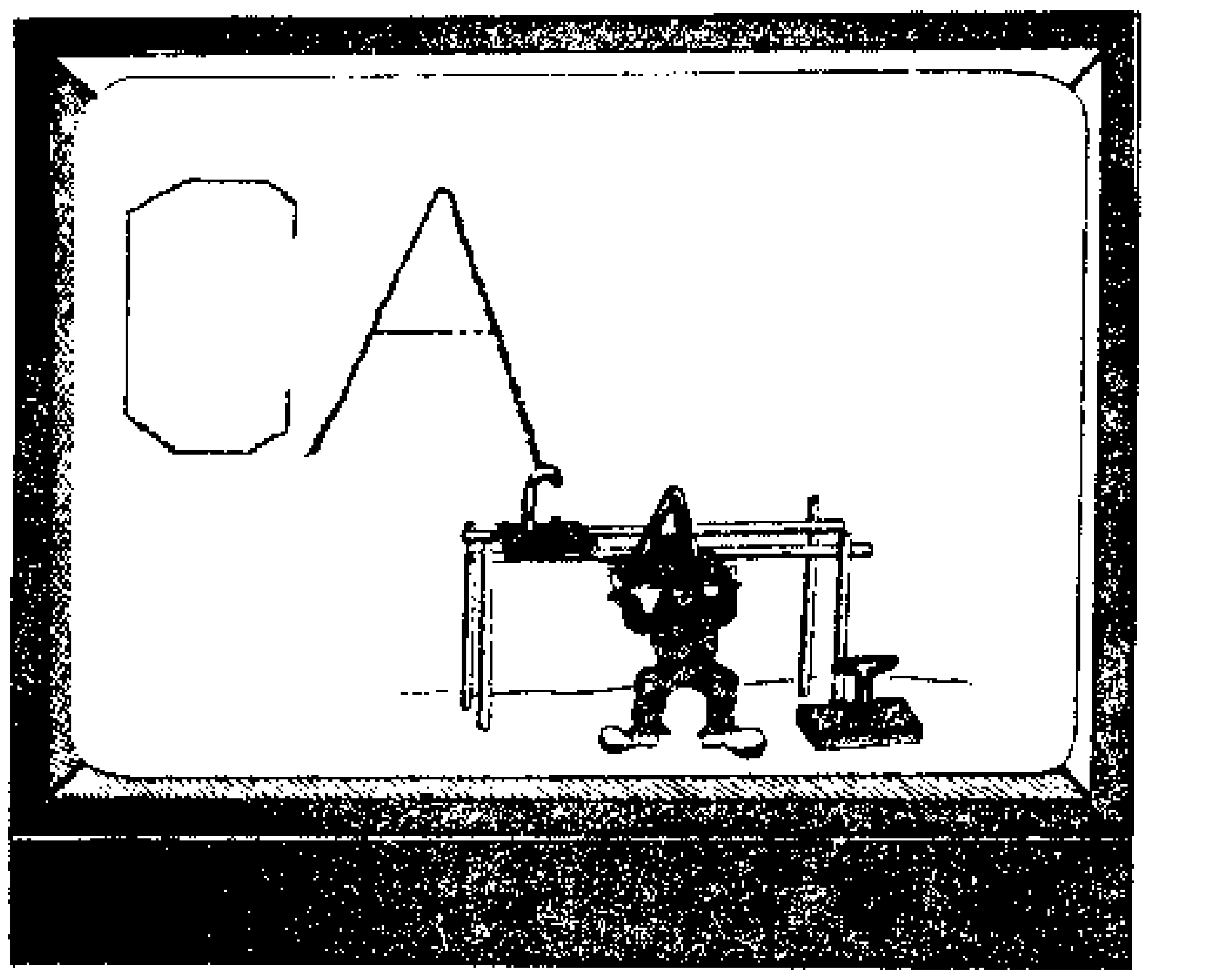
Only these two tones are recognized by the computer. If it "hears" any other tone, your 99/4A will ignore it. Of course, TI has added some tolerance to the computer so that if it hears something almost like the "on" or "off" tone, it will accept it as an "on" or "off." This means that for your 99/4A to properly read data from a cassette recorder, the tone must be both loud enough (i.e., the volume set high enough) to be heard and also be at the right frequency.

Not all Recorders Are Equal

A principal law of physics states that it takes more power to produce or reproduce a high frequency than it does to produce or reproduce a lower frequency tone. If the volume is not high enough during either recording or playback, your computer won't hear anything, or it might not be able to hear the higher frequency tone. Alternately, it may hear all of the lower

Continued on p. 78

PRE-SCHOOL BLOCK LETTERS AND DATA COMPACTION



By Howard G. Drake
239 W. Main Street
Westboro, MA 01581

Most kids aged 100 weeks to 100 years old are fascinated by computers. And small kids are really fascinated by a computer's video screen; it's like a TV but they can control it. My kids were just learning the alphabet and when they could wriggle in between Dad and the computer, they would push the "A" key and an"A" would pop onto the screen. But the popping part was the problem. A computer doesn't draw (write) letters on the video screen, it "pops" up the whole letter at once. (Or at least to our slow eyes it "pops" the whole letter at once.) But kids can't just squeeze a crayon and have a letter pop onto a piece of paper. Kids have to learn a series of hand motions in order to make a recognizable "A" on a piece of paper.

But just maybe the computer could make large letters by popping short line segments in sequence onto the screen . . . This was the start of my idea. The finished product is the program that follows, *Pre-School Block Letters.* And the intervening (gory) details are about data compaction.

Most home computers don't have point addressable graphics, but they do have character graphics that can produce line segments at various angles. Thus, I thought, I would build the letters and numbers from short line segments. Easier said than done. What I really needed to build the letters and numbers was some help. Fortunately, my wife, a teacher, re-taught me the correct way to form letters; I, in turn, taught the computer.

Then I had to store about 3500 pieces of information concerned with what line segment goes where to form each letter and number. Each piece of information as a number requires 4 to 8 bytes depending on the computer. But *strings* require only one byte for each stored character. And between letters, numbers, and punctuation marks, there are enough different characters so that over 40 unique values can be stored using only one byte per value.

Furthermore, strings can be very long, so this helps hold down the overhead to identify each string. Thus, to change the piece of information to the value of a valid ASCII character, I just added a constant onto each piece of information. The characters were therefore grouped into strings, and the strings stored in DATA statements. The SEG$ and ASC functions retrieve the information as required. And that's how computers came to *draw* letters rather than *popping* letters.

Note: Make sure ALPHA LOCK IS DOWN.

EXPLANATION OF THE PROGRAM
(Pre-school Block Letters)

Line Nos.	
130-220	Program Initialization.
230-310	Scan keyboard looking for a letter or number key.
320-340	Change ASCII code to number between 0 and 35.
370-470	Draw line segments of letter or number in an array.
540-570	Store geometry of "W" in array.
1680-2020	Define line segment characters used to make letters and numbers.
2030-2080	Input word from user.
2090-2300	Have little man hold up letter.
2310-2410	Get key pushed. If it matches letter that man is holding, then draw letter.

```
100 REM   ** PRE-SCHOOL BLOCK LETTERS **
110 REM   *     BY HOWARD G. DRAKE    *
120 REM   **  99'ER VERSION 1.6.1     **
130 CALL SCREEN(8)
140 OPTION BASE 0
150 DIM AR$(35,2)
160 GOSUB 1680
170 GOSUB 480
180 CALL CLEAR
190 INPUT "DO YOU WANT TO WORK WITH    WHOLE WORDS
    RATHER THAN     INDIVIDUAL LETTERS ? ":ANS$
200 ANS$=SEG$(ANS$,1,1)
210 IF ANS$="Y" THEN 2030
220 CALL CLEAR
230 PRINT "PRESS A LETTER OR NUMBER KEY"
240 GOSUB 260
250 GOTO 230
260 CALL KEY(0,KEE,STATUS)
270 IF STATUS=0 THEN 260
280 IF KEE<48 THEN 260
290 IF KEE>90 THEN 260
300 IF KEE<58 THEN 330
310 IF KEE>64 THEN 320
320 KEE=KEE-7
330 S1=KEE-48
340 CALL CLEAR
350 FOR DELAY=1 TO 10
360 NEXT DELAY
370 FOR I=1 TO LEN(AR$(S1,1))
380 REM
390 COL=ASC(SEG$(AR$(S1,1),I,1))-40
400 ROW=ASC(SEG$(AR$(S1,2),I,1))-42
410 C=ASC(SEG$(AR$(S1,0),I,1))+63
420 CALL SOUND(100,300,2)
430 CALL HCHAR(ROW,COL,C)
440 FOR DELAY=1 TO 10
450 NEXT DELAY
460 NEXT I
470 RETURN
480 REM
490 REM
500 RESTORE 590
510 FOR S1=0 TO 35
520 READ AR$(S1,1),AR$(S1,2),AR$(S1,0)
530 NEXT S1
540 W1$="WXYWXYWXYWXY"
550 W2$="Z"&CHR$(91)&CHR$(92)
560 W3$=W2$&W2$&W2$&W2$
570 AR$(32,0)=W1$&W3$&W1$&W3$
580 RETURN
590 DATA "3210000000012345677777777654"
600 DATA "00123456789:;;;;:98765432100"
610 DATA "DECJKSSSSIHBFGEDCKJVVVVHIBGF"
620 DATA "000000000000"
630 DATA "0123456789:;"
640 DATA "TTTTTTTTTTTT"
650 DATA "0123456788765432100123456788"
660 DATA "2100000123456789:;<<<<<<<<"
670 DATA "JCEDRFGBIKCCCCCCCCRRRRRRRR"
680 DATA "01234567887655678887654321000"
690 DATA "0000000001234556789:;;;;;:98"
700 DATA "PPPPPPPPPCCCCFGBIUKCDEPGFBHI"
710 DATA "000000123456789666666666666"
720 DATA "0123455555555550123456789:;"
730 DATA "VVVVVVPPPPPPPPPVVVVVMVVVVVV"
740 DATA "0000012345677765432101234567"
750 DATA "123455555556789:;;;;;:0000000"
760 DATA "VVVVVEDRFGBIUKCDEPGFBPPPPPPP"
770 DATA "4433221100123456788 7654321"
780 DATA "0123456789:;;;;;:98777778"
790 DATA "JKJKJKJKVVBFGPEDCKIGFRRDEC"
800 DATA "0123455544332211 0"
810 DATA "000000123456789:;"
820 DATA "PPPPPPJKJKJKJKJ"
830 DATA "76543210012345678876543210012345677"
840 DATA "100000123455666789:;;;;;:9876655432"
850 DATA "BGFRDECJHBFGRFGBIKCDEPGFBHJCEDEDCKJ"
860 DATA "76543210012345678888888888"
870 DATA "1000001234444432234567 89:;"
880 DATA "BGFRDECJHFGPEDCCSSSSSSSSS"
890 DATA "554433221100667788 99:;;;345678"
900 DATA "0123456789:;0123456789:;666666"
910 DATA "JKJKJKJKJKJKIHIHIHIHIHIHRRRRRR"
920 DATA "00000000000012345678876543211234567887654321"
930 DATA "0123456789:;000000123455555566666789:;;;;;;"
940 DATA "VVVVVVVVVVVVRRRRFGBIKCDEPPPPRRRRFGBIKCDEPPPP"
950 DATA "887654321000000000123456788"
960 DATA "32100000123456789:;;;;;;98"
970 DATA "HIBGFRDECJKSSSSIHBFGPEDCKJ"
980 DATA "000000000000012345677888877654321"
990 DATA "0123456789:;0000012345678 9:;;;;;"
1000 DATA "VVVVVVVVVVVVRRRFGBIHITTKJKCDEPPP"
1010 DATA "00000000000001234567123451234567"
1020 DATA "0123456789:;000000055555;;;;;;;"
1030 DATA "VVVVVVVVVVVVRRRRRRPPPPPPPPPPPP"
1040 DATA "0000000000000123456712345"
1050 DATA "0123456789:;000000055555"
1060 DATA "VVVVVVVVVVVVRRRRRRRPPPPP"
1070 DATA "887654321000000000123456788887"
1080 DATA "32100000123456789:;;;;;:98777"
```

Continued on p. 76

COURSE DESIGNER AUTHORING PACKAGE: A REVIEW OF THE TEXAS INSTRUMENTS

Making computers easier to use has been both the goal and accomplishment of high-level programming languages. As computer hardware prices have fallen drastically, ever-increasing numbers of inexperienced people have gained access to these logical machines, a situation that has greatly accelerated the need for ease of use. Attempts at meeting this need in education have fostered the development and use of two approaches: (1) CAI (computer-assisted instruction) authoring languages and (2) CAI authoring systems.

In the language category, IBM has Coursewriter, Control Data's PLATO system has Tutor, and several microcomputers have PILOT. [A large part of the PLATO CAI software will be available for use on the TI-99/4A by the end of this year. TI PILOT, an implementation that runs under the UCSD p-System(TM) will be available even sooner—Ed.]

Authoring systems also exist on both large and small computers and are even easier to use than the authoring languages, because they gather lesson data and organize it according to questions asked by the computer of the course author, rather than require any programming (albeit a simplified form when using authoring languages).

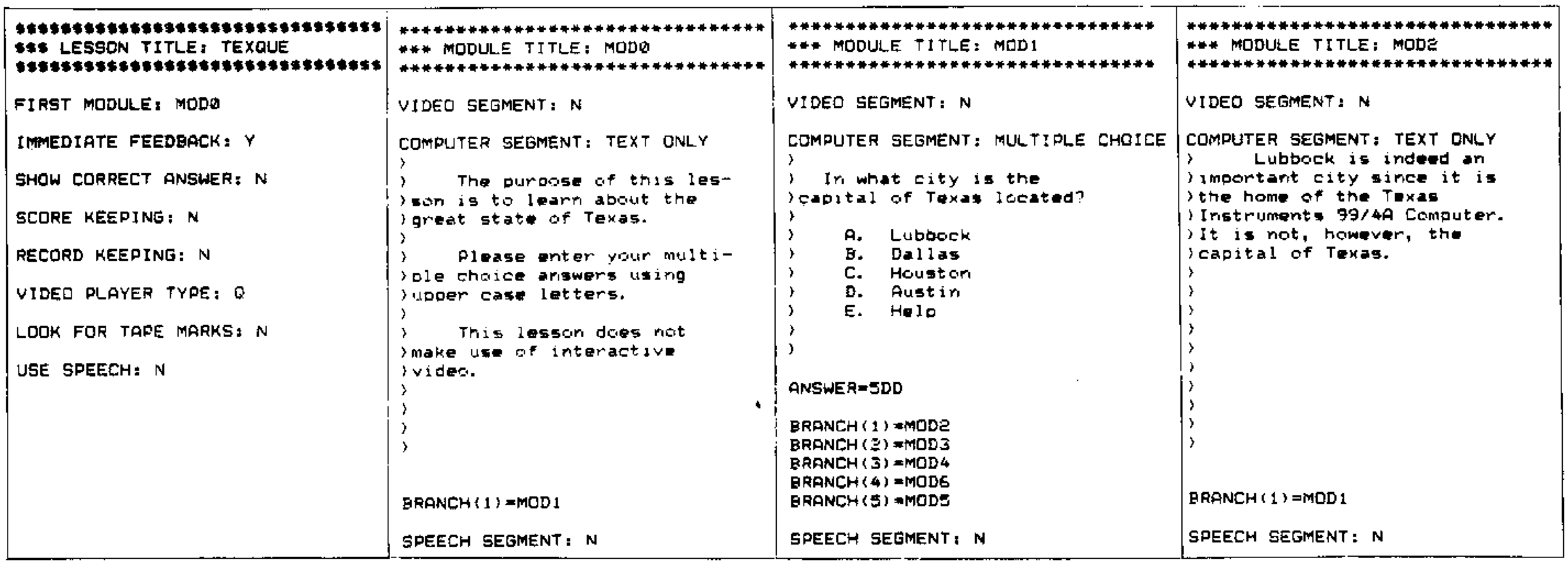
The purpose of using either of the two systems is understandable, but some CAI experts, such as Professor Alfred Bork of the University of California at Irvine, point out the lack of flexibility afforded by authoring systems, and move to high-level languages such as Pascal. He says, for example, that CAI lessons are best developed by a team consisting of a subject matter expert, a programmer, and a lesson reviewer. Other experts mention also the need for an instructional designer. It does not take long to realize that not everyone wishing to put a particular lesson on the computer has access to such a team. It is for this reason that authoring systems exist and new ones continue to be created. This, of course, provides the rationale for reviewing TI's new Course Designer Authoring Package (CDAP).

System Description and Features

Speech and text capabilities notwithstanding, CDAP was created by TI primarily to support their new Video Controller. It is written in TI Extended BASIC, requires at least one disk drive, and is menu-driven. Although one drive is sufficient, some disk swapping during lesson authoring is required—a practice not necessary when two drives are available. Programs are contained on the Designer and Lesson Diskettes, each of which contains the necessary files for Text-to-Speech. The speech is the same as that contained on the Terminal Emulator II, but requires the Memory Expansion peripheral in addition to the Speech Synthesizer. Authors can also produce lessons for use on systems not having the video controller or speech capability.

When used with the video controller, CDAP has the capability to use one-half inch industrial model video-cassette recorders such as the Sony Betamax, and Panasonic VHS machines, or three-quarter inch Panasonic and Sony U-Matic machines. In addition, the Pioneer VP-1000 videodisc player can also be used for the delivery of video materials. For each of these machines, the authoring system allows adequate previewing for lesson creation. Segments can be viewed as each lesson module is authored, with the desired video frame information being stored as an integral part of each module.

Although video presentation is the primary focus of the system, there is a fair amount of flexibility allowed for each module. The author can select from among several options: (1) No text, (2) Text only, (3) Multiple choice question, (4) Fill-in-the-blank question, and (5) Score evaluation. Video is available for use in Options (1), (3), and (4). The multiple choice option is useful because answers can be "judged" or "not judged." Choices that are "not judged"



```
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$$$ LESSON TITLE: TEXQUE
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

FIRST MODULE: MOD0

IMMEDIATE FEEDBACK: Y

SHOW CORRECT ANSWER: N

SCORE KEEPING: N

RECORD KEEPING: N

VIDEO PLAYER TYPE: Q

LOOK FOR TAPE MARKS: N

USE SPEECH: N
```

```
*********************************
*** MODULE TITLE: MOD0
*********************************

VIDEO SEGMENT: N

COMPUTER SEGMENT: TEXT ONLY
>
>     The purpose of this les-
>son is to learn about the
>great state of Texas.
>
>     Please enter your multi-
>ple choice answers using
>upper case letters.
>
>     This lesson does not
>make use of interactive
>video.
>
>
>
>


BRANCH(1)=MOD1

SPEECH SEGMENT: N
```

```
*********************************
*** MODULE TITLE: MOD1
*********************************

VIDEO SEGMENT: N

COMPUTER SEGMENT: MULTIPLE CHOICE
>
>  In what city is the
>capital of Texas located?
>
>     A.  Lubbock
>     B.  Dallas
>     C.  Houston
>     D.  Austin
>     E.  Help
>
>

ANSWER=5DD

BRANCH(1)=MOD2
BRANCH(2)=MOD3
BRANCH(3)=MOD4
BRANCH(4)=MOD6
BRANCH(5)=MOD5

SPEECH SEGMENT: N
```

```
*********************************
*** MODULE TITLE: MOD2
*********************************

VIDEO SEGMENT: N

COMPUTER SEGMENT: TEXT ONLY
>      Lubbock is indeed an
>important city since it is
>the home of the Texas
>Instruments 99/4A Computer.
>It is not, however, the
>capital of Texas.
>
>
>
>
>
>
>
>
>
>


BRANCH(1)=MOD1

SPEECH SEGMENT: N
```

LESSON-DEVELOPMENT SOFTWARE

By Michael Bush
1933 Dublin Blvd.
Colorado Springs, CO 80918

allow the student to choose the modules to be presented next. These can be "help" segments or specialized material that interests the student. The fill-in-the-blank capability is quite flexible in that several alternative answers can be judged as correct. The score evaluation option examines the number of correct answers obtained prior to the current module, and thus allows the author to present material based on the success of the student.

Text-to-speech takes advantage of TI-99/4A capabilities and adds an interesting dimension to the presentation of material. Those familiar with TE II will recognize the speech as the same, with the exception of the initial delay time resulting from the loading of the rules and allophones into the Memory Expansion. While the speech does sound robot-like, there are certainly applications that would be enhanced with this capability.

Authoring

As mentioned above, the system is menu-driven and quite easy to use. The function to be accomplished (authoring or lesson execution) is determined by the diskette loaded in Drive 1 when Extended BASIC is selected. For authoring, six functions are provided: (1) Initialize system, (2) Set-up lesson, (3) Create/edit modules, (4) Print modules for review, (5) Generate program, and (6) Mark tape. Upon selection, each option is self-explanatory, but if any uncertainty does linger in the mind of the user, each screen is referenced by a numbered explanation in the user's manual.

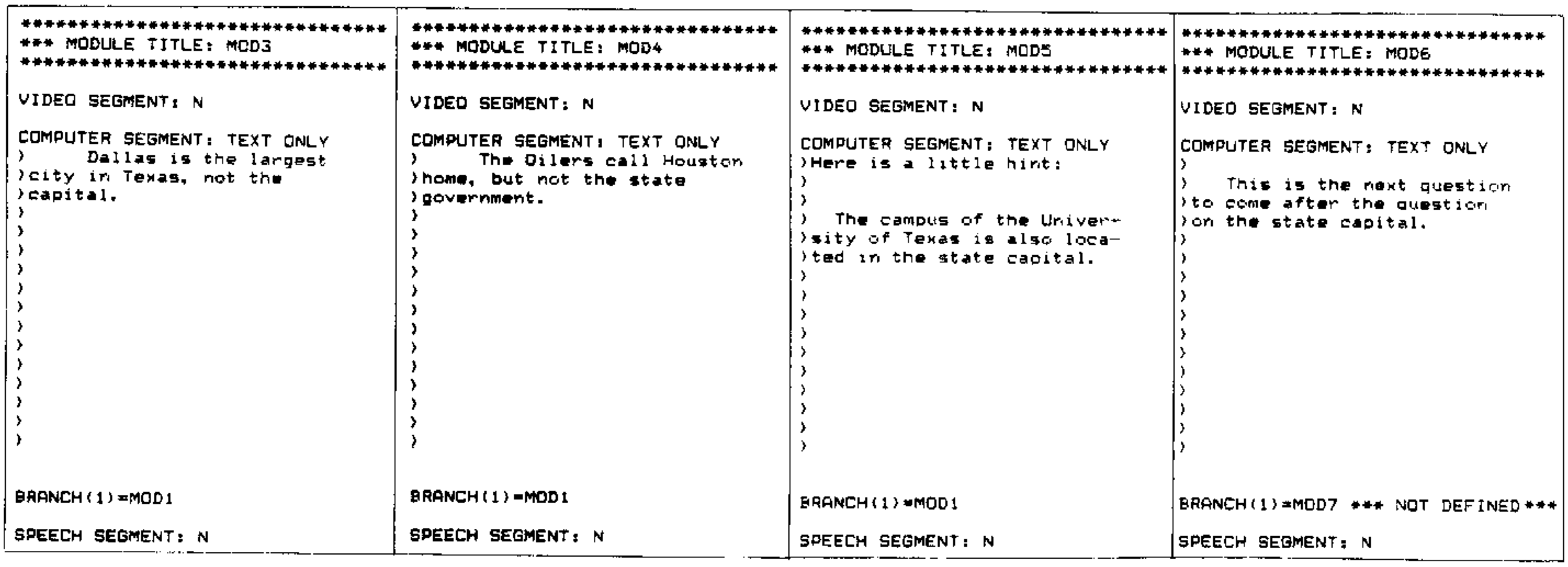
The parameters for system operation are set during the execution of the option, "Initialize System." In effect, the appropriate hardware configuration is established once, for use throughout the CDAP programs. Text-to-speech activation, the type of video player to be used, the printer device attached, and the number of disk drives in use are the principal elements of the questions answered during initialization—a process that takes only a couple of minutes.

The "Set-up lesson" option provides information to the system relative to a particular lesson and how it will be executed for the student. Each lesson is assigned a name which is used for creating a file on the student disk. This file contains the name of the first module in the lesson and information on whether or not: (1) immediate feedback is provided to the student, (2) the student score is displayed on the screen during the lesson, and (3) records are kept (and if so, whether they are to be kept on disk or on the printer).

Modules are created and updated using the "Create/edit" option, with each module title becoming a file that contains parameters relative to the nature of the module's presentation to the student. As is done throughout CDAP, questions are asked of the author at each step of the way to allow for selection of the module content (question, explanation, video, speech, score evaluation, etc.) The text for speech or screen presentation is prepared on the screen using an effective editor that shows the text portions exactly as they will appear to the student.

Printing modules for review allows the author to examine each module as well as module branching. Single modules or all modules for a particular lesson can be printed, thus allowing for creation of story boards. A sample printed copy of a lesson is provided below.

Of particular interest is the "Generate program" capability. As each module is authored, data are stored as files on disk under the module name. During normal execution, these are referenced by disk accesses from the "Student" program with corresponding delay time between modules. The "Generate" program option circumvents these delays by writing DATA statements into a file that can later be MERGED into the



```
********************************
*** MODULE TITLE: MOD3
********************************

VIDEO SEGMENT: N

COMPUTER SEGMENT: TEXT ONLY
>     Dallas is the largest
>city in Texas, not the
>capital.
>
>
>
>
>
>
>
>
>
>
>
>
>


BRANCH(1)=MOD1

SPEECH SEGMENT: N
```

```
********************************
*** MODULE TITLE: MOD4
********************************

VIDEO SEGMENT: N

COMPUTER SEGMENT: TEXT ONLY
>     The Oilers call Houston
>home, but not the state
>government.
>
>
>
>
>
>
>
>
>
>
>
>
>


BRANCH(1)=MOD1

SPEECH SEGMENT: N
```

```
********************************
*** MODULE TITLE: MOD5
********************************

VIDEO SEGMENT: N

COMPUTER SEGMENT: TEXT ONLY
>Here is a little hint:
>
>
>  The campus of the Univer-
>sity of Texas is also loca-
>ted in the state capital.
>
>
>
>
>
>
>
>
>
>


BRANCH(1)=MOD1

SPEECH SEGMENT: N
```

```
********************************
*** MODULE TITLE: MOD6
********************************

VIDEO SEGMENT: N

COMPUTER SEGMENT: TEXT ONLY
>
>   This is the next question
>to come after the question
>on the state capital.
>
>
>
>
>
>
>
>
>
>
>
>


BRANCH(1)=MOD7 *** NOT DEFINED ***

SPEECH SEGMENT: N
```

"Driver" program for delay-free execution.

Another noteworthy capability of the authoring system is the "Mark tape" option. When using video-cassettes, it is impossible to create tape copies that have leaders of exactly the same lengths. Using this option, the tape is marked on the dub track at the beginning of the desired material. When the lesson is executed and this option has been selected by the author, the computer will automatically adjust all tape addresses as necessary. An additional benefit is gained in that the tape will be read during video presentation with comparisons between where the computer thinks the tape is, and where, in reality, it *actually* is. Adjustments are thus made for slippage in the videotape on the player during and between accesses. This error correction will insure that the desired video segments are always shown.

Lesson Execution

With the "Lesson" diskette inserted, the student receives a menu that lists the lessons available on that particular disk. In the version reviewed, it was not yet possible to select the programs that had been generated from previously created lesson modules. This would certainly be helpful for the non-experienced user who does not know how to use the Autoload function of Extended BASIC. Use of the "Driver" program combined with the "Generate" function also removes the speech capability, since the program can be stored on the audio "dub" track of the VCR. This is a positive aspect in the sense that student stations need not have a disk drive. Yet where drives are available and the "Generate" function is desired for increased lesson execution speed, speech should be made available.

During a student session, records can be kept using disk or printer as indicated by the author during lesson creation. If kept on disk, then a report generation program function would be helpful for the teacher to track student progress. A "Record-dump" program is included on the "Lesson" diskette provided with the package being reviewed, but it is merely a short program to read the data file created by the "Student" or "Driver" programs. At least a minimally functioning program to format file contents would be helpful for the "non-programming" user—especially since authoring systems exist, in large part, for use by the non-programmer. [One might argue that the mere fact that the "Record-dump" program prints (1) a list of all modules attempted, (2) the students response, and (3) whether the response is right or wrong, does serve as minimal requirements—Ed.]

General Impressions

It is fair to assess the system as "friendly" due to the clear use of menus. Where the need arises, it is always possible to refer back to the manual using the screen reference number. Throughout the system operation, good use is made of the BACK, REDO, and PROC'D functions. During editing, functions are provided to allow screen editing of the text (and speech) to be presented to the student. Effective use is also made of the program chaining capability provided by Extended BASIC. In essence, it is difficult to go wrong when authoring a lesson.

The powerful notion of branching upon various responses is included in the system and is easy to use. It is possible to branch upon any multiple-choice response to a module suited to the needs of the student. The ability to allow freedom of choice to the student is recommended by CAI experts and incorporated in the system with the "no judge" option. Fill-in-the-blank answers can take up as much as two lines and consist of several possible ways to answer the question correctly.

As mentioned earlier, the primary purpose of the system is to allow use of the Video-Controller. The authoring of modules that use video segments is surprisingly easy. When video segments are needed, the system prompts the user through the steps to properly identify necessary data for accurate replay of video sequences. With the capability to preview segments during module preparation, the author is assured that the desired material will be presented.

One desirable feature included in some other authoring packages for microcomputers, but not in CDAP, is the ability to combine computer-generated graphics with the lesson presentation. It would be hoped that future versions would incorporate this capability—especially given the new graphics capabilities of the TI-99/4A. [The designer of CDAP has not included computer-generated graphics because he felt that (1) the human factors required for this task would be awkward in Extended BASIC, and (2) the external video that the package relies on is a more effective means to present graphics—Ed.]

Summary

All in all, CDAP is a usable piece of software that does what it was designed to do: allow easy authoring of primarily video-based lessons. In comparing it to a much more expensive system that runs on another machine, the author of CDAP describes it as the—"Super Beetle" of authoring programs—not extremely snazzy, but certainly functional.

Pre-School . . . from p. 73

```
1090 DATA "HIBGFRDECJKSSSSIHBFGPEDCKJVLR"
1100 DATA "0000000000007777777777123456"
1110 DATA "0123456789:;0123456789:;555555"
1120 DATA "VVVVVVVVVVVVSSSSSSSSSSSPPPPPP"
1130 DATA "00000000000000101"
1140 DATA "0123456789:;00;;"
1150 DATA "VVVVVVVVVVVVLRMP"
1160 DATA "66666666665432100567"
1170 DATA "123456789:;<<<;:9000"
1180 DATA "UUUUUUUUUKCDQFBHTPPP"
1190 DATA "0000000000001234563344 5566"
1200 DATA "0123456789:;5432104567 89:;"
1210 DATA "VVVVVVVVVVVVCCCCCIHIHIHIH"
1220 DATA "0000000000001234567"
1230 DATA "0123456789:;;;;;;;;"
1240 DATA "VVVVVVVVVVVVPPPPPPP"
1250 DATA "000000000000123487659999999999999"
1260 DATA "0123456789:;012301230123456789:;"
1270 DATA "VVVVVVVVVVVVBBBBCCCCSSSSSSSSSSSS"
1280 DATA "00000000000011223344556677777777777"
1290 DATA "0123456789:;0123456789:;0123456789:;"
1300 DATA "VVVVVVVVVVVVIHIHIHIHIHIHSSSSSSSSSSSS"
1310 DATA "4321000000001234567888888888765"
1320 DATA "000123456789:;;;;;:98765432100"
1330 DATA "RDECJKSSSSIHBFGPEDCKJVVVVHIBGF"
1340 DATA "00000000000012345677654321"
1350 DATA "0123456789:;00000123455555"
1360 DATA "VVVVVVVVVVVVRRRFGBIKCDEPPP"
1370 DATA "43210000000012345678888888876578"
1380 DATA "000123456789:;;;;;:98765432100:;"
1390 DATA "RDECJKSSSSIHBFGPEDCKJVVVVHIBGFAB"
1400 DATA "000000000000123456776543214455 66"
1410 DATA "0123456789:;0000012345555 56789:;"
1420 DATA "VVVVVVVVVVVVRRRFGBIKCDEPPPIHIHIH"
1430 DATA "87654321001234567887654321 0"
1440 DATA "2100000123455566789:;;;;;:9"
1450 DATA "IBGFRDECJHBFGPFGBIKCDEPGFBH"
1460 DATA "4444444444440123456789"
1470 DATA "0123456789:;0000000000"
1480 DATA "VVVVVVVVVVVVRRRRLRRRRR"
1490 DATA "00000000001234567888888888 8"
1500 DATA "0123456789:;;;;;:9876543210"
1510 DATA "SSSSSSSSIHBFGPEDCKJVVVVVVVV"
1520 DATA "0011223344 55;;::99887766"
1530 DATA "0123456789:;0123456789:;"
1540 DATA "IHIHIHIHIHIHJKJKJKJKJKJK"
1550 DATA "0001112223337776665554448889 99:;
     ;:;???>>>===<<<"
1560 DATA "0123456789:;0123456789:;0123456789:
     ;0123456789:;"
1570 DATA "IHIHIHIHIHIHJKJKJKJKJKJKIHIHIHIHIHIH
     JKJKJKJKJKJK"
1580 DATA "0011223344555544332211 00"
1590 DATA "0123456789:;0123456789:;"
1600 DATA "IHIHIHIHIHIHJKJKJKJKJKJK"
1610 DATA "0123498765444444"
1620 DATA "01234 0123456789:;"
1630 DATA "BBBBBCCCCCVVVVVVV"
1640 DATA "0123456789:;;:98765432100123456789:;"
1650 DATA "000000000000123456789:;<============="
1660 DATA "PPPPPPPPPPPPCCCCCCCCCCCCRRRRRRRRRRRR"
1670 STOP
1680 RESTORE 1740
1690 FOR C=128 TO 156
1700 READ CD$
1710 CALL CHAR(C,CD$)
1720 NEXT C
1730 RETURN
1740 DATA "8142241818244281"
1750 DATA "8040201008040201"
1760 DATA "0102040810204080"
1770 DATA "030C30C"
1780 DATA "00000000030C30C"
1790 DATA "C0300C03"
1800 DATA "00000000C0300C03"
1810 DATA "0808040402020101"
1820 DATA "80804040202010 1"
1830 DATA "0101020204040808"
1840 DATA "1010202040408080"
1850 DATA "FF01010101010101"
1860 DATA "01010101010101FF"
1870 DATA "80808080808080FF"
1880 DATA "FF80808080808080"
1890 DATA "00000000000000FF"
1900 DATA "0000000FF"
1910 DATA "FF"
1920 DATA "8080808080808080"
1930 DATA "1010101010101010 1"
1940 DATA "0808080808080808"
1950 DATA "0101010101010101"
1960 DATA "8080804040402020 2"
1970 DATA "2010101008080804"
1980 DATA "0404020202010101"
1990 DATA "0101010202020404"
2000 DATA "0408080810101020"
2010 DATA "2020404040808080"
2020 DATA "82927C1010282828"
2030 REM  LETTER REPEATER
2040 PRINT :"IN THIS ACTIVITY A LITTLE   MAN WILL
     HOLD UP EACH LETTER OF A WORD YOU INPUT."
2050 PRINT "AFTER HE HAS PICKED UP THE  LETTER,
     PUSH THAT KEY ON THE KEYBOARD."
2060 PRINT "DESIRED WORD MAY BE UP TO 16
     LETTERS LONG."
2070 INPUT "INPUT DESIRED WORD:  ":ANS$
2080 CALL CLEAR
2090 BL=32
2100 RX0=5
2110 RX1=7
2120 RX3=8
2130 RXL=6
2140 RY0=21
2150 MAN=156
2160 RYL=20
2170 FOR RX=1 TO LEN(ANS$)
2180 CALL HCHAR(RY0,8+RX,
     ASC(SEG$(ANS$,RX,1)))
2190 NEXT RX
2200 FOR DELAY=1 TO 100
2210 NEXT DELAY
2220 FOR RX=1 TO LEN(ANS$)
2230 CALL GCHAR(RYL,RXL+RX,RC)
2240 CALL HCHAR(6,5,32,448)
2250 CALL VCHAR(RYL,RXL+RX,BL,2)
```

```
2260 CALL HCHAR(RY0,
     RX0+RX,RC)
2270 CALL HCHAR(RY0,
     RX1+RX,MAN)
2280 CALL GCHAR(RY0,
     RX3+RX,RC)
2290 CALL HCHAR(RY0,
     RX3+RX,BL)
2300 CALL HCHAR(RYL,
     RX1+RX,RC)
2310 IF RC=32
     THEN 2400
2320 CALL KEY(0,KEE,
     STATUS)
2330 IF KEE=RC
     THEN 2340
     ELSE 2320
2340 IF KEE<58
     THEN 2360
2350 KEE=KEE-7
2360 S1=KEE-48
2370 GOSUB 370
2380 FOR DELAY=1
     TO 200
2390 NEXT DELAY
2400 NEXT RX
2410 GOTO 2070
2420 END
```

Dynamic . . . from p. 71

The problem of rounding up to the next higher grid line (so that 99.9 will show up as 100 as in the earlier example) is taken care of in lines 73 and 74 where a one-row-on character is defined and put on the very top of the bar if, and only if, TOPPATTERN = 9

A graph with only one bar is not very useful. We can generate additional bars with a loop. The routine in Listing 4 plots three bars of different colors. See line 59. (My 13-inch monitor displays a lot of spillover with most colors—especially a lot with red. There is less spillover with light or medium green or blue, and with white and yellow.) As the loop runs, it will shift to succeeding color sets with the expression 89+BAR*8 as can be deduced by considering the statement

```
CALL CHAR(89+BAR*8,TOPPATT
ERN$).
```

When BAR=1, this statement defines character 97; when BAR=2, character 105; and when BAR=3, character 113. The first character is in color set 9, the second in color set 10, and the third in color set 11, allowing for three bars of different colors.

The position of the bars is shifted by the expression 11+BAR*5, where column 11+5=16 is the position of the left edge of the first bar, and the left edges of all bars are 5 columns apart. These bars are three columns wide. Figure 4 shows this graph as photographed on the 13-inch monitor.

This program and earlier ones here might be a little longer than if they were written in the standard way. However, they will not get much longer if the graphics are made more elaborate. For example, the bar graph program does not get much longer if more bars are added.

The bar graph in Figure 5 was made using these techniques. I present it here just to show off the kind of goodlooking graphics that can be made with the TI-99/4A and TI BASIC. This program—with its outlining and the fact that it reads and writes data for eight variables from files, and calculates items such as percentages—is more involved than the listing given here.

This brings up a new problem that has been created: In many of my programs I am running out of characters. I did not notice this limitation when I was typing in so many CALL CHAR, CALL HCHAR, and CALL VCHAR statements. Actually when you think about it, there are not very many characters available. If you start at the left of the screen and put a different character in each space you will run out of characters in the fifth line. This use includes punctuation, numbers, the alphabet, and the eight user-definable sets.

In other words, it only takes about 17% of the screen to display all available characters. Mathematically, we are not about to run out of characters since there are 256 different ways just to put together one row of a character. And the number of characters that can be on the screen in this graphic mode is 24 rows of 32 columns = 768 spaces.

Since my interest is primarily graphics, available user-definable characters are more important to me than memory. Memory problems can often be avoided. To put a unique character on every space on the screen would require 48 character sets—several times more than any home computer presently has. I do not know if this is unreasonable. Two years ago the idea of 48K memory sounded unreasonable. Perhaps some computer architect will devise a method of going to higher resolution with nested character sets . . . [For a discussion of the high-resolution bit-mapped graphics supported by the TI-99/4A, see *3-D Animation with the TMS9918A Video Chip* in this issue—Ed.]

Finally, note that for some applications it can be useful to define random graphic characters. This process, however, really eats up character sets. In Listing 5, random characters are defined that also have a certain amount of shape. Line 15 of this code generates random numbers from 1 to 16, and lines 39 to 53 convert them to hexadecimal notation 0,1,2,3,4,5,6,7,8,9,A,B,C,D,F.

Continued on p. 84

Square 1 . . . from p. 72

frequency data tones that mean "off" but only hear part of the higher frequency tones that means "on." In order to help the TI-99/4A hear the high frequency tones properly, the tone control knob on the recorder should be set at or near the maximum level. Even if this is done, some tape recorders cannot handle the high frequency. If your recorder doesn't have a tone control, there's a good chance it was probably meant to handle only the frequencies of human speech and won't be mechanically able to handle the high frequency tone at all.

Since it is possible that your recorder cannot reproduce the high frequency tones properly, your computer has to be sure that it has read *all* the data. How can it make sure that nothing was lost? Your computer counts the number of "ons" that it heard. After every so many bytes, it expects to read a number on the tape. This number tells the computer how many "ons" it should have read. If the two numbers don't match, a "parity error" has occurred and the computer will tell you that you have a problem.

Now suppose that the volume is set high enough to reproduce the high level tones, but is up *too high*? Well, too much volume causes distortion in a tape recorder. This distortion will mean that some of the tones will not be heard accurately by the computer at all. It's just like someone screamed in your ear. You know something was said, but you don't know what it was.

Let's recap what we just learned. In order for your computer and the tape recorder to communicate properly, three things must happen:

First--the tape recorder has to be able to handle both the high and low frequency tones and it must be capable of reproducing them within a small range of the frequency at which it was recorded. In order to record and playback the higher frequency tone, it is usually necessary to have the tone control of the tape recorder set at, or near, maximum.

Second—the tape recorder must be capable of small volume adjustments so that you can reach a condition where the volume is loud enough to be heard by the computer and yet low enough so that there is no distortion.

Third—the tape recorder must be able to record and playback the very quick shifts in frequency (from "off" to "on") accurately so that no tone is lost when being played back into the computer.

A Remote Possibility

There is one additional problem that may crop up even with tape recorders that satisfy the above criteria: Almost all cassette recorders have a remote control jack which allows you to stop the recorder by pressing a button or switch located on the microphone. Unfortunately, since this jack is meant to work with the manufacturer's own microphone, there is no guarantee that the jack is hooked up the same way in each tape recorder. In fact, there is a 50-50 chance that the tape recorder model you buy or already own will not be compatible with the system your TI-99/4A is expecting. This means that the drive motor of your recorder might not be capable of being turned on and off *automatically* by the computer when the plug on the TI cable is inserted into the recorder's control jack. Luckily, if this is true for your recorder, 99'er-ware sells an inexpensive adapter (called "TI-SETTE") which is used between your recorder and the TI cable. If you don't want to spend the money for this adapter, you can get by without, by *manually* starting and stopping the tape [except if you intend to use cassette files, in which case the automatic operation is necessary—Ed.]

The conclusion you can draw from all this is that your TI-99/4A requires a tape recorder with specific attributes in order to consistantly guarantee good results. TI provides its owners with a list of 2-6 tape recorder models that work well with the 99/4A computer. If you do not already own a recorder, I strongly suggest that you buy one of the recommended models. If you do have a recorder, you can try it out before incurring the expense of purchasing a new one.

Plugging In!

Now that we have discussed why some recorders won't work at all or won't work with the remote control jack plugged in, let's get down to business. Shut off your machine and plug the wide cable (with 9 holes in it) into the back of your computer. The other end of the cable has two cords. One cord has three plugs attached (labeled plug #1), and the other (plug #2) has only two. The tape recorder that you connect to plug #1 will be called "CS1" by the computer. If you are lucky enough to have a second usable tape recorder, you can hook up that one to plug #2. It will be called "CS2" by the computer. Just follow the installation instructions printed on the card that came with the TI cassette cable. If your tape recorder does not have a remote control jack, just ignore the instructions to insert the black plug. Note that CS2 does not have a playback plug. You can only record on CS2.

Plug the tape recorder into an electrical outlet and you are now ready to check out your system. [A battery-operated tape recorder is usually too unreliable for recording and playing back data for your computer because of the fluctuations in speed and amplifier gain—Ed.] Load a high quality (remember we have to record those high tones accurately!) C-10, C-15, or C-30 blank tape into the tape recorder. The number part of the tape code gives the number of minutes of recording time available on both sides of the tape. A C-10 tape has 5 minutes of recording time on each side. You can use a tape as long as a C-60, but never anything longer. This is because longer tapes are thinner, stretch more, and may not maintain proper speed in the recorder. For this first test, make sure the tape is completely blank. Turn on your computer and get into TI BASIC. Key in the following 4 line program:

```
100 PRINT "HELLO"
110 I=30
120  PRINT "MY VALUE IS";I
130 END
```

Turn up the volume on your TV (or monitor) by a few notches so that you can hear a slight hum. Set the volume control on your tape recorder mid-way between the lowest and highest settings. Set the tone control (if there is one) up to maximum. Now type in SAVE CS1 and press the ENTER button. Follow the instructions that the computer gives you to rewind the tape and begin recording. When you press "record" on your tape unit and then press the ENTER button on the computer, the tape should start moving.

If the tape doesn't start moving, you have a non-compatible remote control jack. If this is the case, wait for the computer to get out of recording mode and print the "VERIFY (Y/N)" message. When it does, type in an "N". Now remove the plug from the remote control jack and begin the recording process all over again (by typing SAVE CS1 and pressing the ENTER button). When you are told to record, you should now see the tape moving.

Getting Adjusted

After a short pause, you will actually hear your program being recorded onto the tape. The recording consists of an initial long phrase of a single tone, followed by bursts of sound with a very short pause between bursts. This initial tone is used to tell the computer on playback that data is coming. This tone is recorded before each program and each block of data (which we will talk about later). When the recording is over, you will get the verify message (see above). Type in a "Y" (you don't have to press the ENTER button). Follow the instructions about rewinding the tape. When you play back the tape, listen to the sounds that it is making. Note that the volume is much louder than when you recorded. If that initial tone does not sound pure (it seems to warble with the tone going higher and lower), you are probably using a recorder that won't work well consistently. If the tone does seem pure, you're half-way home!

When the tape goes silent, the program has finished loading. You should get a message that says either "DATA OK" or "ERROR IN DATA". If no message prints, then the volume setting was too low and your computer is still waiting for the first recognizable byte of data. It will eventually get tired of waiting and give you a "NO DATA FOUND" error. Just wait for this message to appear, or shut off your computer and start all over again.

If you got the "DATA OK" message, you are home free! Relax and go on to the next paragraph. If you were unlucky enough to get a "NO DATA FOUND" error, turn up the volume one notch. If you got the "ERROR IN DATA" message, you probably had the volume too high. Lower the volume one notch. Write down the latest notch on a piece of paper. In either case, respond to the computer question by entering an R to re-record. The computer will guide you in another recording session. Keep repeating the process until you can't change the volume any further, or the "DATA OK" message appears, or the error message has changed (i.e., from "NO DATA FOUND" to "ERROR IN DATA"). If you can't change the volume any further, your recorder just isn't good enough. Don't aggravate yourself any longer, go out and buy one from the list. If the DATA OK message has appeared, you are in good shape. If the message has changed, back off your last change by half a notch. For example, if moving the control from 6 to 7 made the "ERROR IN DATA" message appear, try the recording process again at 6½. If that doesn't work, try it at ¼ notch intervals. If *that* doesn't work, forget it. Buy a different recorder.

After you get the "DATA OK" message, mark the volume setting in some way. I usually dip a toothpick in white paint (a light nail polish will also work) and dab a line on both the recorder and the control so that I can easily see that the volume setting is correct. You now have a functioning cassette tape system and are ready for bigger and better things!

Better Safe than Sorry

When you entered the SAVE CS1 command, you told the computer to copy the bytes that represented your program inside the computer onto a tape. The entire program is saved each time. Your program is still in the computer, however. If you agree to verify your tape, TI BASIC will read in the tape and compare it in a byte-for-byte

Continued on p. 82

NEW PRODUCTS AND SERVICES

TEXAS INSTRUMENTS SPONSORS COMPUTER CLUBS FOR CHILDREN

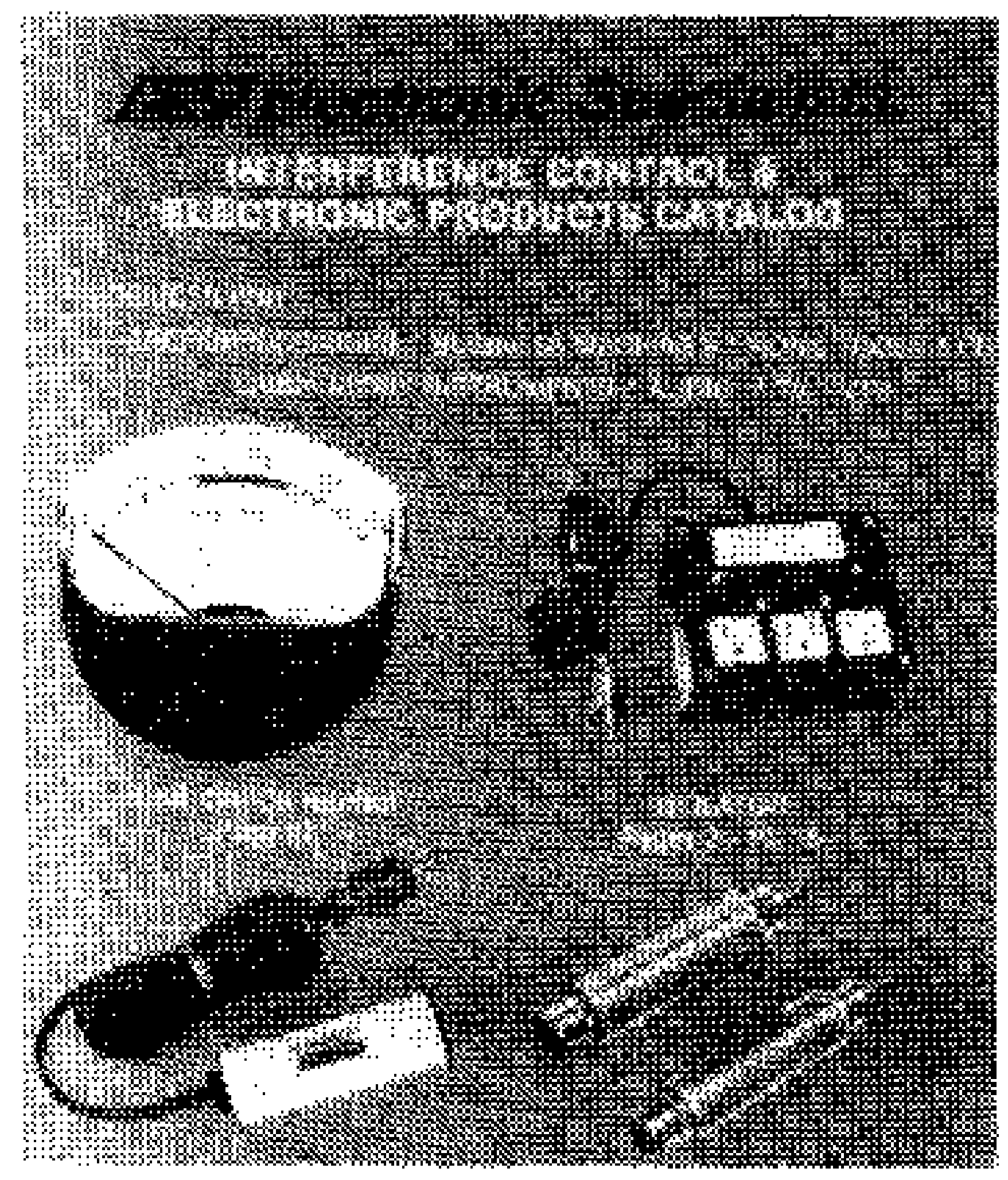
In 24 cities across the country this summer, thousands of 8 to 15-year-olds will participate in Texas Instruments Computer Advantage Clubs. The clubs offer young people extensive hands-on experience with TI home computers. What sets TI's. Computer Advantage Club apart from other summer computer programs is accessibility and affordability. For only $65 per child ($50 for each additional family member), the club offers a Computer Awareness Program covering computer terminology, programming in two computer languages, TI LOGO and TI BASIC, and use of solid state software in music, education, and arcade quality entertainment. The sessions, to be held from mid-June through August, will include two and a half hours of instruction per day for four days in an informal classroom setting.

In addition to the four-day Computer Awareness Program, club membership will provide a quarterly newsletter, a membership card, and the club t-shirt. Parents of members will also be invited to a Parents' Night where they can learn how to help children expand their computer skills.

TI's Education and Communication Center in Dallas, Texas, developed the curriculum and activity book for the program in conjunction with Computer Camps International. The activities were specifically designed for use with the TI-99/4A Home Computer, a system with unique capabilities for use by children, as well as the entire family.

"The rapid advancement of technology has made practical knowledge of computers a virtual necessity," Chuck Digate, Director of The Computer Club Program for TI's Consumer Products Group, said. "Computers affect us from early childhood throughout our adult lives, and TI Computer Advantage Clubs provide opportunities for young people to take advantage of new electronics technology."

For information on the camps, parents outside Texas can call 1-800-858-4069. In Texas, the number is 1-800-692-1318.



INTERFERENCE CONTROL HOT LINE FOR MICROS

Electronic Specialists, Inc announce installation of a new Toll Free Interference Control Hot Line. Call 1-800-225-4879 between 9 AM and 4 PM Eastern Time Monday through Friday to discuss microcomputer interference control problems. Experienced staff will analyze the problem situation and make specific recommendations for control of troublesome processor or peripheral interference.

Analysis of System electrical interference problems is also possible with the help of many typical situations and solutions outlined in FSI's free 40 page Interference Control Product catalog. Electronic Specialists, Inc., 171 South Main Street, P. O. Box 389, Natick, MA 01760, Phone (617) 655-1532.

2700-BAUD CASSETTE LOADER NOW AVAILABLE

A cassette loader for Assembler programs for use with the Extended BASIC Command Module, (the standard basic cassette load is not enabled) has been announced by Data Force Inc. This product will be used by Data Force, to offer its Assembler products to users with Expansion RAM and cassette, and will be available to other Assembler program developers. 2700, 2100 and 1500 baud versions have been developed. Also announced is a new arcade-style game, Kippy's Nightmare. It is written entirely in Assembler Language and is available on both disk and cassette using the newly-released 1500-baud cassette loader. This product runs on both 99/4 and 99/4A with Extended BASIC and Expansion RAM. The price is $34.95 for either version. For additional information contact: Data Force Incorporated, 10 So. 312 Hampshire Lane East, Hinsdale, Illinois 60521. Phone (312) 323-0179

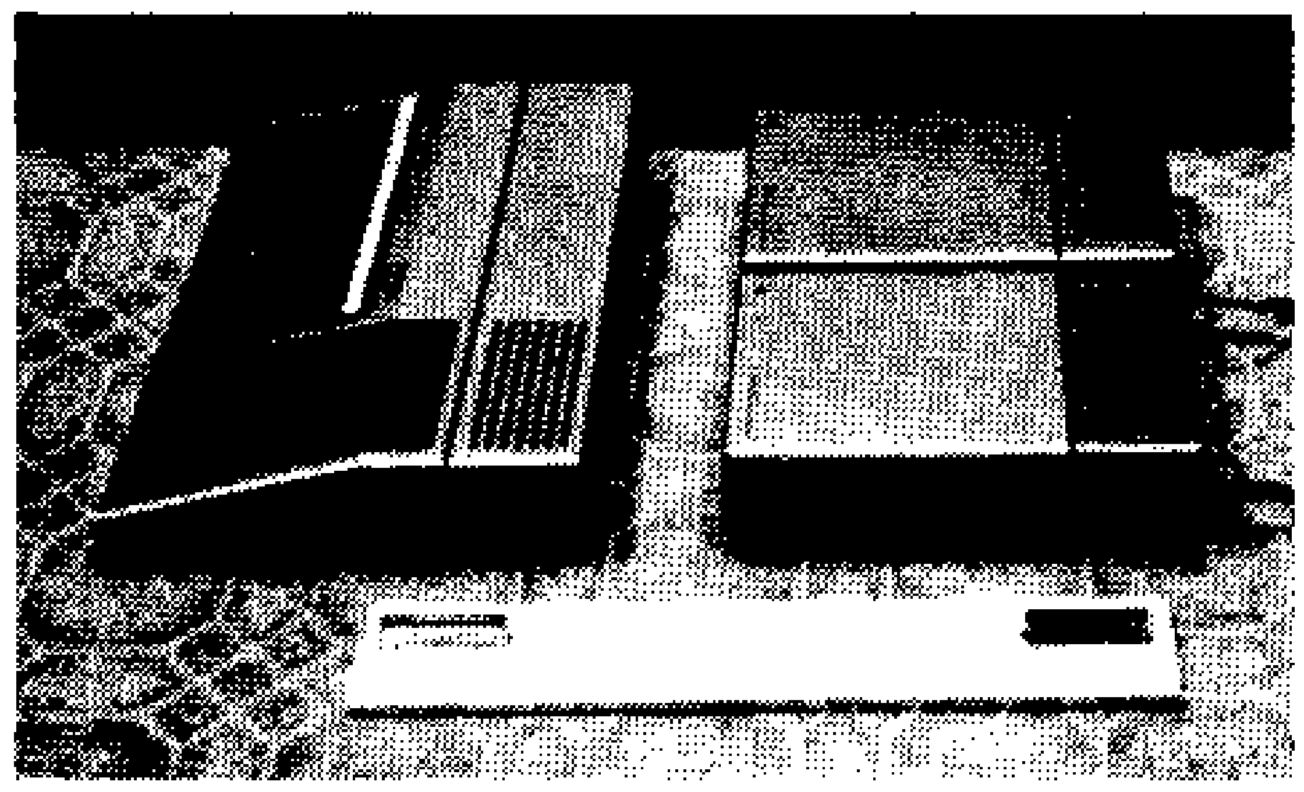
CONTROL DATA AND TI SIGN AGREEMENT ON PLATO COURSEWARE FOR TI HOME COMPUTER

Control Data Corporation and Texas Instruments, Inc. announced an agreement that will make available a comprehensive series of PLATO computer-based education courseware for the TI-99/4A Home Computer. Control Data was a pioneer in developing computer-based educaton with the PLATO system. After 20 years, PLATO courseware includes thousands of hours of integrated instruction materials.

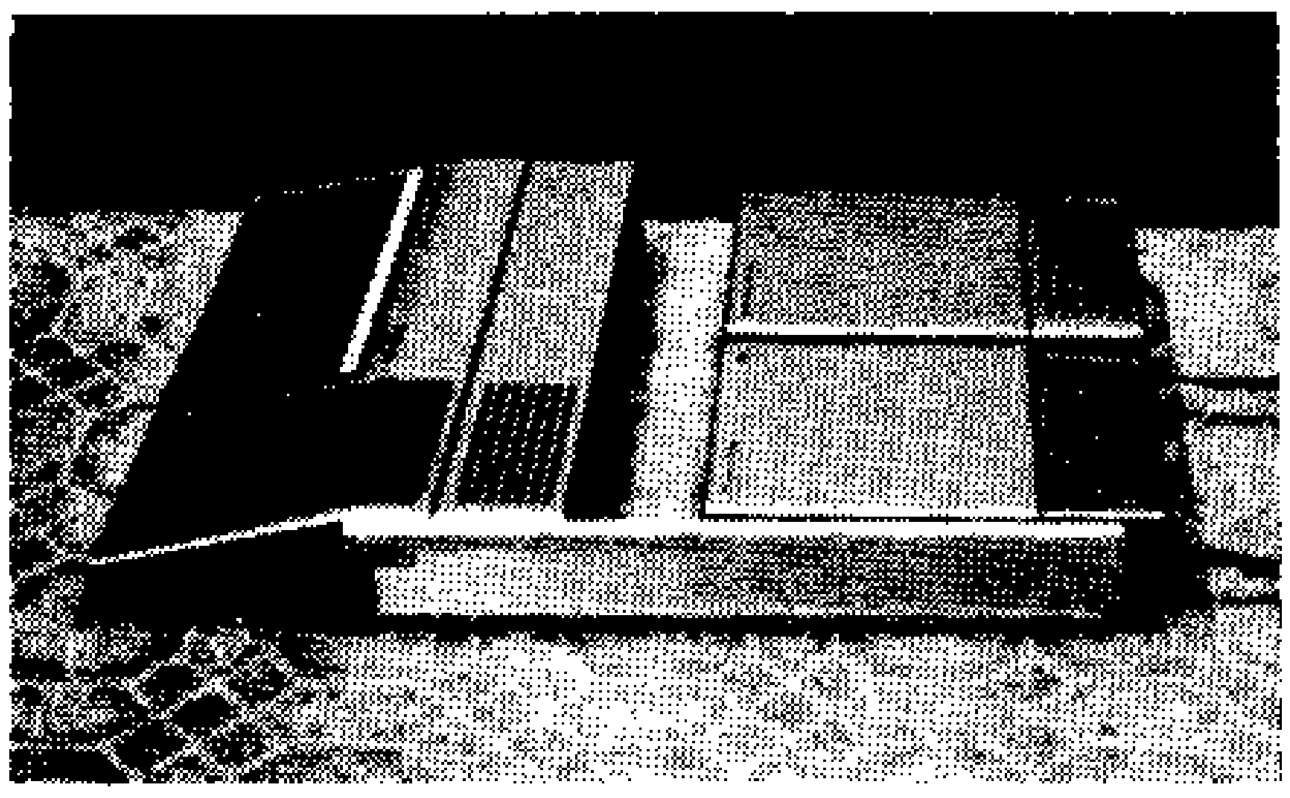
The agreement with Texas Instruments encompasses 430 programs in 108 courseware packages developed for elementary- and secondary- school levels. This library is widely regarded as the most complete and comprehensive available. Using PLATO courseware on the TI-99/4A will make these established educational materials economically practical for many classrooms and homes. Control Data will convert its Basic Skills and High School Skills curriculums to run on the TI 5¼ inch diskettes, and Texas Instruments will manufacture and distribute them.

This PLATO courseware spans grades kindergarten through 12 with a wide range of computer-based instruction in reading, math and language arts, including poetry and literature. It also encompasses physics, chemistry, earth sciences and biology as well as social studies such as geography, economics, behaviorial science, political science, and history. The materials are designed to provide self-paced and individualized instruction for students whose needs range from remedial help to a advanced instruction. The initial packages will be available in the fourth quarter of 1982. For additional information contact: Texas Instruments, P. O. Box 53, Lubbock, Texas 79408. Or call toll-free 1-800-858-4565 (in Texas call 1-800-692-4279).

PERIPHERAL LINE SHORTENED BY NEW BUS EXTENDER



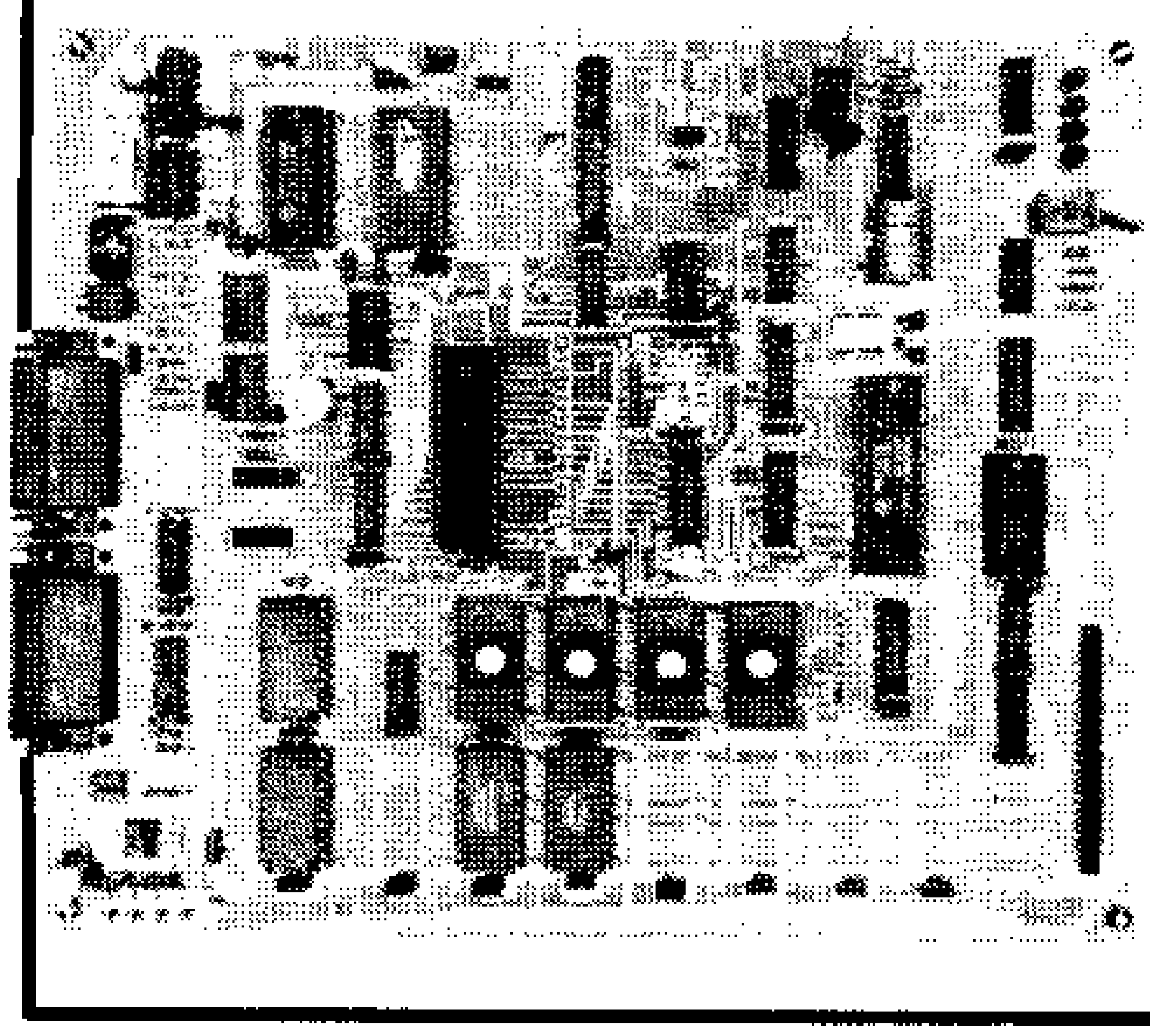
A new Backer Bus (tm) for the TI-99/4 and 99/4A has been announced by Denali Data Design. It allows placement of peripherals behind the console, instead of the continuous straight chain of add-on boxes to the right. A full-system configuration can be contained in an area 25" wide by 24" deep. The price of the Backer Bus is $59.95 in quantity one. For additional information contact: Denali Data Design, 1413 N. McKinley Ave., Oklahoma City, OK 73106. Phone 1-800-654-8499



TI ANNOUNCES VERSATILE TOOL FOR EXPERIMENTING WITH SYNTHESIZED SPEECH

A single-board microcomputer module that provides hands-on experience in working with Linear Predictive Coding (LPC) synthesized speech has been announced by Texas Instruments. The new Speech Education Module features highly flexible hardware capabilities and an interactive Forth monitor that allows extensive user-defined experimentation. This makes the module well suited for use as a speech-applications development aid or as a learning tool for speech-synthesis technology.

The version of Forth contained in the Speech Education Module features a unique set of speech-oriented primitive functions that enables users to interact with the module at high levels (such as concatenating words or speaking a phrase) or at very low levels of function (such as state-by-state control of the module's voice-synthesis processor).



Through a set of experiments provided with the module, users can learn to specify their own unique Forth commands and command sequences. Each of the high-level software operations is available for use in defining new operations, eliminating the necessity for the user to get involved with the details of assembly language. This allows quick adaptation of speech-output capability to almost any application or experiment.

Measuring less than 9.5" x 11", the Speech Education Module includes a TMS7000 8-bit microcomputer chip, TMS5220A voice-synthesis processor, 4K bytes of RAM (random-access memory), 12K bytes of EPROM (erasable programmable ROM), and a VM61002 vocabulary ROM (read-only memory) containing 206 pre-programmed words and phrases. It also has two RS-232C serial ports with independent baud-rate controls, an 8-bit switch register for user applications, four software-controlled LED (light-emitting diode) status indicators, an 8-bit analog-to-digital converter, and an audio amplifier with user-selectable cutoff frequencies for direct speaker drive.

On-board memory can be expanded to include up to 14K bytes of RAM or EPROM using any desired mix of TMS2516 EPROMs and TMS4016 static RAMs. A second TMS 6100 series voice-synthesis memory ROM containing TMS5220-compatible speech data can be included on board or substituted in place of the pre-programmed VM61002 vocabulary ROM. In addition, the TMS7000's control of the system bus can be disabled by jumpers to permit external control of the module.

Minimal hardware is needed to set up the Speech Education Module for operation. For a basic configuration, the user needs a standard RS-232 data terminal, a 4- or 8-ohm loudspeaker, and a three-voltage power supply (+5, +12, and -12 volts) such as the TM990/519. External hardware can be expanded via the module's second RS-232 port to include an oscilloscope, printer, or serial link to a host computer.

The Speech Education Module is available for $600. For additional information contact: Texas Instruments Incorporated, Central Literature Response Center (SC-366), P. O. Box 202129, Dallas, Texas 75220

CATALOG ON TI SEMICONDUCTOR PRODUCTS NOW AVAILABLE

A 128-page short-form catalog entitled "1982 Semiconductor Master Selection Guide" (SCG 682) is available at no charge from Texas Instruments. The catalog gives key specifications for TI's current line of semiconductor products in the areas of microcomputers, memories, logic arrays, voice synthesis, digital logic, linear circuits, telecommunications, optoelectronics, discrete devices, and military products. Also included is an appendix with information on integrated-circuit part-number coding and package-outline drawings. For a copy of the catalog, circuit designers and systems integrators may contact their local TI distributor or write: Texas Instruments Inc., Central Literature Response Center, P. O. Box 202129 (Attn: SCG682), Dallas, Texas 75220.

Who . . . from p. 69

```
TO CHECK
TELL 0
TEST PADDLETOUCH
IFT SETHEADING 270
END
```

And then BOUNCE2 is changed a little more so that if the sprite reaches the top of the screen instead of "wrapping" to the bottom, it bounces back down; and if it reaches the bottom of the screen, it bounces back up; and when it hits the left-hand boundary, it bounces at a 70 degree heading instead of the current 90 degree heading:

```
TO BOUNCE2
TELL 0
TEST XCOR > 85
IFT CHECK
TEST XCOR < -85
IFT SETHEADING 70
TEST YCOR > 90
IFT SETHEADING 135
TEST YCOR < -85
IFT SETHEADING 45
END
```

This leaves just the problem of keeping score. Besides keeping score, a nice feature would be to have different noises made when the player scores and when the computer scores. When the ball bounces off the paddle, then the player's score should increase and be printed; when the ball misses the paddle, then the computer's score should be increased. Notice that the CHECK program is only invoked if the ball is beyond XCOR 85. Therefore, part of the scoring and noises can be controlled after line 3 of BOUNCE2 in the CHECK program: CHECK is rewritten:

```
TO CHECK
TELL 0
TEST PADDLETOUCH
IFT CALL :PLS + 1 "PLS
            ; this line increases the player's
            score
IFT NOISE
            ; this line causes NOISE to run for
            the player's point
IFT SETHEADING 270
IFF CALL :CPS + 1 "CPS
            ; this line increases the computer's
            score
IFF BEEP WAIT 10 NOBEEP
            ; this line causes a short beep
            for the computer's point
TYPE [ YOUR SCORE IS ] PC 32
TYPE :PLS PC 32
TYPE [ THE COMPUTER'S SCORE
  IS ] PC 32
PRINT :CPS
WAIT 90 CS
            ; the wait is added to prevent
            extra scoring on each serve
END
```

```
TO NOISE
REPEAT 5 [ BEEP WAIT 3
  NOBEEP WAIT 3 ]
END
```

It is necessary to set up an initial value for both the computer's score and the player's score as was done with :Y. Since this is done just once, it belongs in SETUP [The initial score is 0 to 0--as in the proverbial "soothsayer's" prediction or score before it begins . . .] So SETUP is revised:

```
TO SETUP
TELL 0
CARRY :BALL
SETCOLOR :BLUE
HOME
SETHEADING 90
SETSPEED 15
TELL [ 1 2 ]
CARRY :BOX
SETHEADING 0
SETCOLOR :BLACK
SXY 100 0
TELL 2
SY 16
CALL 16 "Y
CALL 0 "PLS
CALL 0 "CPS
END
```

This game, like most LOGO projects, is open-ended. It could be altered so that a winner is named at a score of 21, it could be revised for two players, changed to use joysticks, or changed so that the ball has topspin. With each addition, it is necessary to make sure that the conditions to be established once are done so, that procedures to be repeated are placed inside a recursive program, and that there are no Recursion Interface Bugs.

Listings on p. 86

Want to Get Published?

99'er Magazine is looking for articles in all areas of interest that concern the Texas Instruments personal computers. 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.

Please send your double-spaced typed manuscripts, plus disks or cassettes (recorded on both sides) if the article includes program material, to:

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

Square 1 . . . from p. 79

manner with the program still residing in memory. Unless the two match perfectly, your 99/4A will issue a warning that you have a bad tape. **ALWAYS VERIFY ANY SAVEs BEFORE ENDING A PROGRAMMING SESSION!**

The tape version of the program is saved in a "machine image" format that is meaningful only to TI BASIC. You cannot, however, write a TI BASIC program that will read this tape. The only way to get your program back into the 99/4A is via the OLD CS1 command. This will load the program back into the machine. Anything that may have been in the computer before the OLD CS1 will be lost. By the way, you can SAVE CS2 (if you have a recorder hooked up to cable #2) and then read in the tape by entering OLD CS1. Of course, you have to move the tape over to the recorder attached to cable #1 first!

The instructions built into the TI-99/4A whenever you enter the SAVE CS1 or OLD CS1 command assume that you have only one program per side of tape. In reality, you can save more than one program per side. A long program will require about 3-4 minutes of recording time. This means that it is possible to save about 4-5 programs on each side of a C-30 tape. If your recorder has a tape counter, just keep track of where the next free space on the tape is located. Then, when the computer tells you to rewind the tape, just fast-forward to that next free spot on the tape instead. Just make sure to keep a log of what programs are recorded on a tape and where they are located. [If you don't want to be bothered by this, and want maximum reliability, it is better to use C-10 cassettes and record only *one* program per side—Ed.]

A cassette tape recorder will usually have the ability to record a new program directly over an old one. It is good to get into the habit of completely erasing a tape, however, when you no longer need it. This ensures the best possible recording the next time you use the tape.

Filing Data

The cassette recorder also makes a handy data storage device for use in your computer programs. Suppose that you have written a program to keep track of the bowling scores and figure out the handicap of each member of your bowling league. You don't want to re-enter this information each time you run your program. What you need is a way of saving the data when you are through with it so that it can be read in the next time around. Some people do this by coding in the information in DATA statements each time before SAVEing the program. A better way of doing this is to write out a small "file" of data onto a tape. Your program can then read in this data the next time it runs. TI BASIC has an easy way of doing this by using the INPUT # and PRINT # statements.

Before you can read or create a file, you must tell the computer a little about your file. This is done by the OPEN statement. Your reference manual does a pretty good job of explaining this statement, so I'll just go over the parts specifically dealing with cassette tape files.

Unlike the SAVE command which writes out your entire program as a large "chunk" of data, BASIC data files can only handle small chunks of data, called "records," at a time. Each file can contain 1 or more records. All cassette records in a file must be of the same size. They can all be 64 bytes (characters) long, 128 bytes long, or they can all be 192 bytes long. You can specify other lengths as part of the OPEN statement, but TI BASIC will boost the number up to either 64, 128 or 192. If a record you want to write is shorter than the length that you specify, TI BASIC will add enough blanks at the end of the record to make it the right length.

Each record can contain as much data as you can fit in that size record. When you have a statement that says PRINT # and ends with a semi-colon, BASIC will add that data to the record, but will *not* write anything out to the tape. When BASIC sees a statement with PRINT # that *doesn't* end with a semi-colon, it will write out everything in a record (including this last piece of data) to the tape. When the record is written to tape, it is preceeded by the same steady high-pitch tone that starts off a SAVE. That means that BASIC uses a lot of tape to write a single record. In fact, if you use records that are only 64 bytes long, it is possible that more room is spent on the tape for the start tone then is used to record the data! Remember that more room on the tape means slower reading by the computer. That's why I usually use 192 byte records and try to fit as much data as possible into each record. Doing this will cut down on the number of records written to tape, and make the program run faster.

Since TI BASIC only writes to tape when you tell it to, the computer must have total control of the cassette recorder so that it can start and stop the recorder as needed. This means that the black remote-control plug must be inserted (and functional!). If your remote jack is not compatible with the TI-99/4A, you will *not* be able to use the recorder for saving and reading data under program control. [This is where the TI-SETTE Adaptor, listed at the end of this article, can help—Ed.]

You can store in two different formats. DISPLAY format means the data is saved just the way it would look in a DATA statement. INTERNAL format saves the data in the same way that the computer stores the information internally. Numbers require 8 characters (bytes). Strings (i.e., names) require 1 byte (for the length) plus the data itself. I usually save my data in INTERNAL format so that I know the length needed for numbers no matter how big or small they are.

The BASICs of Record Keeping

Let's write a part of a program that will save each bowler's name, his pin average and his handicap. Pretend that we have 60 bowlers in our league. If we restrict each bowler's name to a maximum of 47 characters, we will need a total of 64 bytes per bowler (47 bytes + 1 = 48 for the name + 8 for the average + 8 for the handicap = 64). We can therefore fit the data for 3 bowlers into one 192 byte record. (see figure 1) If you have filled up a record by the time the program hits the CLOSE statement, TI BASIC will fill the record with blanks and write it out. You do not have to worry about writing out a last record that is partially full. Just remember to always code in a CLOSE statement. To read the data file into your program, you need code that almost duplicates the write code. (see figure 2)

Continued on p. 84

```
090  REM ROOM FOR 60 BOWLERS NAMES, AVERAGES, HANDICAPS
100  DIMENSION B_NAME(60),B_AVG(60),B_HANDI(60)

995  REM OPEN THE FILE FOR OUTPUT
1000 OPEN #1:"CS1",OUTPUT,INTERNAL,SEQUENTIAL,FIXED 192
1010 X=1
1020 FOR I=1 TO 60
1025 REM SEE IF RECORD IS FULL
1030 IF X=3 THEN 1100
1040 X=X+1
1050 REM ADD TO RECORD– BUT DON'T WRITE IT  OUT
1060 PRINT #1:B_NAME(I);B_AVG(I);B_HANDI(I);
1070 GOTO 1120
1090 REM ADD TO RECORD AND WRITE IT OUT!
1100 PRINT #1:B_NAME(I);B_AVG(I);B_HANDI(I)
1110 X=1
1120 NEXT I
1130 CLOSE #1
```

Figure 1

```
195  REM OPEN THE FILE FOR INPUT
200  OPEN #1:"CS1",INPUT,INTERNAL,SEQUENTIAL,FIXED 192
210  X=1
220  FOR I=1 TO 60
230  REM SEE IF RECORD IS FULL
240  IF X=3 THEN 300
250  X=X+1
260  REM READ RECORD– BUT DON'T READ TAPE
270  INPUT #1:B_NAME(I);B_AVG(I);B_HANDI(I);
280  GOTO 320
290  REM READ RECORD AND GET NEXT TAPE
300  INPUT #1:B_NAME(I);B_AVG(I);B_HANDI(I)
310  X=1
320  NEXT I
330  CLOSE #1
```

Figure 2

Crayon . . . from p. 61

defined in the operand field. The label is assigned the address of the first byte at the time the object program is loaded. All of these buffer areas are contiguous. For example, look at the instructions immediately after the label MARKER. The pattern codes for two double-size sprites, the cursor and arrow, are loaded into the Sprite Descriptor Table in VDP RAM. Since the pattern data for ARROW is contiguous with that of CURSOR in both CPU and VDP RAM, all 64 bytes can be loaded in one shot.

You should have little trouble figuring out the rest of the program by reading the comments provided and referring to the manual. But don't stop after you understand how it works —try to make some changes. To start with, try changing the shape and colors of the sprite cursor, the arrangement of the color palette on the screen, etc. Then try to make the program more efficient in speed and utilization of memory.

Be prepared to run into problems; it's through encountering and solving them that you'll learn most rapidly. When I decided to stop reading and start trying to write a program, I had visions of seeing a curl of white smoke rise from the computer's cooling vents, but that didn't happen to me, and probably won't happen to you either. So don't be afraid to experiment.

MAGIC CRAYON
99'er Version 1.6.1 AL

```
x        DEF   MARKER
       REF   VSBW,VMBW,VMBR,VSBR
       REF   VWTR,KSCAN,DSRLNK
*
* DEFINITION OF LABELS
*
SCREEN BSS   >300
PALET  BSS   >600
PATRN  BSS   >600
ROW    BSS   1
COL    BSS   1
CURSOR DATA  >8040,>2010,>0804,>0000
       DATA  >0000,>0408,>1020,>4080
       DATA  >0102,>0408,>1020,>0000
       DATA  >0000,>2010,>0804,>0201
ARROW  DATA  >0102,>0408,>0000,>0000
       DATA  >0000,>0000,>0000,>0000
       DATA  >0080,>4020,>0000,>0000
       DATA  >0000,>0000,>0000,>0000
ATTRIB DATA  >5878,>800F,>D000
ARRATT DATA  >6578,>8401
PDATA  DATA  >0600,>1000,>0000,>0600
       DATA  >000B
       TEXT  'DSK1.SCREEN'
ZERO   DATA  >0000
D32    DATA  >0020
D8     DATA  >0008
GRAY   DATA  >EEEE
MAX    DATA  >05FF
COLMAX DATA  >0100
LOAD   BYTE  >05
BLACK  BYTE  >11
ONE    BYTE  >01
TWO    BYTE  >02
FCOLOR BYTE  >10
BCOLOR BYTE  >0E
H18    BYTE  >12
H14    BYTE  >0E
H11    BYTE  >0B
H07    BYTE  >07
H06    BYTE  >06
H05    BYTE  >05
H02    BYTE  >02
NOKEY  BYTE  >FF
PAB    EQU   >0F80
USRWS  EQU   >20BA
PNTR   EQU   >8356
UNIT   EQU   >8374
FIRE   EQU   >8375
JOYSTY EQU   >8376
JOYSTX EQU   >8377
SPRITE EQU   >837A
STATUS EQU   >837C
GPLWS  EQU   >83E0
*
* DEFINE SPRITE PATTERNS FOR CHRS 128 AND 132
*
MARKER LWPI USRWS              LOAD WORKSPACE POINTER / START
       LI   R0,>400            VDP ADDRESS CH 128 SPRITE DESCRIPTOR TABLE
       LI   R1,CURSOR          CPU ADDRESS OF CHAR PATTERN
       LI   R2,64              64 BYTES TO MOVE (2 PATERNS)
       BLWP @VMBW              LOAD DATA TO VDP RAM
*
* SET FOREGROUND AND BACKGROUND TO GRAY
*
       LI   R0,>01F0           PLACE IN TEXT MODE
       BLWP @VWTR              WRITE TO VDP R1
       LI   R0,>07EE           SET FORE AND BACKGROUND TO GRAY
       BLWP @VWTR              WRITE TO VDP R7
*
* INITIALIZE SCREEN IMAGE TABLE FOR MULTICOLOR MODE
*
       LI   R0,SCREEN          INITIALIZE POINTER
       LI   R1,6               INITIALIZE GROUP COUNTER
       CLR  R2                 INITIALIZE VALUE
LOOP0  LI   R3,4               INITIALIZE REPETITIONS COUNTER
LOOP1  LI   R4,>20             INITIALIZE VALUE COUNTER
       MOVB R2,R5              START REPETITION
LOOP2  MOVB R5,*R0+            STORE VALUE IN ARRAY SCREEN
       AI   R5,>0100           CHANGE TO NEXT VALUE
       DEC  R4                 COUNT DOWN FOR NEXT VALUE
       JNE  LOOP2              DO NEXT VALUE
       DEC  R3                 DEC REPETITION COUNTER
       JNE  LOOP1              DO NEXT REPETITION
       AI   R2,>2000           NEXT STARTING VALUE
       DEC  R1                 DEC GROUP COUNTER
       JNE  LOOP0              DO NEXT GROUP
       LI   R0,>00             VDP ADDRESS FOR SCREEN IMAGE
       LI   R1,SCREEN          CPU ADDRESS OF DATA BUFFER
       LI   R2,>300            768 BYTES TO WRITE
       BLWP @VMBW              INITIALIZE VDP SCREEN IMAGE
*
* INITIALIZE COLOR PALETTE SCREEN
*
       LI   R0,>100            INITIALIZE WORD COUNTER
       LI   R1,PALET           INITIALIZE POINTER FOR
                               PALET ARRAY
LOOP3  MOV  @GRAY,*R1+         STORE GRAY COLOR >EEEE
       DEC  R0                 DEC WORD COUNTER
       JNE  LOOP3              WRITE NEXT WORD
       CLR  R0                 INITIALIZE CULOR VALUE
       LI   R3,16              INITIALIZE COLOR COUNTER
LOOP4  LI   R4,2               INITIALIZE COLUMN COUNTER
LOOP5  MOVB @GRAY,*R1+         STORE GRAY BYTE
       MOVB @GRAY,*R1+         STORE ANOTHER GRAY BYTE
       MOVB @BLACK,*R1+        STORE BLACK BYTE
       LI   R5,4               LOAD COUNTER FOR COLOR BYTES
LOOP6  MOVB R0,*R1+            STORE A COLOR BYTE
       DEC  R5                 DEC COLOR BYTE COUNTER
       JNE  LOOP6              STORE ANOTHER COLOR BYTE
       MOVB @BLACK,*R1+        STORE A BLACK BYTE
       DEC  R4                 DEC COLUMN COUNTER
       JNE  LOOP5              DO SECOND COLUMN
       SWPB R0                 SHIFT TO LEAST SIG BYTE
       AI   R0,>11             ADD 1 FOR NEXT COLOR NUMBER
       SWPB R0                 SHIFT BACK TO MOST SIG BYTE
       DEC  R3                 COUNT DOWN COLOR COUNTER
       JNE  LOOP4              DO NEXT TWO COLUMNS
       LI   R0,>300            SET BYTE COUNTER FOR
                               REMAINING SCREEN
```

Continued on p. 85

Note that statement 300 reads in the last piece of the record and tell TI BASIC to read in a new tape record *the next time* it sees a READ #1 statement.

When your program hits the OPEN statements, it will issue commands about rewinding the tape and pressing ENTER. Just before it reads the first record, the screen will scroll up one line to indicate that it has begun processing the tape.

I have often been asked why TI provides the CS2 plug. I have to admit that most manufacturers do not provide dual cassette support. It is useful if you must process more data in your program than the computer can handle inside its memory. You would need *two* recorders hooked up, and would read in as much data as possible (for example, as file #1) on CS1, then do whatever you have to, and finally write the updated data out on CS2 (as a different file number). You would then go back and read in the next batch of data from CS1, update it, and write it out. You repeat this until there is no more data on CS1. This allows a *small* computer to handle very *large* files.

At this point you should have the basic knowledge for choosing a cassette recorder, and getting it to work with your computer. Keep in mind that tape storage transforms your Home Computer into a very powerful and versatile machine. And once you get familiar with the few simple procedures and precautions, each occasion of saving and loading programs and data files will become second nature . . . one might even say, "filled with memories . . . "

Dynamic . . . from p. 77

These numbers are assembled into a 16-space string. This hexadecimal string then goes into a CALL CHAR statement to define a random graphic character.

Shape is forced on the character in lines 19 to 38 by rejecting certain numbers generated by the random number generator. In this particular application, the edges of the characters are "rounded off" so they will not appear square.

I use such random-patterned screen characters to soften up the edges of my "block graphics" designs ("Blockhead graphics?") Another application is to create dramatic effects as is done in the program "Twinkle" given in Listing 4.

I also use random characters to induce variations on things that, as in nature, change with time—shadows or explosions, for instance. Some video games could undoubtedly profit from this technique. I get a little tired of aliens that always blow up the same way. Hmm—Come to think of it, there is that video game with the pigeon in it . . .

RECORDER ROSTRUM

Using Cassette Recorders with the TI-99/4 and TI-99/4A

By F.O. Armbruster

One of the most frustrating things for a beginning computer user is being unable to find the optimum adjustment on a cassette recorder in order to reliably save a program on tape. It took me about two weeks of experimentation before I finally got it all figured out so that I could do it consistently without a hitch. Compounding the difficulty is the fact that some tape recorders have their motors wired up with polarity reversed from that of the computer. In this case an adapter is needed if the computer is to control the tape drive motor. When 99'er-ware introduced the TI SETTE adapter (pronounced "tie-set"), that problem was neatly solved. Since I thought it would be a great help to beginning users if better information on cassette recorders were available, I started testing the various brands and models of cassette recorders for compatibility with my TI-99/4. The results to date are shown in the accompanying chart. If any readers have information to add about a particular make and model, please communicate it to me in care of *99'er Magazine* and I will try to keep the listing up to date.

Name Model	Requires TI-SETTE (page 92)	Comments, Features	Price Range
G.E. 3-5361	no	small, nice design, works fine, tone cont.	$40-50
G.E. 3-5105 F	no	older model discontinued, works fine	$50-60
G.E. 3-5104 C	yes	no tone control, touchy on playback volume setting.	$25-35
G.E. 3-5151 A	yes	has tone control, works fine	$40-50
G.E. 3-5307	yes	compact, hard to connect, works okay	$60-70
G.E. 5154 A	no	also known as Slimline S-2, works fine, not avail. in some areas	$30-40
G.E. 5005 B	yes	no tone control, touchy on volume control	$25-35
Panasonic RQ335	no	touchy on volume control, otherwise okay very compact, good for travel	$50-60
Panasonic RQ337	no	has dual sensitivity on mic., no tone control, works okay	$50-60
Panasonic RQ 309AS	no	touchy on volume setting, otherwise okay	$40-50
J.C. Penny # 681-3246	no	best tested, very forgiving of volume setting errors	$40-50
Realistic CTR-55	no	no tone control, works okay	$20-30
Realistic CTR-37	no	no tone control, works okay	$30-40
Craig J100	yes	no tone control, works okay	$30-40
Craig J103	yes	small, tone control, nice design, works fine	$70-80

Listing 1

```
1 REM  ****************.
2 REM  *  XMAS-TREE   *
3 REM  ****************
4 REM  BY FRED ELLIS
5 REM   99'ER VERSION 1.6.1
6 REM  ABOUT 7568 BYTES
7 REM  PRESS ANY KEY TO END
      SCREEN DISPLAY.
8 CALL SCREEN(2)
9 CALL COLOR(9,7,1)
10 CALL COLOR(10,14,1)
11 CALL COLOR(11,14,4)
12 CALL COLOR(12,12,4)
13 CALL COLOR(13,2,4)
14 CALL COLOR(14,7,4)
15 CALL COLOR(15,15,1)
16 CALL COLOR(16,5.16)
17 REM  PATTERN-IDENTIFIERS
18 REM  FORMAT: IDENIFICATION,
   CHARACTER NUMBER,
   HEXEDECIMAL STRING...
19 REM  EXAMPLE: TREE TRUNK,
   142.4E53B635C659487A,
   TREE BODY,143,
   0000000000000000...
20 DATA LEFTSIDE OUT,96,
   0107070F0F0F1F7F,
   RIGHTSIDE OUT,97,
   B0E0E0F0F0F0F8FE,BORDE R
   TOP,98,7F7F3F0707010100
21 DATA TREE BOTTOM,99,
   FFF8F8F0F0E0C000,B,100,
   FF3F0F0707030100
22 DATA BORDER TOP,102,
   FFFFFFFFFF000000,BORDER
   BOTTOM,103,000000FFFFFFFFFF
23 DATA PLUM,104,
   10FEFEFEFE7C3810,BELL,107,
   101038387C7C7CFE
24 DATA PLUM,112,10FEFEFEFE7C3810,DIAMOND,
   113,1010387CFE7C3810,BELL,115,10103838 7C7C7CFE
25 DATA PLUM,120,10FEFEFEFE7C3810,
   DIA,121,1010387CFE7C3810,BELL,
   123,101038387C7CFEFE
26 DATA PLUM,128,10FEFEFEFE7C3810,
   DIAMOND,129,1010387CFE7C3810,
   BELL,131,101038387C7C7CFE
27 DATA LEFT INSIDE,136,
   FEF8F0F0E0E0C000,RIGHT INSIDE,
   137,7F1F0F0F07070300,CT,138,
   FFE7E7B301010101
28 DATA BOTTOM IN,139,
   0107070F0F0F3FFF,B,140,
   B0C0E0E0E0F0FCFF
29 DATA TRUNK,142,4E53B635C659487A,
   TREE BODY,143,0000000000000000,
   POT L,144,3F3F3F3F3F3F3F3F
30 DATA POT LEFT BOTTOM,146,
   3F3F3F3F3F3F0F0F,POT R,147,
   FCFCFCFCFCFCFCFC
31 DATA POT R B,149,
   FCFCFCFCFCFCF0F0,P B,150,
   FFFFFFFFFFFFFFFF
32 DATA TOP,152,C3C33C18183CC3C3,
   STAR RADIAL,153,
   FFFFFF0000FFFFFF,
   STAR RADIAL,154,
   E7E7E7E7E7E7E7E7
33 REM DEFINE-LOOP
34 RESTORE 20
35 FOR CODE=96 TO 154
36 READ IDENTIFICATION$,
   CHARACTERNUMBER,HEX$'
37 IF CHARACTERNUMBER>CODE
   THEN 39
38 GOTO 40
39 CODE=CHARACTERNUMBER
40 CALL CHAR(CODE,HEX$)
41 NEXT CODE
42 REM  START SCREEN DISPLAY
43 CALL CLEAR
44 REM  ----TREE BODY----
45 CALL HCHAR(24,1,143,32)
46 CALL HCHAR(19,6,143)
47 CALL HCHAR(18,2,143,11)
48 CALL HCHAR(17,3,143,9)
49 CALL HCHAR(16,3,143,8)
50 CALL HCHAR(15,4,143,7)
51 CALL HCHAR(14,4,143,7)
52 CALL HCHAR(13,4,143,6)
53 CALL HCHAR(12,5,143,5)
54 CALL HCHAR(11,6,143,4)
55 CALL HCHAR(10,6,143,3)
56 CALL HCHAR(9,6,143,3)
57 CALL HCHAR(8,7,143)
58 REM  ----TREE TRUNK----
59 CALL VCHAR(20,7,142,2)
60 REM  ----PLANT POT----
61 CALL VCHAR(22,6,144,2)
62 CALL VCHAR(22,7,150,3)
63 CALL VCHAR(22,8,147,2)
64 REM  SCREEN LOCATION DATA
65 REM  FORMAT:
   IDENTIFICATION$,ROW.
   COLUMN,CHARACTERNUMBER...
66 DATA POT BASE LEFT SIDE,
   24,6,146,POT BASE RIGHT
   SIDE,24,8,149
67 REM  ----FOLLAGE----
68 DATA LO,18,1,96,LI,18,2,
   136,LO,17,1,96,LI,17,2,
   136,LO,16,1,96,LI,16,2,
   136,LO,15,2,96,LI,15,3,136
69 DATA LO,14,2,96,LI,14,3,
   136,LO,13,2,96,LI,13,3,
   136,LO,12,3,96,LI,12,4,
   136,LO,11,4,96,LI,11,5,136
```

Dynamic

```
70 DATA LO,10,4,96,LI,10,5,
   136,LO,9,4,96,LI,9,5,136
71 DATA LO,5,6,96,CT,5,7,138,
   LO,6,6,96,LI,6,7,136,LO,7,
   5,96,LI,7,6,136,
   LO,8,5,96,LI,8,6,136
72 DATA RO,5,8,97,RO,6,8,97,
   RI,6,7,137,RO,7,8,97,RI,7,
   7,137,RO,8,9,97,
   RI,8,8,137,RO,9,10,97
73 DATA RI,9,9,137,RO,10,10,
   97,RI,10,9,137,RO,11,11,
   97,RI,11,10,137,
   RO,12,11,97, RI,12,10,137
74 DATA RO,13,11,97,RI,13,10,
   137,RO,14,12,97,RI,14,11,
   137,RO,15,12,97,
   RI,15,11,137,LO,16,12,97
75 DATA LI,16,11,137,LO,17,
   13,97,LI,17,12,137,LO,18,
   13,97,B,18,12,137,
   B,19,12,99,BI,19,11,139
76 DATA BI,19,10,140,
   BO,19,9,100
77 DATA BOTTOM,20,5,96,B,20,
   6,99,B,19,7,139,B,19,8,99,
   B,19,4,100,B,19,5,140,
   B,19,2,96,B,19,3,99
78 DATA B IN,18,7,139,
   B OUT,18,8,136
79 REM  ----CROSS----
80 DATA TOP,2,7,152,L RADIAL,
   2,6,153,R RADIAL,2,8,153,
   T RADIAL,1,7,154,B RADIAL,
   3,7,154,B RAD,4,7,154
81 REM  ----ORNAMENTS----
82 DATA OUTSIDE BELL,7,9,104,
   PLUM,20,2,107,PLUM,14,4,
   112,DIAMOND,13,9,113,
   BELL,16,10,115
83 DATA PLUM,13,4,112,
   PLUM,12,9,128,
   DIAMOND,14,4,129,
   BELL,17,3,131
84 REM SCREEN LOCATION LOOP
85 HOWMANY=86
86 RESTORE 66
87 FOR CHARACTER=1 TO HOWMANY
88 READ IDENTIFICATION$,ROW,
   COLUMN,CHARACTERNUMBER
89 CALL HCHAR(ROW,COLUMN,
   CHARACTERNUMBER)
90 NEXT CHARACTER
91 CALL KEY(0,K,S)
92 IF S=0 THEN 91
93 END
```

Listing 2

```
1 REM  **************
2 REM  * BAR-TOPPER *
3 REM  **************
4 REM  BY FRED ELLIS
5 REM   99'ER VERSION 1.6.1
6 REM  ABOUT 5392 BYTES
7 REM  PRESS ANY KEY TO
       STOP DISPLAY.`
8 VERTICALMAX=200
9 SCALE=VERTICALMAX/20
10 CALL CLEAR
11 LABEL$="ENTER HORSEPOWER"
12 ROW=12
13 COLUMN=15
14 GOSUB 91
15 LABEL$="0 TO 209"
16 ROW=13
17 COLUMN=19
18 GOSUB 91
19 INPUT "                ":
   HORSEPOWER
20 CALL SCREEN(8)
21 CALL COLOR(9,13,8)
22 CALL COLOR(10,2,5)
23 REM DEFINE CHARACTERS
24 REM  FORMAT:
   IDENTIFICATION$,
   CHARACTERNUMBER,
   HEXADECIMAL$...
25 REM  ---GRID---
26 DATA GRID LINE,91,
   00000000000000FF,
   VERTICAL AXIS,92,
   0101010101010101,
   TIC MARK,93,
   010101010101017F
27 REM  --DEFINE BAR TOPS--
28 DATA BOTTOM ROW OF PIXELS
   ON,96,00000000000000FF,
   SECOND ROW ON,97,
   000000000000FFFF,
   THIRD ROW ON
29 DATA 98,0000000000FFFFFF,
   FOURTH,99,
   00000000FFFFFFFF,FIFTH,
   100,000000FFFFFFFFFF,
   SIXTH,101,
   0000FFFFFFFFFFFF
30 DATA SEVENTH,102,
   00FFFFFFFFFFFFFF,
   EIGHTH,103,
   FFFFFFFFFFFFFFFF
31 REM  ---BASELINE---
32 DATA BASE,104,
   FF0000FF000000FF
33 REM DEFINE LOOP
34 RESTORE 26
35 FOR CODE=91 TO 104
36 READ IDENTIFICATION$,
   CHARACTERNUMBER,HEX$
37 IF CHARACTERNUMBER>CODE
   THEN 39
38 GOTO 40
39 CODE=CHARACTERNUMBER
40 CALL CHAR(CODE,HEX$)
41 NEXT CODE
42 REM START SCREEN DISPLAY
43 REM  ---GRAPH GRID---
44 CALL HCHAR(22,13,104,18)
45 FOR ROW=21 TO 1 STEP -1
46 CALL HCHAR(ROW,14,91,17)
47 NEXT ROW
48 LABEL$="HORSEPOWER"
49 ROW=9
50 COLUMN=1
51 GOSUB 91
52 CALL VCHAR(1,13,92,21)
53 FOR ROW=21 TO 1 STEP -5
54 ROWNUMBER=200-(10*(ROW-1))
55 LABEL$=STR$(ROWNUMBER)
56 COLUMN=10
57 GOSUB 91
58 CALL HCHAR(ROW,13,93)
59 NEXT ROW
60 REM CALCULATE BAR HEIGHT
61 BARHEIGHT=HORSEPOWER/SCALE
62 Y=INT(BARHEIGHT)
63 REMAINDER=BARHEIGHT-INT
   (BARHEIGHT)
64 CALL VCHAR(22-Y,16,103,Y)
65 CALL VCHAR(22-Y,17,103,Y)
66 CALL VCHAR(22-Y,18,103,Y)
67 REM SELECT BAR TOP
68 TOPPATTERN=INT
   ((REMAINDER*8)+.5)
69 ON TOPPATTERN+1 GOTO 70,
   72,74,76,78,80,82,84,86
70 CALL HCHAR(21-Y,16,96,3)
71 GOTO 88
72 CALL HCHAR(21-Y,16,97,3)
73 GOTO 88
74 CALL HCHAR(21-Y,16,98,3)
75 GOTO 88
76 CALL HCHAR(21-Y,16,99,3)
77 GOTO 88
78 CALL HCHAR(21-Y,16,100,3)
79 GOTO 88
80 CALL HCHAR(21-Y,16,101,3)
81 GOTO 88
82 CALL HCHAR(21-Y,16,102,3)
83 GOTO 88
84 CALL HCHAR(21-Y,16,103,3)
85 GOTO 88
86 CALL HCHAR(21-Y,16,103,3)
87 CALL HCHAR(20-Y,16,96,3)
88 CALL KEY(0,K,S)
89 IF S=0 THEN 88
90 END
91 FOR POSITION=1 TO
   LEN(LABEL$)
92 LETTER$=SEG$(LABEL$,
   POSITION,1)
93 CODE=ASC(LETTER$)
94 CALL HCHAR(ROW,
   COLUMN-1+POSITION,CODE)
95 NEXT POSITION
96 RETURN
```

Listing 3

```
1 REM  *************
2 REM  * AUTO-TOP *
3 REM  *************
4 REM  BY FRED ELLIS
5 REM   99'ER VERSION 1.6.1
6 REM  ABOUT 5288 BYTES
7 REM  PRESS ANY KEY TO STOP
       DISPLAY.
8 VERTICALMAX=200
9 SCALE=VERTICALMAX/20
10 CALL CLEAR
11 LABEL$="ENTER HORSEPOWER"
12 ROW=12
13 COLUMN=15
14 GOSUB 78
15 LABEL$="0 TO 209"
16 ROW=13
17 COLUMN=19
18 GOSUB 78
19 INPUT "                ":
   HORSEPOWER
20 CALL SCREEN(8)
21 CALL COLOR(9,13,8)
22 CALL COLOR(10,2,5)
23 REM DEFINE CHARACTERS
24 REM  FORMAT:
   IDENTIFICATION$,
   CHARACTERNUMBER,
   PATTERN$...
25 DATA GRID LINE,91,
   00000000000000FF,
   VERTICAL AXIS,92,
   0101010101010101,
   TIC MARK,93,
   010101010101017F
26 DATA BAR,96,
   FFFFFFFFFFFFFFFF,
   BASELINE,104,
   FF0000FF000000FF
27 DATA RESERVED FOR
   TITLE BOX
28 DATA RESERVED FOR LABELS
29 DATA RESERVED FOR LEGEND
30 DATA RESERVER FOR
   ADDITIONAL CHARACTERS
31 REM DEFINE-LOOP
32 RESTORE 25
33 FOR CODE=91 TO 104
34 READ IDENTIFICATION$,
   CHARACTERNUMBER,PATTERN$
35 IF CHARACTERNUMBER>CODE
   THEN 37
36 GOTO 38
37 CODE=CHARACTERNUMBER
38 CALL CHAR(CODE,PATTERN$)
39 NEXT CODE
40 REM START SCREEN DISPLAY
41 REM  ---GRAPH GRID---
42 CALL HCHAR(22,13,104,18)
43 FOR ROW=21 TO 1 STEP -1
44 CALL HCHAR(ROW,14,91,17)
45 NEXT ROW
46 LABEL$="HORSEPOWER"
47 ROW=9
48 COLUMN=1
49 GOSUB 78
50 CALL VCHAR(1,13,92,21)
51 FOR ROW=21 TO 1 STEP -5
52 ROWNUMBER=200-(10*(ROW-1))
53 LABEL$=STR$(ROWNUMBER)
54 COLUMN=10
55 GOSUB 78
56 CALL HCHAR(ROW,13,93)
57 NEXT ROW
58 REM CALCULATE BAR HEIGHT
59 BARHEIGHT=HORSEPOWER/SCALE
60 Y=INT(BARHEIGHT)
61 REMAINDER=BARHEIGHT-INT
   (BARHEIGHT)
62 CALL VCHAR(22-Y,16,96,Y)
63 CALL VCHAR(22-Y,17,96,Y)
64 CALL VCHAR(22-Y,18,96,Y)
65 REM  SELECT BAR TOP
66 TOPPATTERN=1+INT
   ((REMAINDER*8)+.5)
67 MASTER$="0000000000000
   FFFFFFFFFFFFFFFFF"
68 STARTPOSITION=2*
   TOPPATTERN-1
69 TOPPATTERN$=SEG$(MASTER$,
   STARTPOSITION,16)
70 CALL CHAR(97,TOPPATTERN$)
71 CALL HCHAR(21-Y,16,97,3)
72 IF TOPPATTERN<>9 THEN 75
73 CALL CHAR(98,
   "00000000000000FF")
74 CALL HCHAR(20-Y,16,98,3)
75 CALL KEY(0,K,S)
76 IF S=0 THEN 75
77 END
78 FOR POSITION=1 TO
   LEN(LABEL$)
79 LETTER$=SEG$
   (LABEL$,POSITION,1)
80 CODE=ASC(LETTER$)
81 CALL HCHAR(ROW,
   COLUMN-1+POSITION,CODE)
82 NEXT POSITION
83 RETURN
```

Listing 4

```
1 REM  **************
2 REM  * THREE-BARS *
3 REM  **************
4 REM  BY FRED ELLIS
5 REM   99'ER VERSION 1.6.1
6 REM  ABOUT 5160 BYTES
7 REM  PRESS ANY KEY TO STOP
       DISPLAY.
8 VERTICALMAX=200
9 SCALE=VERTICALMAX/20
10 OPTION BASE 1
11 DIM Y(3)
12 Y(1)=133
13 Y(2)=159
14 Y(3)=99.9
15 CALL SCREEN(8)
16 CALL COLOR(9,5,8)
17 CALL COLOR(10,3,8)
18 CALL COLOR(11,16,8)
19 CALL COLOR(12,2,5)
20 REM DEFINE CHARACTERS
21 REM  FORMAT:
   IDENTIFICATION$,
   CHARACTERNUMBER,
   PATTERN$...
22 DATA GRID LINE,91,
   00000000000000FF,
   VERTICAL AXIS,92,
   0101010101010101,
   TIC MARK,93,
   010101010101017F,
   BAR1,96
23 DATA FFFFFFFFFFFFFFFF,
   BAR2,104,
   FFFFFFFFFFFFFFFF,
   BAR3,112,
   FFFFFFFFFFFFFFFF,
   BASELINE,120,
   FF0000FF000000FF
24 DATA RESERVED FOR
   TITLE BOX
25 DATA RESERVED FOR LABELS
26 DATA RESERVED FOR LEGEND
27 DATA RESERVER FOR
   ADDITIONAL CHARACTERS
28 REM DEFINE-LOOP
29 RESTORE 22
30 FOR CODE=91 TO 120
31 READ IDENTIFICATION$,
   CHARACTERNUMBER,PATTERN$
32 IF CHARACTERNUMBER>CODE
   THEN 34
33 GOTO 35
34 CODE=CHARACTERNUMBER
35 CALL CHAR(CODE,PATTERN$)
36 NEXT CODE
37 REM START SCREEN DISPLAY
38 CALL CLEAR
39 PRINT TAB(13);Y(1);
   TAB(18);Y(2);Y(3)
40 REM  ---GRAPH GRID---
41 CALL HCHAR(22,13,120,18)
42 FOR ROW=21 TO 1 STEP -1
43 CALL HCHAR(ROW,14,91,17)
44 NEXT ROW
45 LABEL$="HORSEPOWER"
46 ROW=9
47 COLUMN=1
48 GOSUB 78
49 CALL VCHAR(1,13,92,21)
50 FOR ROW=21 TO 1 STEP -5
51 ROWNUMBER=200-(10*(ROW-1))
52 LABEL$=STR$(ROWNUMBER)
53 COLUMN=10
54 GOSUB 78
55 CALL HCHAR(ROW,13,93)
56 NEXT ROW
57 REM CALCULATE & PLOT BARS
58 MASTER$="0000000000000
   FFFFFFFFFFFFFFFFF"
59 FOR BAR=1 TO 3
```

Continued on p. 86

Crayon . . . from p. 83

```
LOOP7  MOVB @GRAY,*R1+          STORE A GRAY BYTE
       DEC  R0                  COUNT DOWN
       JNE  LOOP7               REPEAT UNTIL DONE
*
* INITIALIZE PATTERN TABLE - TRANSPARENT
*
CLEAR  LI   R0,>300             INITIALIZE WORD COUNTER
       LI   R1,PATRN            INITIALIZE POINTER FOR PATTERN ARRAY
LOOP8  MOV  @ZERO,*R1+          STORE COLOR = TRANSPARENT
       DEC  R0                  COUNT DOWN FOR NEXT WORD
       JNE  LOOP8               WRITE NEXT WORD IN ARRAY
*
* LOAD PATTERN TABLE
*
       LI   R0,>800             VDP PATTERN TABLE ADDRESS
       LI   R1,PATRN            CPU BUFFER ADDRESS
       LI   R2,>600             1536 BYTES TO WRITE
       BLWP @VMBW               WRITE TO VDP RAM
*
* SELECT DOUBLE SIZE AND MULTICOLOR MODE
*
       LI   R0,>01EA            TO WRITE 11101010 TO VDP R1
       BLWP @VWTR               WRITE TO VDP R1
       SWPB R0                  MOVE >EA TO MOST SIG BYTE
       MOVB R0,@>83D4           STORE COPY (>EA) IN CPU RAM
*
* DEFINE ATTRIBUTES FOR SPRITE #0
*
       LI   R0,>300             VDP SPRITE ATTRIBUTE LIST
       LI   R1,ATTRIB           LOCATION OF ATTRIBUTE LIST
                                FOR SPRITE 0
       LI   R2,6                6 BYTES TO MOVE
       BLWP @VMBW               WRITE DATA TO VDP RAM
*
* DEFINE # OF ACTIVE SPRITES
*
       MOVB @ONE,@SPRITE        STORE NO. OF ACTIVE SPRITES
                                IN CPU RAM
*
* INITIALIZE CURSOR COLOR AND COLOR CHANGE COUNTER
*
       LI   R3,>0F01            SPRITE COLORS - WHITE/BLACK IN R3
       CLR  R4                  INITIALIZE COUNTER - COLOR CHANGE
*
* ----------- START MAIN LOOP ----------------
*
* CHECK JOYST FOR MOTION, FIRE BUTTON AND KEYS
*
CHECK  LIMI 2                   ENABLE INTERRUPTS
       LIMI 0                   DISABLE INTERRUPTS
       LI   R0,1                INDICATE REPETIONS OF CHECKS
       BL   @CHECKS             BRANCH TO SUBROUTINE CHECKS
       MOVB @ONE,@UNIT          SELECT REMOTE UNIT TO SCAN
       BLWP @KSCAN              SCAN LEFT KEYBOARD
       CB   @FIRE,@H05          WAS "E" PRESSED?
       JEQ  CLEAR               IF YES GO TO CLEAR SCREEN
       CB   @FIRE,@H02          WAS "S" PRESSED?
       JNE  NEXT1               IF NOT, GO ON
       B    @SAVE               IF SO, BRANCH TO SAVE ROUTINE
NEXT1  CB   @FIRE,@H06          WAS "R" PRESSED?
       JNE  NEXT2               IF NOT, GO ON
       B    @RECALL             IF SO, BRANCH TO RECALL ROUTINE
NEXT2  CB   @FIRE,@H11          WAS "T" PRESSED?
       JNE  NEXT3               IF NOT, GO ON
       LIMI 2                   ENABLE INTERRUPTS
       LWPI GPLWS               LOAD GPL WORK SPACE
       BLWP @0000               RETURN TO MASTER TITLE SCREEN
NEXT3  CB   @FIRE,@H14          WAS "C" PRESSED?
       JNE  NEXT4               IF NO, GO ON
       B    @SELECT             IF YES, GO TO COLOR SELECT ROUTINE
NEXT4  CB   @FIRE,@H18          WAS FIRE BUTTON PRESSED?
       JNE  SKIP                IF NO, SKIP DRAW ROUTINE
*
* ROUTINE TO PLACE BLOCK ON SCREEN
*
DRAW   LI   R0,>300             VDP SPRITE ATTRIBUTE ADDRESS
       LI   R1,ROW              CPU BUFFER TO RECEIVE DATA
       LI   R2,2                FETCH 2 BYTES
       BLWP @VMBR               FETCH DOT ROW AND DOT COLUMN
       CLR  R7                  INITIALIZE R7 AND R8
       CLR  R8                  --FOR USE IN DIVIDE OPERATION
       CLR  R2                  INITIALIZE OFFSET FOR PATRN ARRAY
       MOVB @ROW,R8             PUT DOT ROW IN R8
       SWPB R8                  MAKE IT LEAST SIG BYTE
       AI   R8,9                ADD ROW OFFSET FOR COLOR BLOCK +1
       C    R8,@COLMAX          IS THE DOT ROW > 255?
       JLT  NOCORR              IF NOT, DO NOT APPLY CORRECTION
       S    @COLMAX,R8          IF SO, SUBTRACT 255
NOCORR DIV  @D32,R7             DIVIDE DOT ROW OF BLOCK BY 32
       SLA  R7,8                CALCULATE BYTES IN PRECEEDING GROUPS
       A    R7,R2               ADD # OF BYTES IN PREVIOUS 32X8 BYTE
                                GROUPS
       SRL  R8,2                DIVIDE REMAINDER BY 4
       A    R8,R2               ADD # BYTES ABOVE IN CURRENT 8 BYTE
                                SET
       CLR  R7                  INITIALIZE R7 AND R8
       CLR  R8                  --FOR USE IN DIVIDE OPERATION
       MOVB @COL,R8             PUT DOT COLUMN IN R8
       SWPB R8                  MAKE IT LEAST SIG BYTE
       AI   R8,8                ADD COLUMN OFFSET FOR COLOR BLOCK
       C    R8,@COLMAX          IS THE DOT COLUMN > 255?
       JLT  NOCORC              IF NOT, DO NOT APPLY CORRECTION
       S    @COLMAX,R8          IF SO, SUBTRACT 256
NOCORC DIV  @D8,R7              DIVIDE BY 8
       SLA  R7,3                CALCULATE BYTES IN PRECEEDING 8 BYTE
                                SETS
       A    R7,R2               ADD # BYTES IN PREVIOUS 8 BYTE SETS,
                                THIS GROUP
       MOV  R2,R2               CHECK IF INSIDE PATTERN ARRAY  N
       JLT  SKIP                IF NOT SKIP SCREEN PLACEMENT
       C    R2,@MAX             CHECK IF INSIDE PATTERN ARRAY EEN
       JGT  SKIP                IF NOT SKIP SCREEN PLACEMENT
       LI   R0,>14              REPEAT SUBROUTINE CHECKS 20 TIMES
       BL   @CHECKS             BRANCH TO SUBROUTINE CHECKS
       CLR  R1                  INITIALIZE R1 FOR BLOCK COLOR
       MOVB @FCOLOR,R1          STORE COLOR IN R1
       SWPB R1                  MAKE IT LEAST SIG BYTE
       CLR  R0                  INITIALIZE R0 FOR CURRENT ARRAY
                                ELEMENT
       MOVB @PATRN(2),R0        COPY ARRAY ELEMENT AT OFFSET INTO R0
       SRL  R8,2                CALCULATE WHETHER BLOCK IS LEFT OR
                                RIGHT
       JEQ  MARK1               IF 0 LEAVE BLOCK AS LEFT NYBBLE
       SRL  R1,4                IF 1 MAKE BLOCK RIGHT NYBBLE
       SWPB R0                  MAKE CURRENT ELEMENT LEAST SIG BYTE
       SRL  R0,4                GET RID OF LEAST SIG NYBBLE
       SLA  R0,4                PUT REMAINING NYBBLE BACK
       JMP  MARK2               SKIP TO LABEL
```

Continued on p. 94

Dynamic . . . from p. 85

```
60 BARHEIGHT=Y(BAR)/SCALE
61 YY=INT(BARHEIGHT)
62 REMAINDER=BARHEIGHT-INT(BARHEIGHT)
63 CALL VCHAR(22-YY,11+BAR*5,
   88+BAR*8,YY)
64 CALL VCHAR(22-YY,12+BAR*5,
   88+BAR*8,YY)
65 CALL VCHAR(22-YY,13+BAR*5,
   88+BAR*8,YY)
66 TOPPATTERN=1+INT((REMAINDER*8)+.5)
67 STARTPOSITION=2*TOPPATTERN-1
68 TOPPATTERN$=SEG$(MASTER$,
   STARTPOSITION,16)
69 CALL CHAR(89+BAR*8,TOPPATTERN$)
70 CALL HCHAR(21-YY,11+BAR*5,
   89+BAR*8,3)
71 IF TOPPATTERN<>9 THEN 74
72 CALL CHAR(90+BAR*8,
   "00000000000000FF")
73 CALL HCHAR(20-YY,11+BAR*5,
   90+BAR*8,3)
74 NEXT BAR
75 CALL KEY(0,K,S)
76 IF S=0 THEN 75
77 END
78 FOR POSITION=1 TO LEN(LABEL$)
79 LETTER$=SEG$(LABEL$,POSITION,1)
80 CODE=ASC(LETTER$)
81 CALL HCHAR(ROW,
   COLUMN-1+POSITION,CODE)
82 NEXT POSITION
83 RETURN
```

Listing 5

```
1 REM  ***********
2 REM  * TWINKLE *
3 REM  ***********
4 REM  FRED ELLIS
5 REM   99'ER VERSION 1.6.1
6 REM  ABOUT 4504 BYTES
7 REM  HOLD DOWN ANY KEY TO STOP
       DISPLAY
8 CALL SCREEN(8)
9 CALL COLOR(9,10,8)
10 REM SHAPED RANDOM CHARACTERS
11 RANDOMIZE
12 FOR J=1 TO 4
13 FOR I=0 TO 15
14 REM  GENERATE RANDOM NUMBERS
   BETWEEN 0 AND 15
15 N=INT((15-0+1)*RND)+0
16 REM  PUT ON CONSTRAINTS TO
   ELIMANATE CORNERS
17 ON I+1 GOTO 19,21,25,27,32,34,
   40,40,40,40,32,34,25,27,19,21
18 REM  TOP & BOTTOM ROWS
19 IF N>1 THEN 15
20 GOTO 40
21 IF N=0 THEN 40
22 IF N=8 THEN 40
23 GOTO 15
24 REM  2ND & 7TH ROWS
25 IF N>3 THEN 15
26 GOTO 40
27 IF N>12 THEN 15
28 FOURTEST=N/4-INT(N/4)
29 IF FOURTEST=0 THEN 40
30 GOTO 15
31 REM  3RD & 6TH ROWS
32 IF N>7 THEN 15
33 GOTO 40
34 IF N=15 THEN 15
35 EVENTEST=N/2-INT(N/2)
36 IF EVENTEST=0 THEN 40
37 GOTO 15
38 REM  4TH & 5TH ROWS NO
   CONSTRAINTS
39 REM  FOR N>9 MUST CONVERT
   TO HEX NOTATION, NOTE IN HEX
   NOTATION A=10,B=11,C=12 ETC.
40 IF N>9 THEN 41 ELSE 54
41 ON N-9 GOTO 42,44,46,48,50,52
42 G$="A"
43 GOTO 55
44 G$="B"
45 GOTO 55
46 G$="C"
47 GOTO 55
48 G$="D"
49 GOTO 55
50 G$="E"
51 GOTO 55
52 G$="F"
53 GOTO 55
54 G$=STR$(N)
55 HEX$=HEX$&G$
56 NEXT I
57 CALL CHAR(95+J,HEX$)
58 GOTO 59
59 HEX$=""
60 NEXT J
61 REM DISPLAY TITLE
62 DATA 57,57,39,69,82,32,77,65,71,65,90,73,78,69
63 CALL CLEAR
64 REM  ... BORDER...
65 FOR COL=6 TO 26
66 N=INT((4-1+1)*RND)+1
67 CALL HCHAR(14,32-COL,95+N)
68 CALL HCHAR(10,COL,95+N)
69 NEXT COL
70 FOR ROW=10 TO 14
71 N=INT(4*RND+1)
72 CALL VCHAR(ROW,6,95+N)
73 CALL VCHAR(24-ROW,26,95+N)
74 NEXT ROW
75 REM  ...TITLE...
76 CALL HCHAR(12,10,32,14)
77 RESTORE 62
78 FOR I=1 TO 14
79 READ LETTER
80 COLUMN=9+I
81 CALL HCHAR(12,COLUMN,96)
82 CALL HCHAR(12,COLUMN,LETTER)
83 NEXT I
84 REM  ...TWINKLE...
85 C=0
86 COLUMN=INT((26-6+1)*RND)+6
87 N=INT((4-1+1)*RND)+1
88 CALL HCHAR(10,COLUMN,95+N)
89 CALL HCHAR(14,32-COLUMN,95+N)
90 C=C+1
91 IF C>45 THEN 76
92 CALL KEY(0,K,S)
93 IF K>31 THEN 95
94 GOTO 86
95 END
```

Verbose . . . from p. 56

```
400 CALL CLEAR
410 PRINT "WORD #";J: : :
420 CALL SAY("",F$(J))
430 INPUT "SPELL IT- ":X$
440 IF X$=WORD$(J)THEN 470
450 CALL SAY("UHOH")
460 SCORE=SCORE-1
470 SCORE=SCORE+1
480 PRINT : :"CORRECT SPELLING IS              >>";
    WORD$(J);"<<": :
490 PRINT "YOUR SCORE IS ";SCORE;" OUT OF";J
500 FOR Y=1 TO 500
510 NEXT Y
520 NEXT J
530 CALL CLEAR
540 PRINT "YOU GOT ";INT((SCORE/LAST)*100);
    "% CORRECT": : : : :
550 INPUT "ENTER ""Y"" TO TRY AGAIN ":Z$
560 IF Z$="Y" THEN 330
570 CALL CLEAR
580 END
```

Listing 4

```
100 REM ++++++++++++++++++++++++
110 REM +   . V E R B O S E     +
120 REM +                       +
130 REM + BY DAVID G.BRADER     +
140 REM +                       +
150 REM + 99'ER VERSION 1.6.1   +
160 REM +                       +
170 REM ++++++++++++++++++++++++
180 REM
190 REM SPEECH MAKER  ROUTINE
200 REM WILL COMBINE WORDS INTO NEW STRINGS.
210 REM
220 CALL CLEAR
230 PRINT "    +++ WORD BUILDER +++": : : : :
240 PRINT "ENTER NUMBER OF YOUR CHOICE": :
250 PRINT :"  1 - JOIN TWO WORDS": :
260 PRINT "  2 - PRINT SPEECH DATA": :
270 PRINT "  3 - STORE NEW WORD ON DISK": :
280 PRINT "  4 - EXIT": : :
290 INPUT CHOICE
300 IF (CHOICE<1)+(CHOICE>4)=-1 THEN 220
310 ON CHOICE GOSUB 550,870,1100,330
320 GOTO 220
330 CALL CLEAR
340 END
350 REM +++ TRUNCATE FIRSTWORD +++
360 CALL CLEAR
370 INPUT "TRUNCATE HOW MANY BYTES?":BYTES
380 MAXBYTES=LEN(B$)-3
390 IF BYTES<MAXBYTES THEN 420
400 PRINT "TOO MANY BYTES...."
410 GOTO 370
420 IF BYTES>-1 THEN 450
430 PRINT "NO NEGATIVE NUMBERS"
440 GOTO 370
450 L=MAXBYTES-BYTES
460 C$=SEG$(B$,1,2)&CHR$(L)&SEG$(B$,4,L)
470 RETURN
480 REM +++ SPEAK NEW WORD +++
490 CALL CLEAR
500 CALL SAY("",NEWDATA$)
510 INPUT "SAY AGAIN? (Y OR N)":CHOICE$
520 IF CHOICE$="Y" THEN 500
530 RETURN
540 REM  +++ JOIN TWO WORDS SUBROUTINE +++
550 CALL CLEAR
560 PRINT "ENTER FIRST WORD TO JOIN"
570 INPUT FIRSTWORD$
580 IF FIRSTWORD$=LASTMADE$ THEN 610
590 CALL SPGET(FIRSTWORD$,B$)
600 GOTO 620
610 B$=LASTDATA$
620 CALL CLEAR
630 PRINT "ENTER SECOND WORD TO JOIN"
640 INPUT SECONDWORD$
650 IF SECONDWORD$=LASTMADE$ THEN 680
660 CALL SPGET(SECONDWORD$,D$)
670 GOTO 690
680 D$=LASTDATA$
690 CALL CLEAR
700 PRINT "ENTER THE SPELLING OF THE
    NEW WORD"
710 INPUT NEWWORD$
720 REM
730 GO SUB 350
740 NEWDATA$=C$&D$
750 GO SUB 480
760 PRINT "     1 - CHANGE SOME MORE":
    "     2 - BACK TO MAIN MENU"
770 INPUT CHOICE
780 IF (CHOICE<1)+(CHOICE>2)=-1 THEN 760
790 IF CHOICE=1 THEN 730
800 LASTMADE$=NEWWORD$
810 LASTDATA$=NEWDATA$
820 RETURN
830 REM  +++ PRINT SPEECH DATA SUBROUTINE +++
840 REM ----------------------------------------
850 REM THIS OPEN STATEMENT MAY NEED TO BE MODIFIED
860 REM FOR YOUR PRINTER......
870 OPEN #1:"RS232/2.DA=8.BA=9600"
880 REM ----------------------------------------
890 CALL CLEAR
900 PRINT "ENTER THE WORD WHOSE DATA
    YOU WANT TO PRINT--     ": : :
910 GOSUB 1230
920 IF L=0 THEN 1070
930 VALUES$=""
940 PRINT #1: : :"THE WORD IS ** ";WORD$;" **";
950 PRINT #1:"LENGTH =";L;"BYTES": :
960 FOR I=1 TO L
970 VALUES$=VALUES$&STR$(ASC(SEG$(F$,I,1)))
980 IF I/10<>INT(I/10)THEN 1020
990 PRINT #1:"DATA ";VALUES$
1000 VALUES$=""
1010 GOTO 1030
1020 VALUES$=VALUES$&","
1030 NEXT I
1040 IF VALUES$="" THEN 1070
1050 VALUES$=SEG$(VALUES$,1,LEN(VALUES$)-1)
1060 PRINT #1:"DATA ";VALUES$
1070 CLOSE #1
1080 RETURN
1090 REM  +++ ADD NEW WORD TO VOCABULARY FILE +++
1100 CALL CLEAR
1110 PRINT "PUT THE DISK WITH ""WORDS""     FILE IN DRIVE ONE."
1120 INPUT "PRESS ENTER WHEN READY ":X$
1130 PRINT : : :"ENTER THE WORD WHOSE DATA
     YOU WANT TO SAVE--     ": : :
1140 GOSUB 1230
1150 IF L=0 THEN 1200
1160 OPEN #1:"DSK1.WORDS",INTERNAL,APPEND,VARIABLE 254
1170 PRINT #1:WORD$
1180 PRINT #1:F$
1190 CLOSE #1
1200 RETURN
1210 REM
1220 REM  +++ FIND WORD SUBROUTINE
1230 INPUT WORD$
1240 F$=""
1250 IF WORD$="" THEN 1300
1260 IF WORD$=LASTMADE$ THEN 1290
1270 CALL SPGET(WORD$,F$)
1280 GOTO 1300
1290 F$=LASTDATA$
1300 L=LEN(F$)
1310 RETURN
```

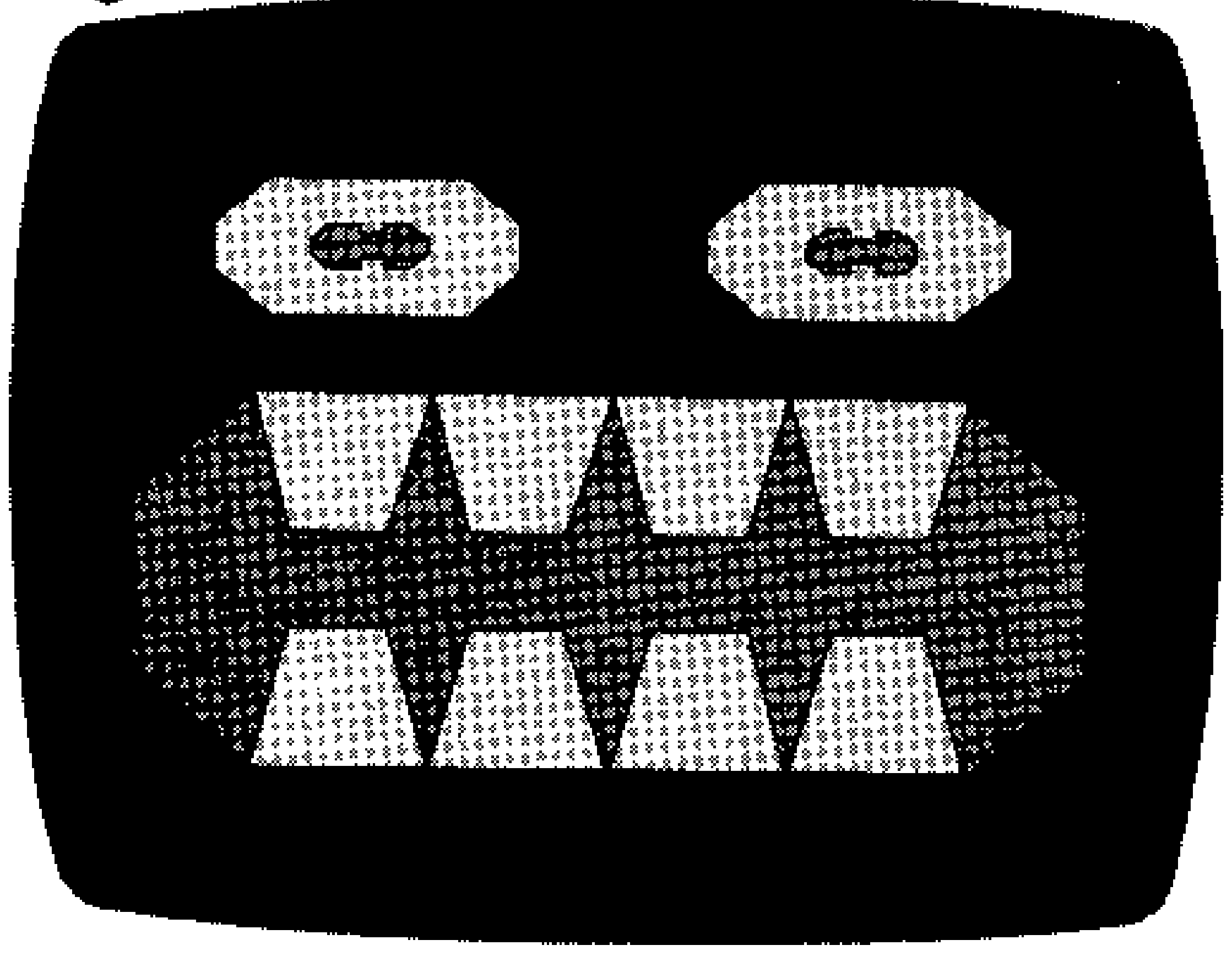
Who . . . from p. 81

Note 1: Due to slight differences in timing between TI-99/4A consoles (depending on date of manufacture), the position of the ball and paddle on "impact" might not be the most realistic for your particular machine. We suggest you try changing TEST XCOR>85 to a slightly lower value, and SXY 100 0 to an X value above 100.

```
TO PADDLE
CALL RC "A
IF :A = "E TELL [1 2 ] FORWARD 1
6 CALL :Y + 16 "Y
IF :A = "X TELL [1 2 ] BACK 16 C
ALL :Y - 16 "Y
END

TO PADDLETOUCH
TELL 0
TEST EITHER YCOR > :Y YCOR < ( :
Y - 32 )
IFT OUTPUT "FALSE
OUTPUT "TRUE
END

TO CHECK
TELL 0
TEST PADDLETOUCH
IFT CALL :PLS + 1 "PLS ; THIS LI
NE INCREASES THEPLAYER'S SCORE
IFT NOISE ; THIS LINE CAUSES NOI
SE TO BE RUN WHEN THE PLAYER SCO
RES
IFT SETHEADING 270
IFF CALL :CPS + 1 "CPS ; THIS LI
NE INCREASES THE COMPUTER'S SCOR
E
IFF BEEP WAIT 10 NOBEEP ; THIS C
AUSES A SHORT BEEP FOR THE COMPU
TER'S POINT
TYPE [YOUR SCORE IS ] PC 32
TYPE :PLS PC 32
TYPE [THE COMPUTER'S SCORE IS ]
PC 32
PRINT :CPS
WAIT 90 CS
END

TO NOISE
REPEAT 5 [BEEP WAIT 3 NOBEEP WAI
T 3 ]
END

TO GAME
TEST RC?
IFT PADDLE
BOUNCE2
GAME
END

TO SETUP
TELL 0
CARRY :BALL
SETCOLOR :BLUE
HOME
SETHEADING 90
SETSPEED 15
TELL [1 2 ]
CARRY :BOX
SETHEADING 0
SETCOLOR :BLACK
SXY 100 0
TELL 2
SY 16
CALL 16 "Y CALL 0 "PLS
CALL 0 "CPS
END

TO BOUNCE2
TELL 0
TEST XCOR > 85
IFT CHECK
TEST XCOR < - 85
IFT SETHEADING 70
TEST YCOR > 90
IFT SETHEADING 135
TEST YCOR < - 85
IFT SETHEADING 45
END

TO PLAY
SETUP
GAME
END
```

Note 2: Don't "wrap" the paddle over the top or under the bottom of the screen, or the game won't work correctly.

THE MONSTER IS FREE!



During the Texas Instruments Home Computer Free Software Offer.

The Monster, otherwise known to his friends as "Munchman," is one of the most fun and exciting games Texas Instruments has ever had: And now, he can be yours at no cost whatsoever during our Texas Instruments Home Computer Free Software Offer. Here's how it works:

If you buy any four Texas Instruments Solid State Software™ modules, or a Texas Instruments software album (three modules in a convenient storage package) between now and August 15, 1982, we'll send you the exciting new "Munchman Arcade Game" absolutely free. The modules you can choose can be from any category: Home Finance, Children's Education or Entertainment. And the Munchman will be on us.

Munchman is an offer worth $39.95, but with a deal like this, people are going to eat it up. So hurry. Take the coupon in this ad to your participating retail outlet... and set the monster free.

TEXAS INSTRUMENTS
INCORPORATED

Mail to:
"Munchman" Offer
P.O. Box 725, Dept. HC-HBO,
Lubbock, Texas 79491

Yes, I want to set the Monster free. I am enclosing a proof-of-purchase with this coupon. Please send "Munchman" to:

Name ______________________

Address ______________________

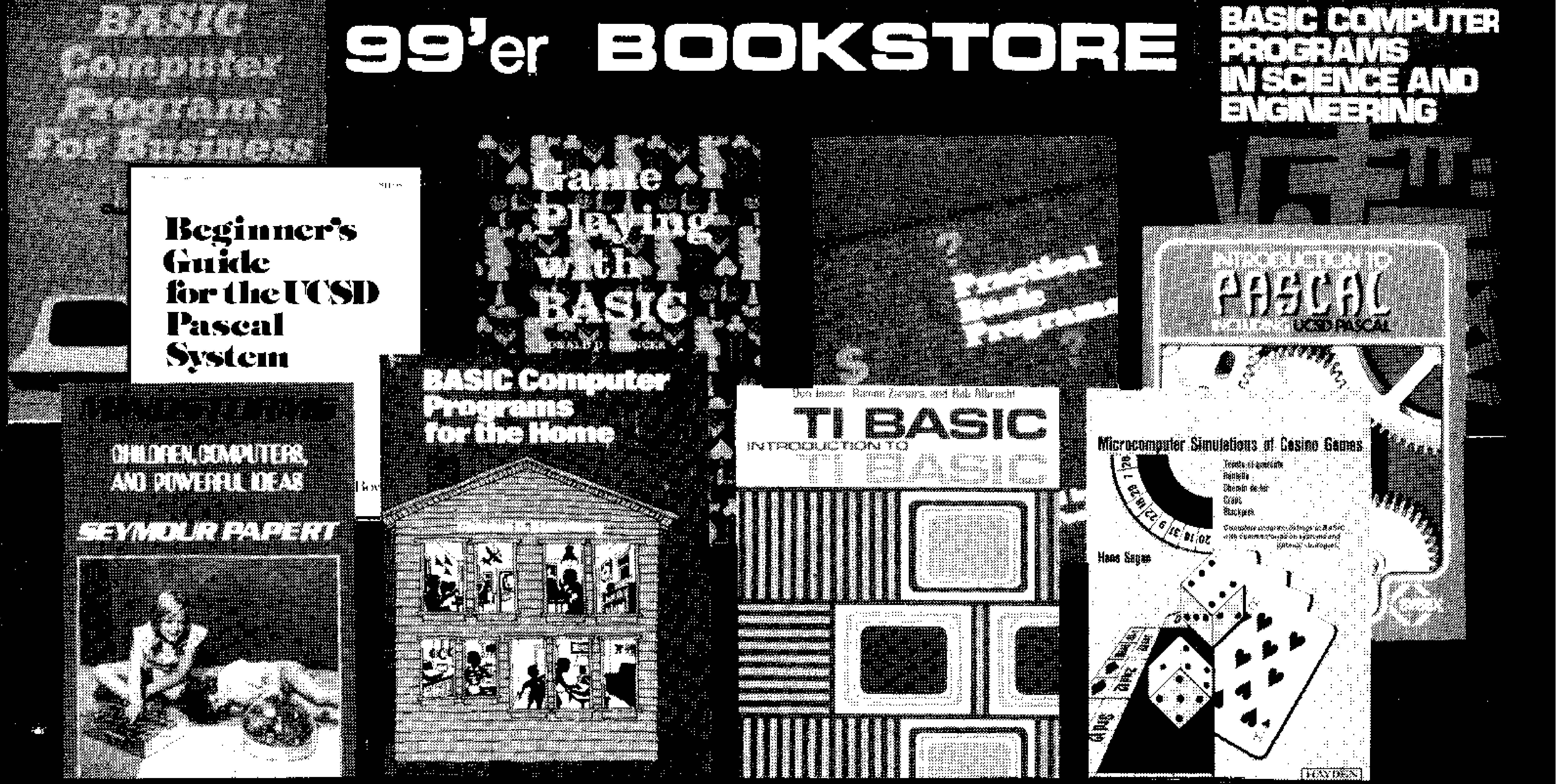
City ______ State ______ Zip ______ Phone ______

Serial # of TI 99/4 Console ______ Age of Buyer ______ Occupation ______

No. of children ______ Ages ______ Name of store where purchased ______

Intended Use ______ Home ______ Personal Finance ______ Education ______

99'er BOOKSTORE


BASIC Computer Programs For Business
BASIC COMPUTER PROGRAMS IN SCIENCE AND ENGINEERING
Beginner's Guide for the UCSD Pascal System
Game Playing with BASIC
Practical Basic Programs
INTRODUCTION TO PASCAL INCLUDING UCSD PASCAL
MINDSTORMS CHILDREN, COMPUTERS, AND POWERFUL IDEAS
SEYMOUR PAPERT
BASIC Computer Programs for the Home
Don Inman, Ramon Zamora, and Bob Albrecht
INTRODUCTION TO TI BASIC
Microcomputer Simulations of Casino Games
Hans Sagan
HAYDEN

MINDSTORMS: CHILDREN, COMPUTERS AND POWERFUL IDEAS

By Seymour Papert

The definitive work on the philosophy behind LOGO. Excerpted in the Vol. 1, No. 1 issue of this magazine.

Hardcover, **$13.95**
1980, 230 pages, 6 x 9

BASIC COMPUTER PROGRAMS FOR BUSINESS : VOL. 1

By Charles D. Sternberg.

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, **$12.95**
1980, 384 pages, 7 x 10

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, **$12.95**
1980, 204 pages, 6 x 9

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.

paper, **$11.95**
1979, 336 pages, 7 x 10

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.

paper, **$11.50**
1977, 176 pages, 6 x 9, illus.

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.

paper, **$11.95**
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.

paper, **$16.50**
1980, 200 pages, 8½ x 11

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, **$12.95**
1980, 384 pages, 7 x 10

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.

paper, **$16.95**
1981, 440 pages, 7 x 9

BEAT THE ODDS: MICROCOMPUTER 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.

paper, **$9.95**
1980, 128 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.

99'er MAGAZINE ON TAPE

QTY	ITEM	PRICE	TOTAL
	PACKAGE # E1 ($12/sub; $20/non-sub)		
	PACKAGE # U1 ($10/sub; $16/non-sub)		
	PACKAGE # G1 ($15/sub; $25/non-sub)		
	PACKAGE # G2X ($10/sub; $16/non-sub)		
	PACKAGE #M1/5 ($12/sub; $20/non-sub)		
	PACKAGE #M1/6 ($12/sub; $20/non-sub)		
SHIPPING – $1.00 for 1st TAPE, 50¢ ea. add. Foreign Orders Shipped Airmail – $3.00 for 1st TAPE, 75¢ ea. add.			
		TOTAL	

99'er magazine™ SUBSCRIPTION

□ YES—Please sign me up as a subscriber. Enclosed is my payment or credit card billing information.

Term	U.S.A.	Canada & Mexico	Foreign Surface	Foreign Air
1-yr (6 issues)	□ $18	□ $22	□ $28	□ $43
2-yr (12 issues)	□ $32	□ $42		
3-yr (18 issues)	□ $45	□ $60		

99'er BOOKSTORE

QTY	TITLE	PRICE	TOTAL AMOUNT

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.

POSTAGE ____

TOTAL ____

BACK ISSUES of *99'er Magazine* AVAILABLE

WHILE SUPPLIES LAST $3.95 ea. (postpaid in USA)

qty.	vol.	no.	qty.	vol.	no.	qty.	vol.	no.

TOTAL ____

Please Use Separate Renewal Card For Subscription Renewals

99'er MAGAZINE SUBSCRIPTION	
TOTAL 99'er Magazine ON TAPE	
TOTAL 99'er BOOKSTORE	
TOTAL 99'er-ware	
TOTAL	

99'er-ware™

*U.S. ONLY—FOR SHIPMENT OUTSIDE THE U.S. INQUIRE SEPARATELY

QTY	ITEM	PRICE	TOTAL AMOUNT
	99'er FINDER-BINDER	$10.95	
	SHIPPING – $3.00 ea.*		
	6–DIGITAL COMPUTER CASSETTES	$7.00	
	SHIPPING – $2.00 ea.*		
	SPECIAL: 99'er Finder-Binder • with 6 of the above cassettes for only $16.95 • with 12 of the above cassettes for only $21.95		
	SHIPPING – $4.00 ea.*		
	DUST COVERS: See prices on page 92 Indicate choices below.		
	SHIPPING–$2.00 for 1st COVER, 50¢ ea. add.*		
		TOTAL	

Address shown is □ Business □ Home

□ Check enclosed

MUST BE IN U.S. FUNDS DRAWN ON A U.S. BANK

NAME ______________________ PLEASE

ADDRESS ______________________ PRINT

CITY ______________ STATE ______ ZIP ______

Bill my:

□ VISA

□ Master Charge

Expiration Date ______________

Signature ______________

Account No. □□□□□□□□□□□□□□□□

MC Only — List 4 digits above your name □□□□

____

____

____

PLACE STAMP HERE

99'er TI Fest

P.O. Box 5537
Eugene, OR 97405

99'er magazine™

99'er TI fest

A Personal Computer Conference, Trade Show & Seminar

San Francisco
October 22-24, 1982
(See Reverse Side For Details)

A LEGEND IS BORN . . .

SEE & OPERATE the Latest Personal Computer Hardware & Software Products from TI & Third Parties!

SAVE Money by Purchasing the Products You Need at Special Show Discounts!

A UNIQUE OPPORTUNITY for Dealers, Manufacturers, & Product Developers to Display & Sell Directly to Thousands of Eager Buyers!

	Time	Room #1	Room #2	Room #3
Friday	9:30 – 10:45	Beginning Programming with TI BASIC (Part 1)	Assembly Language on the 99/4A System (Part 1)	Small Business Bookkeeping & Accounting Systems
	11:00 – 12:15	Beginning Programming with TI BASIC (Part 2)	Assembly Language on the 99/4A System (Part 2)	Applications for Hand-Held Computing
	1:30 – 2:45	Introduction to TI LOGO (Part 1)	Introduction to TI PILOT (Part 1)	The 99/4A p-System
	3:00 – 4:15	Introduction to TI LOGO (Part 2)	Introduction to TI PILOT (Part 2)	UCSD Pascal: An Overview of the Language
	4:30 – 5:45	CAI Software for Music Education	Scientific & Engineering Applications for Personal Computing	Third-Party Software Development
Saturday	9:30 – 10:45	Assembly Language on the 99/4A System (Part 1)	A Primer on Electronic Mail with the Home Computer	Applications Software for School Management
	11:00 – 12:15	Assembly Language on the 99/4A System (Part 2)	Using the Dow Jones Information Service with the Home Computer	Interactive Video Course Design
	1:30 – 2:45	Beginning Programming with TI BASIC (Part 1)	Financial Management with the Home Computer	Third-Party Software Development
	3:00 – 4:15	Beginning Programming with TI BASIC (Part 2)	The Design of Arcade-Style Games	The 99/4A p-System
	4:30 – 5:45	Small Business Bookkeeping & Accounting Systems	The Design of Adventure Games	UCSD Pascal: An Overview of the Language
Sunday	10:30 – 11:45	Introduction to TI PILOT (Part 1)	Introduction to TI LOGO (Part 1)	Sprite Operations in Extended BASIC
	12:00 – 1:15	Introduction to TI PILOT (Part 2)	Introduction to TI LOGO (Part 2)	Computer Graphics & Animation on the Home Computer
	1:45 – 3:00	Applications for Hand-Held Computing	Microcomputing & Electronic Learning Aids for the Handicapped	How To Start & Run a Users' Group
	3:00 – 5:00	"Birds of a Feather" Open Meetings	"Birds of a Feather" Open Meetings	"Birds of a Feather" Open Meetings

EXHIBITS, BARGAINS,
CONTESTS, PRIZES,
& FUN!
DAILY SEMINARS
AT NO EXTRA COST!

PLAY Spectacular Personal Computer Arcade Games . . . Enter the Competition to Win Computers, Peripherals, and Software!

MEET with Users' Groups and Share Ideas with Others of Similar Interest!

LEARN How to Profitably Produce Software and Hardware for this Dynamic, New Mass Market!

• • • the Legend of 99'er TI-Fest — —A LIVING Magazine℠

Hotel Information

The following list of hotels are offering special rates for 99'er TI-Fest attendees. Please reserve rooms as soon as possible since October is the busiest convention month in San Francisco, and rooms are often difficult to find.

	Single	Double	Comments
San Franciscan Hotel 1231 Market Street San Francisco, CA 94103 Tel. (800) 227-4747 outside CA *(415) 626-8000 CA call collect*	$59 – $65	$69 – $75	*Across the street from Civic Center. Prefers written reservations and room guarantees.*
Holiday Inn, Convention Center Eighth at Market Street San Francisco, CA 94103 Tel. (415) 626-6103 (800) 238-8000 Nationwide Reservations	$60	$75	*Across the street from Civic Center.*
Jack Tar Hotel Van Ness at Geary San Francisco, CA 94101 Tel. (415) 776-8200 (800) 527-2544 outside CA (800) 622-0855 in CA	$58 – $68	$68 – $78	*6 Blocks down Van Ness from Civic Center.*
Grosvenor Inn 1050 Van Ness Ave. San Francisco, CA 94109 Tel. (415) 673-4711	$59	$69	*6 Blocks down Van Ness from Civic Center. Rooms include Continental Breakfast and complimentary cocktails.*

℠ A service mark of Emerald Valley Publishing Co.

Brooks Hall at Civic Center
San Francisco

Friday Oct. 22, 1982 (9:00 AM – 6:00 PM)
Saturday Oct. 23, 1982 (9:00 AM – 6.00 PM)
Sunday Oct. 24, 1982 (10:00 AM – 5:00 PM)

Pre-Registration Information

1-Day Attendance Ticket	**$7.00**
2-Day Attendance Ticket	**$10.00**
3-Day Attendance Ticket	**$12.00**
Special Week-End (Sat. & Sun.) "Vid-Kid" Pass (must be under 12 years of age)	**$7.00**

❖ **FREE TI-Fest Poster (Suitable for Framing) To All Who Pre-Register Prior to October 8, 1982. Use Bind-In Pre-Registration Card Between Pages 88 & 89.**

PRE-SHOW & POST-SHOW MULTI-DAY WORKSHOPS

- **TMS9900 Assembly Language**
- **TI LOGO**
- **UCSD Pascal™**

Call or write for times, locations, & fees.

99'er Magazine
2715 Terrace View Drive
Eugene, OR 97405
Tel. (503) 485-8796

DEALER DIRECTORY

Los Angeles, CA

National Distributor of the TI-99/4A computer and related products offers the best opportunity to buy the computer or expand existing systems of continuing discount prices. Call evenings or write for prices. **Terry Glover 11746 Castillo Lane, Northridge, CA 91326, (213) 368-8204.**

Los Angeles, CA

TEX-COMP, a direct TI dealer supporting the 99/4 line at professional discounts. See our full page ad in this issue. Deal by mail for best service and prices. Send for free order kit and price-list. **TEX-COMP Users Supply Division, P.O. Box 33084, Granada Hills, CA 91344, (213) 781-6029, 873-4750.**

Denver, CO

Your headquarters for Texas Instruments TI-99/4A Personal Computers, software and peripherals is CoxCo Personal Computer Center. We maintain a complete stock of Command Modules and related equipment for TI and Atari Personal Computers. **Coxco Personal Computer Center, 12351 West 64th Ave., Arvada, CO 80004, (303) 421-6361.**

New Fairfield, CT

We specialize in only Texas Instruments 99/4A computers, accessories, and software, all at discount prices. Call or write for our price list to: **Home Computer-Ware, RR-3, Box 627, New Fairfield, CT 06810, (203) 746-2161.**

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. Airline Highway, Waterloo, IA 50701, (319) 232-6225.**

Call a Dealer Today

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, (800) 354-9099.**

Louisville, KY

Serving the Kentuckiana area with the full line of TI-99/4A computers, accessories and software including custom business applications. All TI calculators, DS990 minicomputer series and the new Business System 200 series. **Corporate Computer Services, Inc., Suite 455, 10170 Linn Station Road, Louisville, KY 40223, (502) 423-9187.**

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.**

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., 7008 Northland Drive, Brooklyn Park, MN 55428, (612) 536-6000.**

St. Louis, MO

We serve the St. Louis area with a complete line of TI-99/4A hardware and software. Specialists in educational applications of the 99/4A. Classes in BASIC and TI LOGO. **Ashford Computer Center, 11714 Manchester Rd., St. Louis, MO 63131, (314) 821-5666.**

Armonk, 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, 34 Maple Avenue, Armonk, NY 10504, (914) 273-6480.**

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. Epson Printers. **BLOISE Electronics, 1018 West State Street, Olean, NY 14760. (716) 372-1493.**

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**

Tacoma, WA

Specializing in Texas Instruments Computers all hardware and software. Also specializing in Satellite Earth Station, Video Recorders, and TV, will install Earth Stations anywhere in the United States, very low prices we want your business. **VIDSAT, 11002 Pacific Ave., Tacoma, WA 98444, (206) 531-9131 or 535-6141.**

Appleton, WI

Complete TI-99/4 hardware and software support. We deal by mail. Send SASE for complete price list. Custom Software written. Weekends call our B.B.S. at (414) 731-4957. **The Micro House, 1406 E. Wisconsin Avenue, Appleton, WI 54911 (414) 731-7183.**

Washington, D. C.

Marinchip systems 9900 and Z800 dealer. Networking software and CPM interface. Technico experience in-house. Epson printers, televideo CRT's, cables, TI 990/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.**

Mississauga, Ontario CANADA

Phoenix Software in Canada now offers the complete TI-99/4(A) product line at less than list prices. Complete price list available upon request. **Phoenix Software, 159-3175 Kirwin Avenue, Mississauga, Ontario, Canada L5A 3M4, (416) 275-6259.**

RATES

Listings here are $35 per issue; minimum insertion, 3 issues (6 months). Prepayment of $105 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.

Keep Your Magazines & Tapes Together With a 99'er FINDER-BINDER™

- Holds 6 Magazines and 6 Tapes
- Magazines Insert With Wire Straps, So That No Hole Punching Is Necessary
- Attractive and Durable

—Only $10.95*

*Only $10.95 without cassettes, plus $3.00 shipping & handling.
6 High-Quality 99'er-ware C-10 Digital Computer Cassettes
(with special BASF tape and 5-screw housing for data integrity)
Available separately for $7.00 plus $2.00 shipping & handling.

☆SPECIAL: 99'er Finder-Binder that is packed:
- with 6 of the above cassettes for only $16.95
- with 12 of the above cassettes for only $21.95

Add $4.00 shipping & handling to either order.

ALL 99'er-ware PRODUCTS ON THIS PAGE MAY BE ORDERED USING THE BIND-IN CARDS NEAR THE FRONT & REAR OF THIS MAGAZINE

The TI-SETTE™ Adapter

Cassette Compatibility At Last!

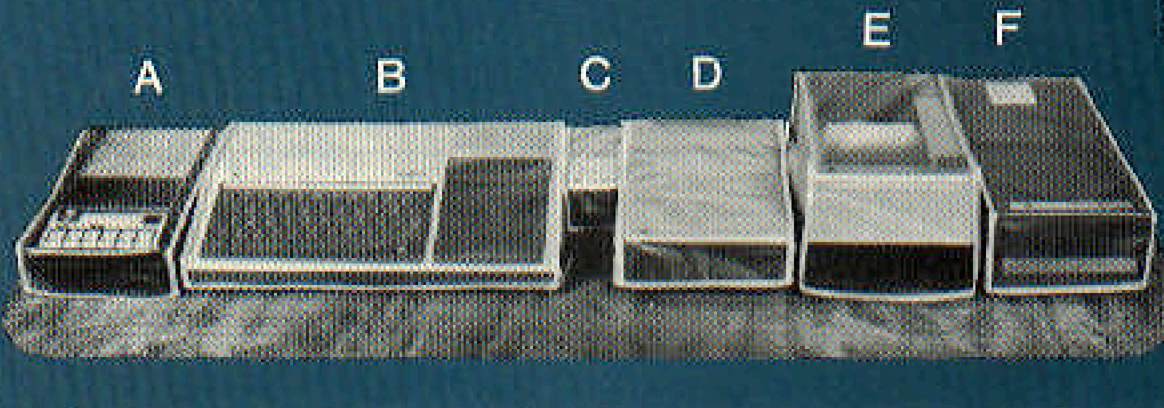
If The TI-99/4A Will Not Control Your Cassette Recorder Through Its Remote Jack, We Have The Solution For You . . .

- Low cost – Only **$4.50**

plus 75 cents each for postage & handling.

★ NEW ★ Peripheral Expansion Box Dustcover $12.95

DUST COVERS



Features:
- Equipment Protection
- Custom–Fit
- Handsome Appearance
- Antistatic Treated
- Quality Construction

A	Cassette Recorder Cover (1 size fits up to 10"x 6")	$4.95
B	TI-99/4(A) Console Cover	$8.95
C	Speech Synthesizer Cover	$3.95
D	Peripheral Box Cover (Specify: 32K Memory Expansion, RS232 Interface, or Disk Controller)	$5.95
E	Thermal Printer Cover	$8.95
F	Disk Memory Drive Cover	$5.95
	—Covers Not Shown in Photo:—	
	10" Color Monitor Cover	$10.95
	13" Color Monitor Cover	$12.95
	Epson MX-80 Cover (same as TI's 99/4 Matrix Printer)	$9.95

DEALER INQUIRIES INVITED

Add $2.00 shipping/handling for the first dustcover; 50 cents for each additional cover.

99'er-ware™
INNOVATIVE PRODUCTS FOR TMS9900-BASED PERSONAL COMPUTING

P.O. Box 5537
Eugene, Oregon 97405
Tel. (503) 485-8796

99'er magazine™ ON TAPE

Selected Programs from 99'er magazine™ Are Now Available on Cassette Tape

- All purchasers of these package are responsible for obtaining the individual documentation and program instructions in the indicated back issues of the magazines. When an issue goes out-of-print, copies of the program articles will be made available for purchase.
- If programs have been updated/enhanced since original publication, a printed copy of documentation changes will be provided with the tapes.
- Both subscriber & non-subscriber prices are indicated. To be eligible for the lower subscriber prices, you must be a current, paid magazine subscriber. All orders will be verified prior to shipment.
- Authors of these and future programs distributed on tape will receive Bonus Payments based on the number of tapes sold; so in fairness to them, please observe the Copyright laws, and report any incidents of "piracy" to our office.
- Recognized TI User Groups should contact us for special bulk terms & rates. No dealers, please.

Use the bind-in card in the back of the magazine for your convenience in ordering.

PACKAGE #M1/6
From Vol. 1, No. 6

Verbose (TI BASIC with Speech)
Color Mapping (TI BASIC)
County Fair Derby (TI BASIC)
Battle Star (Extended BASIC)
N-VADER (Extended BASIC)
Pre-School Block Letters (TI BASIC)

$12/sub; $20/non-sub

This package available for shipment after September 1, 1982

PACKAGE #G1
TI BASIC

Space War (Vol. 1, No. 3)
Battle at Sea (Vol. 1, No. 2)
Maze Race (Vol. 1, No. 4)
Anti-Aircraft Gun (Vol. 1, No. 1)
Invasion from Space (Vol. 1, No. 1)
Enemy Attack (Vol. 1, No. 1)

$15/sub; $25/non-sub

PACKAGE #M1/5
From Vol. 1, No. 5

Tex-Thello (TI BASIC)
San Francisco Tourist (TI BASIC)
Name That Bone (TI BASIC)
Space Patrol (Extended BASIC)
Force 1 (Extended BASIC)
Spriter (Extended BASIC)

$12/sub; $20/non-sub

PACKAGE #E1
TI BASIC

Typing for Accuracy (Vol 1, No. 3)
Let's Learn Notes (Vol 1, No. 3)
Mystery Words (Vol. 1, No. 4)
Homework Helper: Fractions (Vol. 1, No. 1)
Homework Helper: Division (Vol. 1, No. 4)

$12/sub; $20/non-sub

PACKAGE #U1
TI BASIC

Electronic Home Secretary (Vol. 1, No. 2)
Micro Bartender (Vol 1, No. 4)
Interactive Forms Generator (Vol. 1, No. 3)
Music Text Editor & File Player (Vol 1, No. 1)

$10/sub; $16/non-sub

PACKAGE #G2X
Extended BASIC

Interplanetary Rescue (Vol. 1, No. 4)
Dogfight (Vol. 1, No. 3)
Sprite Chase (Vol. 1, No. 2)
Dodge'em (Vol. 1, No. 4)

$10/sub; $16/non-sub

SHIPPING — $1.00 for 1st TAPE, 50¢ ea. add. Foreign Orders Shipped Airmail — $3.00 for 1st TAPE, 75¢ ea. add.

Back Issues of 99'er 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.

ISSUE #3 (Partial Contents):
•Interactive Forms Generator •Business Software Reviews •A BASIC Program that Writes BASIC Programs •Applesoft to TI BASIC •Games of Agility & Derring—Do •Pascal Development System •Personal Record Keeping •Adding Your Own Disk Drives •Preliminary Look at TI's Editor/Assembler •Memory & I/O plus EPROM Programmer for University Board •State of the Art in Computer Assisted Instruction •Precautions & Programming Ideas for Beginners •and much, much more.

ISSUE # 4 (Partial Contents):
•The Joy of Computer Gaming •Designing Your Own Computer Game •Bit-Plot Printer Graphics •A BASIC Program That Writes BASIC Programs •Micro Bartender •Fundamentals of Assembly Language Programming •Games of Time and Direction •Computer Inventory •A Joystick Video Game in TI LOGO •Homework Helper: Division •Computer Techniques for Tutoring the Mentally Handicapped •Computer Games to Learn Music •New LOGO Primitives and the DYNATURTLE •The Secret of Personal Record Keeping •Tape Recorder Survey •Around the Users Groups •New Products •Programming Tips •and much, much more.

ISSUE # 5 (Partial Contents):
•Calculating Loan Payoffs with Rule of 78 •Overview of Mini Memory Module •How to Buy a Computer •Screen-Dump Design •Name That Bone •Avoiding Turtle Traps in LOGO •Beginners' Tips •Game Designer Interview •Using Random Access Files •High Speed Animation with Sprites •Computer Chess •Space Arcade Games •The Tex-Thello Computer Challenge •Pig Latin & Math Puzzles in TI LOGO •Music & CAI with Gifted Children •Learning to Type with a "Pocket Program" Drill •Development of Microcomputer-Based Management Applications •3rd-Party News •and much, much more.

Index to Advertisers

Crayon . . . from p. 85

```
MARK1  SLA  R0,4              GET RID OF MOST SIG NYBBLE
       SRL  R0,4              PUT BACK REMAINING NYBBLE
       SWPB R0                MAKE IT LEAST SIG BYTE
MARK2  A    R1,R0             ADD NEW COLOR TO ADJACENT VALUE
       SWPB R0                MAKE IT MOST SIG BYTE
       MOVB R0,@PATRN(2)      MOVE IT TO ARRAY AT OFFSET
       LI   R0,>0800          VDP PATTERN TABLE ADDRESS
       LI   R1,PATRN  °       CPU BUFFER
       LI   R2,>600           1536 BYTES TO MOVE
       BLWP @VMBW             WRITE TO REDRAW SCREEN
SKIP   CLR  R5                CLEAR R5 AND R6 TO RECEIVE JOYST VALUES
       CLR  R6
       MOVB @JOYSTY,R5        PUT Y RETURN IN R5
       NEG  R5                MULTIPLY BY -1
       SLA  R5,2              MULTIPLY BY 4
       MOVB @JOYSTX,R6        PUT X RETURN IN R6
       SLA  R6,2              MULTIPLY TIMES 4
       SWPB R6                MAKE XVEL LEAST SIG BYTE
       MOVB R5,R6             MOVE YVEL TO R6 AS MOST SIG BYTE
       LI   R1,USRWS+12       CPU ADDRESS OF VELOCITY BYTES (R6)
       LI   R0,>0780          VDP ADDRESS OF MOTION TABLE
       LI   R2,2              2 BYTES TO MOVE
       BLWP @VMBW             WRITE DATA TO VDP RAM
       B    @CHECK             START LOOP OVER AGAIN
*
* ----- END OF MAIN PROGRAM LOOP ---------------------
*
* COLOR SELECT ROUTINE
*
SELECT LI   R0,>07EE          CHANGE BACKGROUND TO GRAY
       BLWP @VWTR             WRITE TO VDP R7
       LI   R0,>800           VDP BUFFER FOR PATTERN TABLE
       LI   R1,PALET          CPU BUFFER FOR PALETTE
       LI   R2,>600           1536 BYTES TO MOVE
       BLWP @VMBW             DISPLAY PALETTE
       LI   R0,>300           VDP BUFFER FOR ATTRIBUTE LIST
       LI   R1,ARRATT         ARROW ATTRIBUTES
       LI   R2,4              4 BYTES TO MOVE
       BLWP @VMBW             WRITE DATA
       BL   @DEBNC            BRANCH TO "DEBOUNCE" SUBROUTINE
LOOP9  LIMI 2                 ENABLE VDP INTERRUPT
       LIMI 0                 DISABLE INTERRUPT
       MOVB @ONE,@UNIT        IDENTIFY REMOTE UNIT TO SCAN
       BLWP @KSCAN            SCAN LEFT KBD AND REMOTE UNIT #1
       CB   @FIRE,@H18        CHECK FIRE BUTTON
       JEQ  CMARK             IF PRESSED, CHANGE MARK COLOR
       CB   @FIRE,@H14        CHECK "C" KEY
       JEQ  CSCRN             IF PRESSED, CHANGE SCREEN COLOR
       CLR  R6                INITIALIZE R6
       MOVB @JOYSTX,R6        PUT JOYST X IN R6
       SLA  R6,2              MPY BY 4
       SWPB R6                MAKE LEAST SIG BYTE
       LI   R1,USRWS+12       LOAD CPU ADDRESS (R6)
       LI   R0,>07B0          LOAD ADDRESS OF MOTION TABLE
       LI   R2,2              MOVE 2 BYTES
       BLWP @VMBW             LOAD DATA TO VDP RAM
       JMP  LOOP9             GOTO LOOP9
CSCRN  BL   @DOTCOL           DETERMINE COLOR FROM DOT COLUMN OF ARROW
       SWPB R1                MAKE IT MOST SIG BYTE
       MOVB R1,@BCOLOR        MOVE IT TO BCOLOR
       JMP  BACK              JUMP TO BACK
CMARK  BL   @DOTCOL           DETERMINE COLOR FROM DOT COLUMN OF ARROW
       SLA  R1,12             PUT IN PROPER POSITION FOR @FCOLOR
       MOVB R1,@FCOLOR        MOVE IT TO FCOLOR
BACK   BL   @DEBNC            DEBOUNCE
       CLR  R0                PREPARE TO RETURN SCREEN COLOR
       MOVB @BCOLOR,R0        PUT BACKGROUND COLOR IN R0
       SWPB R0                MAKE IT LEAST SIG BYTE
       MOVB @H07,R0           INDICATE WRITE TO VDP R7
       BLWP @VWTR             WRITE IT TO R7
       LI   R0,>B00           VDP PATTERN TABLE ADDRESS
       LI   R1,PATRN          PATTERN BUFFER IN CPU RAM
       LI   R2,>600           1536 BYTES TO WRITE
       BLWP @VMBW             LOAD PATTERN SCREEN
       LI   R0,>300           VDP SPRITE ATTRIBUTE TABLE ADDRESS
       LI   R1,ATTRIB         ADDRESS OF CURSOR ATTRIBUTES
       LI   R2,4              4 BYTES TO MOVE
       BLWP @VMBW             LOAD DATA TO GET CURSOR SPRITE
       B    @SKIP             BRANCH TO LABLE SKIP
*
* DSR ROUTINE TO SAVE "SCREEN" -- PATTERN TABLE
*
SAVE   LI   R0,>1000          PREPARE TO MOVE PATRN TO VDP BUFFER
       LI   R1,PATRN          CPU BUFFER ADDRESS
       LI   R2,>600           1536 BYTES TO MOVE
       BLWP @VMBW             WRITE DATA
       LI   R0,PAB            VDP PERIPHERAL ACCESS BLOCK ADDRESS
       LI   R1,PDATA          CPU BUFFER TO BE WRITTEN TO VDP
       LI   R2,21             21 BYTES TO WRITE
       BLWP @VMBW             WRITE PAB
       LI   R6,PAB+9          SET POINTER TO NAME LENGTH
       MOV  R6,@PNTR          STORE IN >8356 >8357
       BLWP @DSRLNK           EXECUTE SAVE OR LOAD
       DATA 8
       B    @CHECK            IF SO, BRANCH BACK TO BEGINNING
*
* DSR ROUTINE TO RECALL "SCREEN" -- PATTERN TABLE
*
RECALL LI   R0,PAB            VDP PERIPHERAL ACCESS BLOCK ADDRESS
       LI   R1,PDATA          CPU BUFFER TO WRITE
       LI   R2,21             21 BYTES TO WRITE
       BLWP @VMBW             WRITE PAB
       LI   R0,PAB            SUBSTITUTE "LOAD" I/O OP CODE
       MOVB @LOAD,R1          MOVE OP CODE TO R1
       BLWP @VSBW             WRITE BYTE TO PAB
       LI   R6,PAB+9          SET POINTER TO NAME LENGTH
       MOV  R6,@PNTR          STORE IN >8356 >8357
       BLWP @DSRLNK           COPY DATA TO VDP BUFFER
       DATA 8
       LI   R0,>1000          PREPARE TO COPY FROM VDP TO PATRN
       LI   R1,PATRN          CPU BUFFER ADDRESS
       LI   R2,>600           1536 BYTES TO COPY
       BLWP @VMBR             COPY BUFFER
       LI   R0,>0800          NOW COPY TO PATTERN TABLE
       LI   R1,PATRN          ADDRESS OF CPU BUFFER
       LI   R2,>600           1536 BYTES TO COPY
       BLWP @VMBW             COPY TO TABLE
       B    @CHECK            BACK TO THE BEGINNING
*
* SUBROUTINE TO PERIODICALLY CHANGE SPRITE COLORS
*
CHECKS AI   R4,>100           ADD 256 TO R4
       JEQ  CHANGE            WHEN R4 REACHES 0, CHANGE COLOR
       DEC  R0                DEC COUNTER
       JNE  CHECKS            IF NOT 0 ADD ANOTHER 256
       JMP  RETURN            BACK TO MAIN PROGRAM
CHANGE SWPB R3                SWITCH COLOR BYTES IN R3
       MOV  R3,R1             PUT R3 IN R1
       LI   R0,>303           ADDRESS OF SPRITE #0 COLOR IN VDP RAM
       BLWP @VSBW             WRITE MOST SIG BYTE OF R1
RETURN RT                     BACK TO MAIN PROGRAM
*
* DEBOUNCE SUBROUTINE
*
DEBNC  MOVB @ONE,@UNIT        KEY UNIT TO CHECK
       BLWP @KSCAN            SCAN KEYBOARD
       CB   @FIRE,@NOKEY      IS NO KEY PRESSED?
       JNE  DEBNC             IF A KEY IS PRESSED, CHECK AGAIN.
       RT                     GO BACK TO MAIN PROGRAM
*
* SUBROUTINE TO DETERMINE COLOR FOR ARROW
*
DOTCOL CLR  R1                INITIALIZE R1 TO RECEIVE DOT COLUMN
       LI   R0,>301           VDP ADDRESS OF DOT COLUMN
       BLWP @VSBR             READ BYTE FROM ATTRIBUTE TABLE
       SWPB R1                MAKE IT LEAST SIG BYTE
       AI   R1,>07            ADD OFFSET FOR POINT OF ARROW
       SRL  R1,4              DIVIDE BY 16
       RT                     RETURN
*
* "END START"
*
AUTO   END  MARKER            AUTOSTART
```

HUGE ELEK-TEK DISCOUNTS ON

TI-99/4A Home Computer System

TI-994A Keyboard $300.00

TEXAS INSTRUMENTS
INCORPORATED

Model	Name	Mfr. Sugg. Ret.	Elek-Tek Price
CONSOLE			
PHC 004A	TI-99/4A Home Computer (incl. RF Modulator)	525 00	300.00
PERIPHERALS			
PHP 1200	Peripheral Expansion Box	249 95	180.00
PHP 1220	RS-232 Card.	174 95	130 00
PHP 1240	Disk Controller Card (One Disk Manager module packed with each Disk Controller)	249 95	180.00
PHP 1250	Expansion System Disk Drive (Disk Drive Controller required)	399 95	285.00
PHP 1260	Memory Expansion Card (32K RAM)	299 95	215.00
PHP 1270	P-Code Card (32K RAM Memory Expansion required)	249 95	180.00
PHP 1500	Solid State Speech™ Synthesizer	149 95	110.00
PHP 1600	Telephone Coupler (Modem)	224 95	163.00
PHP 2500	TI 80 Column Impact Printer	750 00	500.00
PHP 2300	VCR Controller	699 95	500.00
PHP 2400	P-Code Peripheral	399 95	290.00
PHA 2100	R.F. Modulator (TV Adapter)	49.95	38.00
PHA 4100	10" Color Monitor	399 95	320.00
OPTIONAL ACCESSORIES			
PHP 1100	Wired Remote Controllers (Pair)	34 95	28.00
PHA 1950	Thermal Paper (2 Pack)	9 95	8.00
PHA 2000	Dual Cassette Cable	14 95	12.00
PHA 2010	Monitor Cable	19.95	15.00
PHA 2020	Audio Adapter (Headphone Jack)	19 95	15.00
PHA 2310	Panasonic VCR Controller Cables	99 95	85.00
PHA 2320	Sony Video Controller Cables	99 95	85.00
PHA 2330	Pioneer Video Controller Cables	99 95	85.00
APPLICATION PROGRAMS			
Home Management/Personal Finance			
	Command Modules		
PHM 3006	Home Financial Decisions	29 95	24.00
PHM 3007	Household Budget Management (Data storage system is required)	39 95	32.00
PHM 3012	Securities Analysis	54 95	44.00
PHM 3013	Personal Record Keeping (Data storage system is recommended)	49 95	40.00
PHM 3016	Tax/Investment Record Keeping (Disk system is recommended)	69 95	56.00
PHM 3022	Personal Real Estate (Data storage system is recommended)	69 95	56.00
PHM 3044	Personal Report Generator (Data storage system is recommended)	49 95	40.00
PHM 3111	TI Writer (32K Memory Expansion required)	99 95	75.00
PHM 3113	Microsoft™ Multiplan™††† (32K Memory Expansion required)	99 95	75.00
	Diskette		
PHD 5001	Mailing List	69 95	56.00
PHD 5003	Personal Financial Aids	19 95	16.00
PHD 5021	Checkbook Manager	19 95	16.00
PHD 5022	Business Aids Library—Finance Management (Extended BASIC Command Module is required)	39 95	32.00
PHD 5024	Business Aids Library—Inventory Management (Persogal Record Keeping or Statistics Command Module is required)	69 95	56.00
PHD 5027	Business Aids Library—Invoice Management (Personal Record Keeping or Statistics Command Module is required)	69 95	56.00
PHD 5029	Business Aids Library—Cash Management (Extended BASIC Command Module is required)	39 95	32.00
PHD 5038	Business Aids Library—Lease/Purchase Decisions	69 95	56.00
	Cassette		
PHT 6003	Personal Financial Aids	14 95	12.00
PHT 6038	Business Aids Library—Lease/Purchase Decisions	59 95	45.00
Educational/Personal Enrichment			
	Command Modules		
Texas Instruments Packages			
PHM 3002	Early Learning Fun	29 95	24.00
PHM 3003	Beginning Grammer	29 95	24.00
PHM 3004	Number Magic	19 95	16.00
PHM 3005	Video Graphs	19 95	16.00
PHM 3008	Video Chess	69 95	56.00
PHM 3010	Physical Fitness	29 95	24.00
PHM 3020	Music Maker (Data storage system is recommended)	39 95	32.00
PHM 3021	Weight Control and Nutrition (Data storage system is recommended)	59 95	48.00
PHM 3040	TI LOGO (Memory Expansion is required)	129 95	75.00
PHM 3064	Touch Typing Tutor* (Available for TI-99/4A only)	39 95	32.00
PHM 3109	TI Logo II* (32K Memory Expansion is required)	129 95	75.00
Scott, Foresman Reading and Math Packages (Developed by Scott, Foresman)			
PHM 3015	Early Reading (Solid State Speech™ Synthesizer is required)	54 95	44.00
PHM 3043	Reading Fun (Solid State Speech™ Synthesizer is recommended)	54 95	44.00
PHM 3046	Reading On	54 95	44.00
PHM 3047	Reading Roundup	54 95	44.00
PHM 3048	Reading Rally	54 95	44.00
PHM 3082	Reading Flight	54 95	44.00
PHM 3027	Addition and Subtraction I (Solid State Speech™ Synthesizer is recommended)	39 95	32.00
PHM 3028	Addition and Subtraction II (Solid State Speech™ Synthesizer is recommended)	39 95	32.00
PHM 3029	Multiplication I (Solid State Speech™ Synthesizer is recommended)	39 95	32.00
PHM 3049	Division I (Solid State Speech™ Synthesizer is recommended)	39 95	32.00
Scholastic Packages (Developed by Scholastic, Inc.)			
PHM 3059	Scholastic Spelling Level 3 (Solid State Speech™ Synthesizer is required)	59 95	48.00
PHM 3060	Scholastic Spelling—Level 4 (Solid State Speech™ Synthesizer is required)	59 95	48.00
PHM 3061	Scholastic Spelling—Level 5 (Solid State Speech™ Synthesizer is required)	59 95	48.00
PHM 3062	Scholastic Spelling—Level 6 (Solid State Speech™ Synthesizer is required)	59 95	48.00
Addision-Wesley Computer Math Games (Developed by Addison-Wesley Publishing Co.)			
PHM 3083	Computer Math Games II	39 95	32.00
PHM 3088	Computer Math Games VI	39 95	32.00
Milliken Home Math Series—K thru 8th grade (Developed by Milliken Publishing Co.)			
PHM 3090	Addition*	39 95	32.00
PHM 3091	Subtraction*	39 95	32.00
PHM 3092	Multiplication*	39 95	32.00
PHM 3093	Division*	39 95	32.00
PHM 3094	Integers*	39 95	32.00
PHM 3095	Fractions*	39 95	32.00
PHM 3096	Decimals*	39 95	32.00
PHM 3097	Percents*	39 95	32.00
	Diskette		
Texas Instruments Packages			
PHD 5007	Teach Yourself BASIC	34 95	28.00
PHD 5009	Music Skills Trainer	29 95	24.00
PHD 5011	Computer Music Box	19 95	16 00
PHD 5018	Market Simulation	19 95	16.00
PHD 5019	Teach Yourself Extended BASIC (Extended BASIC Command Module is required)	24 95	20 00
PHD 5020	Music Maker Demonstration (Music Maker Command Module is required)	14 95	12.00
PHD 5023	Basketball Statistics (Extended BASIC Command Module is required)	24 95	20.00
PHD 5026	Bridge Bidding I	29 95	24.00
PHD 5030	Speak & Spell™ Program (Solid State Speech™ Synthesizer is required)	29 95	24.00
PHD 5031	Speak & Math™ Program (Solid State Speech™ Synthesizer and Terminal Emulator II are required)	29 95	24.00
PHD 5039	Bridge Bidding II	29 95	24.00
PHD 5041	Bridge Bidding III	29 95	24.00
PHD 5042	Speller Writer (Terminal Emulator II Command Module and Solid State Speech™ Synthesizer are required)	29 95	24.00
PHD 5067	Beginner's BASIC/Tutor	29 95	24.00
MECC Packages (Developed by Minnesota Educational Computing Consortium)			
PHD 5078	Metric and Counting* (Extended Basic Command Module is required)	29 95	24.00
PHD 5079	Elementary Economics*	29 95	24.00
PHD 5080	Elementary Math and Science* (Extended BASIC Command Module is required)	29 95	24.00
PHD 5081	Astronomy*(Extended BASIC Command Module is required)	29 95	24.00
PHD 5082	Word Beginnings* (Extended BASIC Command Module is required)	29 95	24.00
PHD 5083	Exploring* (Extended BASIC Command Module is required)	29 95	24.00
PHD 5084	Math Practice* (Extended BASIC Command Module is required)	29 95	24.00
PHD 5085	Science Facts* (Extended BASIC Command Module is required)	29 95	24.00

* Available in Fourth Quarter 1982
† Developed by Scott Foresman
†† Developed by Milton Bradley—The Attack, Blasto, Hustle, ZeroZap, Connect Four and Yahtzee are trademarks of Milton Bradley
††† Developed by Microsoft™, Inc. Multiplan™ is a trademark of Microsoft™, Inc.

Model	Name	Mfr. Sugg. Ret.	Elek-Tek Price
APPLICATION PROGRAMS CONTINUED			
Education/Personal Enrichment Continued			
PHD 5086	Natural Science* (Extended BASIC Command Module is required)	29 95	24.00
PHD 5087	Social Science* (Extended BASIC Command Module is required)	29 95	24.00
PHD 5088	Teacher's Tool Box* (Extended BASIC Command Module and printer are required)	29.95	24.00
	Cassette		
PHT 6007	Teach Yourself BASIC	29.95	24.00
PHT 6009	Music Skills Trainer	24 95	20.00
PHT 6011	Computer Music Box	14 95	12.00
PHT 6018	Market Simulation	14.95	12.00
PHT 6019	Teach Yourself Extended BASIC (Extended BASIC Command Module is required)	19 95	16.00
PHT 6026	Bridge Bidding I	24 95	20.00
PHT 6031	Speak & Math™ Program (Solid State Speech™ Synthesizer and Terminal Emulator II are required)	24 95	20.00
PHT 6039	Bridge Bidding II	24.95	20.00
PHT 6041	Bridge Bidding III	24 95	20.00
PHT 6042	Spell Writer (Terminal Emulator II Command Module and Solid State Speech™ Synthesizer are required)	24.95	20.00
PHT 6067	Beginner's BASIC Tutor	24.95	20.00
Entertainment			
	Command Modules		
Texas Instruments Packages			
PHM 3009	Football	29 95	24.00
PHM 3018	Video Games I	29.95	24.00
PHM 3023	Hunt the Wumpus	24 95	20.00
PHM 3024	Indoor Soccer	29 95	24.00
PHM 3025	Mind Challengers	24 95	20.00
PHM 3030	A-Maze-Ing	24 95	20.00
PHM 3052	Tombstone City 21st Century	39 95	32.00
PHM 3053	TI Invaders	39 95	32.00
PHM 3054	Car Wars	39 95	32.00
PHM 3057	Munch Man	39 95	32.00
PHM 3042D	Tunnels of Doom (2 Diskette Games Included)	59 95	45.00
PHM 3042T	Tunnels of Doom (2 Cassette Games Included)	59 95	45.00
PHM 3056	Alpiner*	39 95	32.00
PHM 3110	Chisholm Trail	39 95	32.00
PHM 3112	Parsec*	39 95	32.00
Milton Bradley Packages (Developed by Milton Bradley Company)			
PHM 3031	The Attack††	39 95	32.00
PHM 3032	Blasto††	24 95	20.00
PHM 3033	Blackjack and Poker††	24 95	20.00
PHM 3034	Hustle††	24 95	20.00
PHM 3036	ZeroZap††	19 95	16.00
PHM 3037	Hangman††	19 95	16.00
PHM 3038	Connect Four††	19 95	16.00
PHM 3039	Yahtzee††	24.95	20.00
Adventure International Packages (Developed by Scott Adams)			
PHM 3041D	Adventure (Pirate Adventure Diskette Game Included)	49 95	40.00
PHM 3041T	Adventure (Pirate Adventure Cassette Game Included)	49 95	40.00
Gabriel Industries Packages (Developed by Gabriel Industries)			
PHM 3067	Othello† (Developed by Gabriel Industries)	39 95	32.00
	Diskette		
Texas Instruments Packages			
PHD 5002	TI-Trek (with optional speech)	14 95	12.00
PHD 5010	Mystery Melody	14 95	12.00
PHD 5015	Oldies But Goodies—Games I	19 95	16.00
PHD 5017	Oldies But Goodies Games II	24 95	20.00
PHD 5025	Saturday Night Bingo (Solid State Speech™ Synthesizer is required)	29 95	24.00
PHD 5037	Draw Poker (Extended BASIC Command Modules is required)	24 95	20.00
PHD 5057	Tombstone City: 21st Century (32K Memory Expansion and Extended BASIC Command Module are required)	19 95	16.00
PHD 5058	TI Invaders (32K Memory Expansion and Either Extended BASIC or Editor/Assembler Command Module are required)	19 95	16.00
PHD 5060	Munch Man (32K Memory Expansion and Either Extended BASIC or Editor/Assembler Command Module are required)	19 95	16.00
Adventure International Adventure Series (Developed by Scott Adams) (PHM 3041D required)			
PHM 5046	Adventureland	29 95	24.00
PHD 5047	Mission Impossible	29 95	24.00
PHD 5048	VooDoo Castle	29 95	24.00
PHD 5049	The Count	29 95	24.00
PHD 5050	Strange Odyssey	29 95	24.00
PHD 5051	Mystery Fun House	29 95	24.00
PHD 5052	Pyramid of Doom	29 95	24.00
PHD 5053	Ghost Town	29 95	24.00
PHD 5054	Savage Island I & II	39 95	32.00
PHD 5056	Golden Voyage	29 95	24.00
	Cassette		
Texas Instruments Packages			
PHT 6010	Mystery Melody	9 95	9.00
PHT 6015	Oldies But Goodies—Games I	14 95	12.00
PHT 6017	Oldies But Goodies Games II	19 95	16.00
PHT 6025	Saturday Night Bingo (Solid State Speech™ Synthesizer is required)	24 95	20.00
PHT 6037	Draw Poker	19 95	16.00
Adventure International Adventure Series (Developed by Scott Adams) (PHM 3041T required)			
PHT 6046	Adventureland	29 95	24.00
PHT 6047	Mission Impossible	29.95	24.00
PHT 6048	Voodoo Castle	29 95	24.00
PHT 6049	The Count	29 95	24.00
PHT 6050	Strange Odyssey	29.95	24.00
PHT 6051	Mystery Fun House	29 95	24.00
PHT 6052	Pyramid of Doom	29 95	24.00
PHT 6053	Ghost Town	29 95	24.00
PHT 6054	Savage Island I & II	39 95	32.00
PHT 6056	Golden Voyage	29 95	24.00
OTHER APPLICATION PROGRAMS			
	Command Modules		
PHM 3001	Demonstration	69 95	56.00
PHM 3011	Speech Editor (Solid State Speech™ Synthesizer is required)	44 95	36.00
PHM 3014	Statistics (Data storage system is recommended)	44 95	36.00
PHM 3026	Extended BASIC	99 95	75.00
PHM 3035	Terminal Emulator II	49 95	40.00
PHM 3055	Editor/Assembler	99 95	80.00
PHM 3058	Mini-Memory (4K)	99 95	80.00
PHM 3045D	SMU Electrical Engineering Library I (2 Diskettes included)	149 95	120.00
PHM 3045T	SMU Electrical Engineering Library I (10 Cassettes included)	149 95	120.00
	Diskette		
PHD 5004	Programming Aids I	14 95	12.00
PHD 5005	Programming Aids II	24 95	20.00
PHD 5006	Math Routine Library	29 95	24.00
PHD 5008	Electrical Engineering Library	29 95	24.00
PHD 5012	Programming Aids III	19 95	16.00
PHD 5013	Graphing Package	19 95	16.00
PHD 5016	Structural Engineering Library	29 95	24.00
PHD 5044	AC Circuit Analysis	29 95	24.00
PHD 5063	UCSD PASCAL™ Compiler (32K Memory Expansion and P-Code required)	124 95	100.00
PHD 5064	UCSD p-System™ Assembler/Linker (32K Memory Expansion and P-code required)	99 95	60 00
PHD 5065	UCSD p-System™ Editor/Filer/Utilities (32K Memory Expansion and P-code required)	74 95	60.00
PHD 5066	TI PILOT (32K Memory Expansion and P-code required)	79 95	60.00
PHD 5068	Course Designer Authoring Package (Extended BASIC required and Video Controller optional)	199 95	150.00
PHD 5075	Text-To-Speech (English) a (Solid State Speech™ Synthesizer, 32K Memory Expansion and Extended BASIC Command Module are required)	29 95	24.00
	Cassette		
PHT 6004	Programming Aids I	9 95	9.00
PHT 6006	Math Routine Library	24 95	20.00
PHT 6008	Electrical Engineering Library	24 95	20.00
PHT 6013	Graphing Package	14 95	12.00
PHT 6016	Structural Engineering Library	24 95	20.00
PHT 6044	AC Circuit Analysis	24 95	20.00
SOFTWARE LIBRARIES			
PHL 7001	The Home Financial Manager	139 95	100.00
PHL 7002	The Family Entertainer	89 95	65.00
PHL 7003	The Elementary Educator	99 95	75.00
PHL 7004	The Music Educator	64 90	50.00
PHL 7005	The Super Programmer	119 00	90.00
PHL 7006	The Speaking Math Teacher	119 85	90.00
PHL 7007	The Speaking Reading Teacher	109 90	85.00
PHL 7008	The Speaking Scholastic Spelling Teacher	219 80	160.00
PHL 7009	The TI Arcade Game Series	119 85	90.00
PHL 7010	The Milton Bradley Game Series	114 75	90.00
PHL 7011	The Computer Introductory Package	119 85	90.00

* Available only until replaced by peripheral card
* UCSD, UCSD Pascal and UCSD p System are all trademarks of the Regents of the University of California
* Othello is a trademark of Gabriel Industries
* Course is designed to be used with Circuit Analysis I textbook

ELEK-TEK, inc.
6557 N. Lincoln Ave., Chicago, IL 60645
(800) 621-1269 (312) 677-7660

CALL TOLL FREE 800-621-1269 (EXCEPT IL, AK, HI) MasterCard or Visa by mail or phone. Mail Cashier's Ck., Mon. Ord., Pers. Ck. (2 wks to clr). Add $4.00 1st item. **(AK, HI, P.R., Canada add $8.00 1st item)** $1.00 ea. add'l shpg. & handl. Shipments to IL address add 6% tax. Prices subject to change. **Write for free catalog. ALL ELEK-TEK MERCHANDISE IS BRAND NEW, FIRST QUALITY AND COMPLETE.**

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