For People Interested in the Enrichment of Personal Computing

Vol. 1, No. 5 $3.50 in U.S.A.

99'er magazine™

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

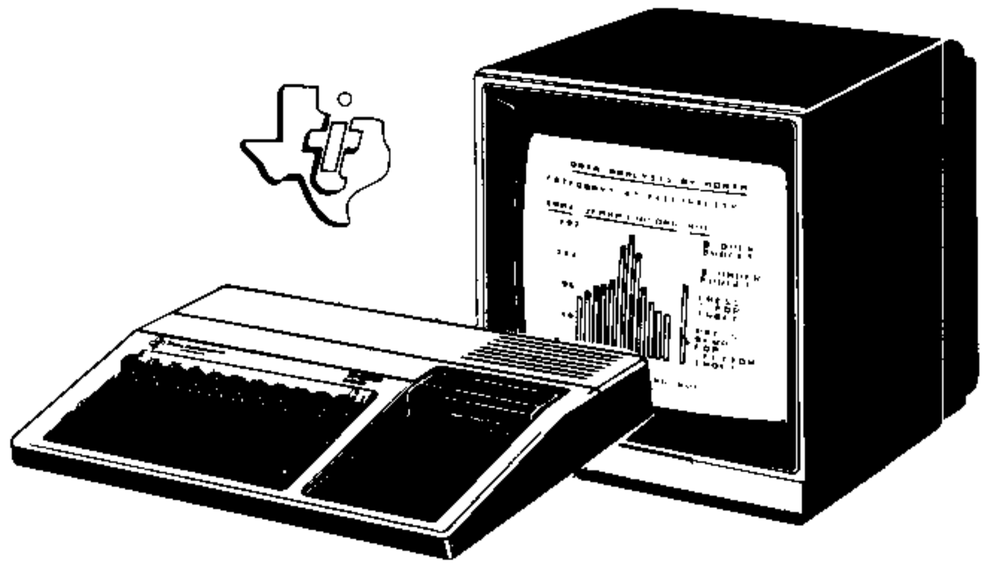
Also Featuring:

PLUS

Man and Personal Computer:
Partners in Decision Making

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** (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** (32K RAM)*	265.95
PHP	2300	VCR Controller?†	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™ Compiler**	99.95
PHD	5064	UCSD p-System™ Assembler Linker**	79.95
PHD	5065	UCSD p-System™ 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 Foresmen.
††Developed by Milton Bradley — The Attack, Blasto, Hustle, Zero Zap, Connect Four and Yahtzee are trademarks of Milton Bradley.
‡Available only until replaced by peripheral card
?Pending FCC approval, available Second Quarter 1982
™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

A dventures are

D isruptingly addictive,

V irtually thrusting

E gos into

N arcosis — beware!

T ime itself

U unravels while

R isking these

E nchantments by

S cott Adams.

#1 ADVENTURELAND — Wander through an enchanted realm and try to recover the 13 lost treasures. There are wild animals and magical beings to reckon with, as well as many other perils and mysteries. Can you rescue the Blue Ox from the quicksand? You'll never know until you try ADVENTURE #1! This is the Adams Classic which started the Whole Ball of Wax! Try it, you won't be sorry. Difficulty Level: Moderate

#2 PIRATE ADVENTURE — The lost treasures of Long John Silver lie hidden somewhere — will you be able to recover them? Only by exploring this strange island will you be able to uncover the clues necessary to lead you to your elusive goal! Difficulty Level: Easy

#3 MISSION IMPOSSIBLE ADVENTURE — In this exciting Adventure, time is of the essence as you race the clock to complete your mission in time — or else the world's first automated nuclear reactor is doomed! So, tread lightly and don't forget your bomb detector! If you survive this challenging mission, consider yourself a true Adventurer! Difficulty Level: Hard

#4 VOODOO CASTLE — The Count has fallen victim to a fiendish curse placed on him by his enemies. There he lies, with you his only possible hope. Will you pull off a rescue, or is he down for the Count for good? Difficulty Level: Moderate

#5 THE COUNT — It begins when you awake in a large brass bed in a castle somewhere in Transylvania. Who are you, what are you doing here and WHY did the postman deliver a bottle of blood? Who can say . . . but somewhere a centuries-old evil lies in dark wait . . . Difficulty Level: Moderate

#6 STRANGE ODYSSEY — At the galaxy's rim, there are rewards aplenty to be harvested from a long-dead alien civilization, including fabulous treasures and advanced technologies far beyond human ken! Will you be able to recover them and return home? Prepare yourself for the incredible! Difficulty Level: Moderate

#7 MYSTERY FUN HOUSE — This Adventure puts you into a mystery fun house and challenges you to find your way through and back out of it. Sure to baffle you for quite a while, the MYSTERY FUN HOUSE is patiently waiting for you to enter. So, step right up and get your tickets he-yah! Difficulty Level: Moderate

#8 PYRAMID OF DOOM — This is an Adventure that will transport you into a maddening dangerous land of crumbling ruins and trackless desert wastes — into the very PYRAMID OF DOOM! Jewels, gold — it's all here for the plundering — if you have the expertise to pull its recovery off! Difficulty Level: Moderate

#9 GHOST TOWN — You must explore a once thriving mining town in search of the 13 hidden treasures. With everything from rattlesnakes to runaway horses, it sure ain't going to be easy! And — they don't call them ghost towns for nothing, pardner! Includes a special bonus scoring system too! Difficulty Level: Hard

#10 SAVAGE ISLAND PART I — A small island in a remote ocean holds an awesome secret — will you be able to discover it? This is the beginning of a two-part Adventure, the second half concluding as SAVAGE ISLAND PART 2, ADVENTURE #11. NOTE: This one's a toughie — for experienced Adventures only! Difficulty Level: Harder

#11 SAVAGE ISLAND PART II — The suspense begun with Adventure #10 now comes to an explosive conclusion with SAVAGE ISLAND PART II! This Adventure requires you to have successfully finished #10 wherein you were given the password to begin this final half. The plot thickens as you wind your way through glowing corridors in search of the elusive clue that will enable you to solve the riddle of the island. NOTE: For experienced Adventurers only! Difficulty Level: Harder

#12 GOLDEN VOYAGE — The king lies near death in the royal palace. You have only three days to bring back the elixir needed to rejuvenate him. Journey through the lands of magic fountains, sacred temples, stormy seas and gold, gold, GOLD! Can you find the elixir in time? This one is for experienced Adventurers only! Difficulty Level: Hard

With the Adventure Command Module and one of these cassette or diskette based games developed by Adventure International you can experience many different adventures without leaving the comfort of your home. Each game challenges your powers of logical reasoning and may require hours, or even weeks, to complete. Pirate Adventure comes with the Command Module needed for all adventures and is $33.95. All other Adventures are $22.95 except Savage Island where both parts are sold as a unit for $28.95. All Adventures are available on either disk or cassette. Please specify which is wanted.

Master Charge, Visa, money order or any good payment. Add $2.98 shipping and handling any size order continental USA.

MORE AND BETTER SOFTWARE MAKES YOUR INVESTMENT A VALUE THAT GROWS AND GROWS.

Would you like a FREE copy of the A ACE BULLETIN? The current issue is a 20 page newspaper, tabloid size, just for the 99/4(A) computer. Call or write for yours today.

A ACE Computer

7 W. Airline Hwy.
Corner of Hwys. 57 & 63
Waterloo, Iowa 50701
Phone: 319/236-3861

THIS ISSUE'S COVER
Hayder Amir's cover painting depicts the forging of an ideal partnership in decision making—the merging of man's infinitely creative mind with the personal computer's prodigious calculating power and emotional detachment. The magazine is recursively positioned on both sides of the decision continuum, to graphically portray the cyclic nature of the decision process itself—completed decisions begetting still other YES-NO choices within the ever-changing spectrum of possibility.

Publisher / Editor
Gary M. Kaplan

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

Program Editor
Cheryl Whitelaw

Office Manager
Pat Kaplan

Circulation & Fulfillment
Rod Hoyle
Benjamin Kaplan
Janet Lovelace
Coleen Peplow

Advertising
Patana Ratanapreux
Tel. 503-485-8796

Contributing Editors
F. T. Berkey
Norma Clulow
Henry Gorman, Jr.
Roger Kirchner
Ira McComic
Mark Moseley
James Muller
Samuel Pincus
Regena
George Struble
Dennis Thurlow
Jerry Wolfe

Production Manager
Norman Winney, Jr.

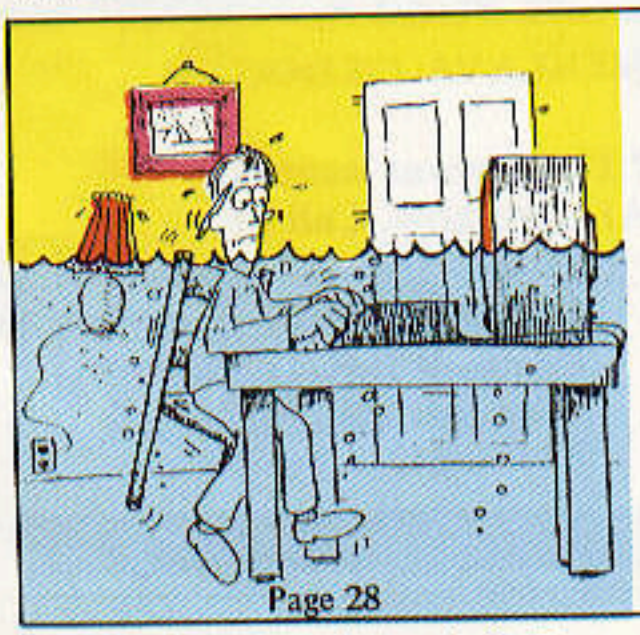
Production & Design
Greg Davis
June Gaber
Corby Poticha
Alice Sundstrom

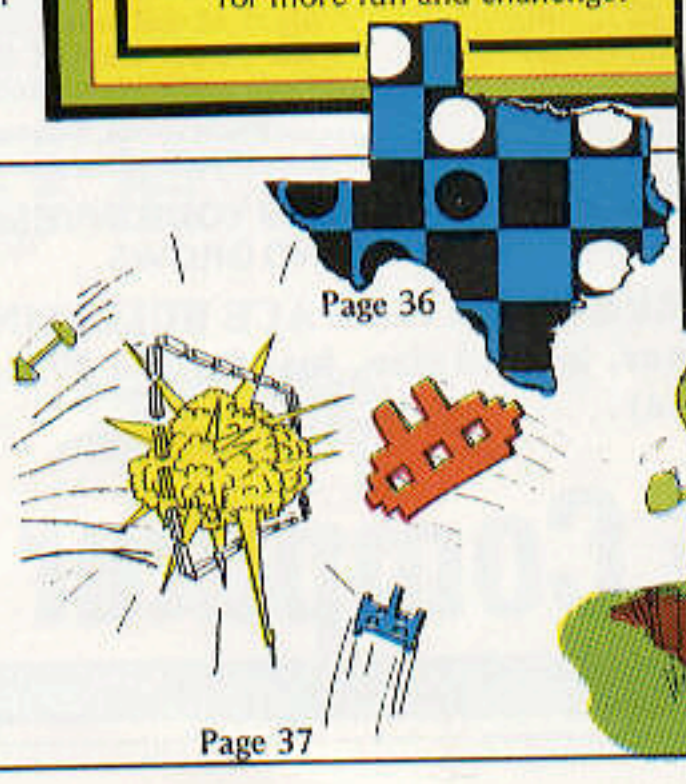
99'er magazine™

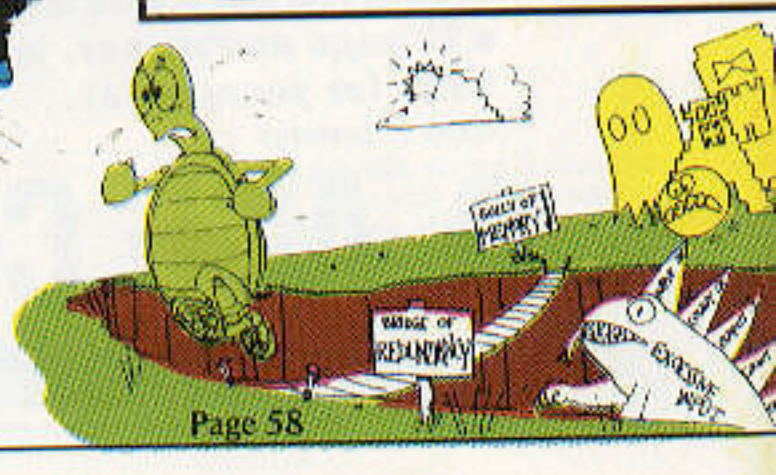
Programming Conventions

= Program as listed will completely available memory of TI-99/4A cannot be RUN with disk contro (and possibly RS232 Interface) tur on. It must be SAVEd and R from cassette.

= End of Program or Article


Page 28


Page 36
Page 37


Page 58

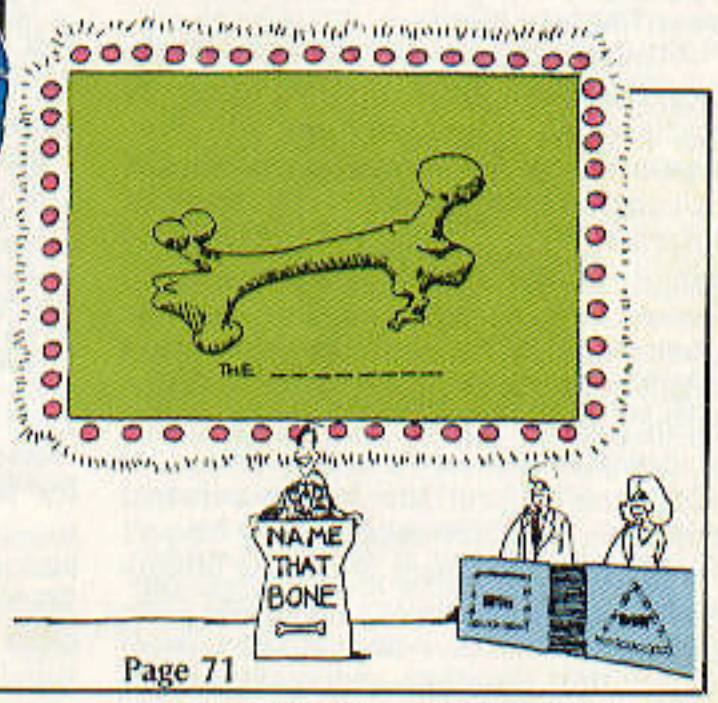
A Resource for People Interested in the Enrichment of Personal Computing

Volume 1 No. 5

99'ER VERSION 1.5.1XBM
- volume no.
- issue no.
- version
  - 1 = original program
  - 2 ... n = no. of update
- TI Extended BASIC
- Expansion Memory Required

Page 37

Page 39
Page 60



Page 71

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.

ON SCREEN

By Gary M. Kaplan
Editor & Publisher

Decisions, decisions . . . Life in the last two decades of the twentieth century seems destined to be alternately plagued and blessed by an ever-increasing number of important decisions—questions that need answering, options that need exploring, and choices that need choosing. It sure won't be an easy time for the human brain: Neurons and synapses will receive quite a workout under this flood of yes and no responses that we are required to make. Wouldn't it be great if we had a partner to help us perform some of the operations in information processing that our brains don't particularly do well—things like keeping track of the dozens of interrelated variables often necessary in evaluating many parallel options, or performing many thousands or millions of repetitive error-free calculations in "real time" minutes or hours, instead of taking years and decades of human labor.

Well, as many of you have known for quite some time, this ideal partner does indeed exist. And as the issue's cover art reveals, our helpful partner is none other than a personal computer.

What better example is there to illustrate the types of decisions we all seem to be faced with than an example involving *money*. If there is an area of human thought that is responsible for generating more decisions than the making or saving of money does, I'd be very surprised. In our leadoff article, *The Rule of 78*, you'll learn how to let the computer calculate loan payoff amounts so that you can decide the most prudent course of action in your financial affairs.

And just so you don't think that help in decision-making is confined to money matters only, *Overland Flow* will demonstrate how the computer's powers of calculation and its ability to represent information graphically is extremely useful in the design process as well. These two examples should provide plenty of ideas for applying the versatile talents of your dependable "calculating" partner.

Some of our readers, of course, cannot benefit from the help a personal computer provides because they haven't as yet decided (there are those ubiquitous "decisions" again . . .) *which* computer to buy, or *where* to buy it. If that's the situation you're in, or if you're trying to influence someone else's purchase decision, be sure to read *How to Buy a Personal Computer*, our "Square One" feature in this issue.

Those of you who already have a computer require information on which to base your purchase decisions for other add-on products. In this issue we take an in-depth look at a significant new product (that will be available this spring) in our "eye-opening" article, *TI's New Mini Memory Module*.

With all the great graphic screens you'll soon be creating or seeing in games, LOGO procedures, or CAI applications, you'll undoubtedly appreciate having a "hard copy" of what appears on screen. You can now learn how to implement *A Screen Printing Utility* (otherwise known as a "screen dump") in our regular "Super Language" assembler column.

But whether you are running programs in Assembly Language, TI BASIC, or Extended BASIC, *files* play an important part in many applications. And it's really *random-access* files that bring out the true power of a computer. Learn how to use them with your disk system in this issue's *Getting Down to Business* feature. For those of you in need of how-to information for using cassette tape files, watch forthcoming issues for a new tutorial series.

Our *Pages From OnLoCAltion* this time contain an interesting mix of material: First, for those of you who want to take advantage of computer-assisted instruction (CAI) but don't have the time to type in long programs, our new column of "pocket programs" should be just the thing. We think you'll heartily agree that *Pocket Typing Tutor* is quite a useful and sophisticated program for its tiny size.

On the other extreme, for those of you with the time, patience, and accuracy to type in a lengthy program, *Name That Bone* offers an excellent graphic example of colorful memory-drill programming techniques that can be adapted to most subjects.

But programs aren't the only things you'll find on these *Pages* . . . We return on location to Ohio once again for Part 2 of *Notes on a Computer Score* where you'll see how a class of gifted children reacted to having a computer available for programming their own music. And for all the schools out there who are looking for good management software, *The Scott, Foresman School Management Applications Development Process* should be illuminating. Any third-party software developer interested in using the medium of Command Modules would also do well to read this article.

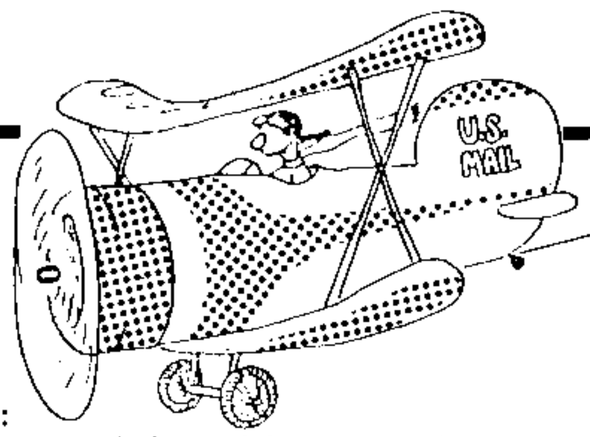
LOGO Times, our new magazine-within-a magazine launched last issue, is back again with a wealth of information and ideas to get you started in this exciting learning environment. *Avoiding Turtle Traps* offers some timely advice on writing better LOGO procedures. Then you can find out what Pig Latin and logical puzzles have in common when you explore *Problem Solving with LOGO*. Incidentally, this article sets up the framework for next issue's fantastic Tower of Hanoi game. And if you've ever asked yourself how hard it is to get started setting up the computer as a LOGO system, and learning enough to help get a group of children started, our final article entitled *TI LOGO with a Little T.L.C.* should be reassuring and help you decide to take the plunge . . .

This particular issue's big news is the brand new magazine-within-a-magazine that we announced last time. *Computer Gaming* has been officially born. Keep in mind that it's just an infant: we need plenty of material from readers to nurture it and help it grow to maturity. Though we have quite a journey ahead of us before *Computer Gaming* becomes the type of magazine I invision it eventually becoming, I think you'll agree that we're off to a good start in this premier issue.

First, there's the *Designer's Spotlight* interview with the designer and programmer of the super-fast TI arcade game, *Tombstone City: 21st Century*. Then in Part 2 of *Chuck-A-Luck* you'll learn how to use "top-down design" in actually writing a game program. For those of you who would like to dress up your games with some exciting animation, *Spriter* is a ready-to-use utility for just this purpose. And if you enjoy a good game of chess and are ready for some interesting variations, be sure to check out *Computer Chess Corner*.

No issue of *Computer Gaming* would be complete without some game reviews. We have them for you in our regular departments *Adventure Registry* and *Arcade Arbiter Reviews*. But lest you go hungry when late one night or perhaps one rainy Sunday afternoon you suddenly have a mad craving to play a new game, turn to *Gameware Buffet* for a selection of both BASIC and Extended BASIC fare—a virtual smorgasbord of fun and entertainment.

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



LETTERS TO THE EDITOR

Dear Sir:

I sure was glad to pick-up one of your magazines. This is just what I needed. I bought my TI-99/4A in November and am struggling to understand the basics about "BASIC!" Your magazine is a help as well as a great source for software games, books, and hardware. For me, the super novice, this information is invaluable! Keep up the good work!

Again thanks for the great articles (even the one's that hopefully I will be able to understand a couple of year's from now), the game programs for me to study and learn from, and for the software advertisements for us *"99'ers"*!

David Hurd
Lubbock, TX

Thanks for your comments, David. Watch for a lot more beginner-oriented articles in each issue so that you won't have to wait that "couple of years."

Dear Sir:

The article *How To Write A Basic Program That Writes Basic Programs* by John Clulow (*99'er* Vol. 1, No. 3; pg. 20), was most enlightening. I look forward to further articles in the series.

Three small "bugs" appear to have crept into the program "Condensed Record Structure" (Page 79). Lines 240, 250 and 380 should perhaps read:

```
240 LINENUM=BYTE1*256+BYTE2
250 IF LINENUM=65535 THEN 430
380 COL=COL+10 : : J=0
```

Thank you for such a stimulating and informative magazine. I hope you will be able to conclude your first year with the announcement that *99'er* is to become a monthly publication!

Bernie Elsner
Mount Lawley, Western Australia

Thank you Bernie for spotting this. Hope to see some of you 99'ers from "down under" make it to the 99'er TI-Fest where we are hoping to make that announcement.

Dear Sir:

In regards to your excellent article *The Secret of Personal Record Keeping* [Vol. 1 No. 4], you stated "Warning: If you write a program that continually loops to a CALL A statement, CLEAR cannot be used to break the program."

I have discovered that by using the FCTN 4 (CLEAR) *and* FCTN 5 (BEGIN) *simultaneously*, on the TI-99/4A, it will break into this loop. I hope you will pass this information on to your readers. Keep up the good work.

Jim Sheridan
Bolingbrook, IL

Dear Sir:

I believe it is important to caution new disk users not to place disks atop the drive. It's a natural shelf, but I know many who have lost not only their own programs that way, but purchased software as well. Disks must be handled carefully—they're only just barely non-volatile!

My 35 track drive yields a problem with the two Editor/Assembler disks, as they are 40 tracks and full. Thus, it is necessary to transfer to 3 disks, using a 40 track drive to read the original. I find Verbatim disks work perfectly. I tried Memorex, and the box I got was a disaster—seeingly incompatible.

R. M. Bies
Pittsburgh, PA

Dear Sir:

When my new Extended BASIC and Memory Expansion arrived, I anticipated keying in John Clulow's excellent adaptation of Charles Ehninger's idea of a "general purpose" load program (*99'er*, Vol. 1, No. 3, pg. 91).

Actually, since there are many places where I could go wrong, I wasn't too surprised when it didn't work the first time out. But I got somewhat annoyed when, several attempts later, I still could not get the program to work, even though I had been following Mr. Clulow's instructions religiously!

In trying to put my somewhat limited understanding of the 99/4 to use (a deficit which *99'er* has been instrumental in reducing), I at first guessed that perhaps the differences between the 99/4 and 4A were responsible (I have the 4A), but I discarded this idea when it became clear that the program works by placing values into the Memory Expansion unit, which uses a different form of memory than the console, and therefore the console model should make no difference.

I finally nailed the culprit in this tale when I more carefully read the small insert which comes with the newly released version of Extended BASIC. On page 11 of this booklet, and easily overlooked, is a note to the effect that there are now 24,488 "BYTES OF PROGRAM SPACE FREE" in the Memory Expansion, while the manual for this peripheral states that you "always" have "24,512 bytes of program space available." (Wrong, peripheral-box breath!)

Apparently, when a Memory Expansion is attached, Extended BASIC loads some of its nifty subroutines into it. The new version evidently uses somewhat more memory than the old. At this point it was clear that line 41 of the program (A=-991) contained the wrong memory address at which to load the program name you wish to run. Further, since 24,512–24,488 = 24, there are 24 less bytes with the new version, in effect pushing memory locations off by 24. Since the new address contains four digits instead of three, adding one more byte gives 991 + 24 + 1 = 1016. The solution to the problem then becomes:

```
41 A=-1016
```

When line 41 is changed as above, the program works perfectly, as long as the rest of the instructions are followed exactly; *99'er* readers who have received, or are planning to obtain, the new version of Extended BASIC will need to make the above change to run the "General Purpose Program Loader."

David Wakely
Oak Park, IL

Thank you, David. We should also note that A. Kludge's solution (Vol. 1, No. 4) works with either version.

Dear Sir:

In the July/August issue of the *99'er Magazine* Mr. Charles Ehninger expressed his frustration over the inabliity of Extended BASIC to allow the creation of a "General Purpose Program Loader." In an editorial comment, you "challenged" the readers to make Mr. Ehninger's idea a reality. The next two issues of *99'er* carried several reader's solutions to that challenge. However, each required that a 32K memory expansion unit be attached in order for their programs to function.

Now, I am a "poor" man who cannot yet afford the purchase price of a 32K memory expansion unit. And, because necessity is the mother of invention, I have developed, with the help of John Clulow's article *How To Write A Basic Program That Writes BASIC Programs: Part 2* (Vol. 1, No. 4), a POOR MAN'S PROGRAM LOADER which works in Extended BASIC *without* the need for the memory expansion. There is, however, one drawback to my POOR MAN'S LOADER: Unlike the "rich" man's version, more than one step is involved in implementing it.

The attached listing should be SAVEd and then RUN on each and every diskette containing programs capable of being run in Extended BASIC. I have given this program the file name LOADER. (If you wish to give it another name, you will have to make the appropriate change in Line 270.) LOADER should also be RUN whenever programs (not data files) are added to, or deleted from, a diskette. Doing so will cause LOADER to read the diskette's catalog and create, in a MERGE format file, a separate program that displays the names of all *programs* on the diskette, except LOADER, and will automatically RUN one at the touch of a button.

I have given this second program the file name CAT—short for catalog. (If you wish to give it another name, you will have to make the appropriate change in Line 290.) To enter CAT into your computer, simply type MERGE DSK1.CAT and then, when the cursor returns, RUN. (Extended BASIC does not respond to the singly-typed command MERGE DSK1.CAT : : RUN) The screen will clear, the name of the diskette will be printed, and a menu will be displayed showing the program contents of the diskette. In front of each program name will be a letter of the alphabet. Pressing one of them will automatically RUN the program listed next to it.

When MERGEing the CAT program, it is *not* necessary to erase an existing program already in memory unless it is extremely large or uses the array name(s) K@ and/or S@. The line numbering of CAT starts with 1 and increases in increments of one. Hence, upon MERGEing, CAT will either load in front of, or over-write, as necessary, any program lines already in memory.

Four final points:

1. *Always* use Disk Drive One (DSK1) when executing either LOADER or CAT. This is because LOADER will always store the CAT file to Disk Drive One and CAT will always "open" Disk Drive One in order to RUN the selected program.
2. With no memory expansion unit attached, CAT will not be able to RUN any program which is so large that it requires a CALL FILES(1) command to be issued *unless* CALL FILES(1) was executed prior to MERGEing CAT.
3. Once created, CAT is totally independent of LOADER and may be MERGEd and RUN at any time. The purpose of LOADER is to update CAT whenever changes are made in the program content of the diskette.
4. Apparently, a typographical error was made in Line 21 of the original Ehninger program; and this error was carried through on the subsequent modifications done by others. After the THEN statement, the line number should be 17—not 19. The purpose of Lines 17 and 18 are to count the number of records read from the diskette, to a maximum of 127, to insure that the program never tries to read past the end of the files. Thus, the counting variable I must be incremented whenever a record is read, even if "discarded" by Line 21.

In closing, I would like to echo the praises for the *99'er* that have appeared in the Letters section to date. Without a doubt, it is the finest computer magazine I have seen. Your articles and features are well written, extremely clear, and cover a wide

Continued on p. 27

99'er ROAD MAP

Traveler's Guide to Sights & Sounds Along the Journey to Understanding

LEGEND – Suggested Pathway for:

- A Newcomers to personal computing who don't as yet have their own microcomputer.
- B TI-99/4 and TI-99/4A users who are beginners to microcomputing.
- C More experienced users & programmers who are new to this magazine.

A program listing and its documentation.

An article meant for reading.

HOW TO READ THIS MAGAZINE

This road map is a reader aid. It suggests alternate pathways through the magazine's contents that are appropriate for different groups of readers. Since the articles cover a wide range of interest and are aimed at readers with all levels of experience—from beginners wanting to learn more before purchasing a microcomputer, all the way on up to professional programmers—additional reference information to help bridge the knowledge gap was also placed on the map.

First, there are the annotations that reference related articles in back issues. (New readers should refer to page 93 for order and availability information.)

Second, is the central glossary that further explains certain terms used in the articles.

Finally, the road map contains symbols which indicate if an article is simply meant to be read alone, or is primarily the documention for the accompanying program listing (the sequentially numbered lines of instructions that are meant to be typed, as shown, into the computer's memory, saved on cassette tape or disk, and then actually run—versatile software that transforms the computer into a video game, an instructor in education, a business or scientific tool, or even a personal manager). The appearance of *both of these symbols* indicates, as you'd expect, that the article is complete in itself, but also has an example program appended (for those inclined to run the software).

Please keep in mind that the pathways are just *suggestions*. Readers should explore the entire contents of the magazine in whatever order they feel comfortable. If an article seems too difficult at first, put it away for re-reading at a later date when your knowledge and level of understanding is certain to be greater.

Welcome Aboard
Your Journey is Under Way . . .

Rule of 78

ON SCREEN

Assembly Language—The symbolic, easier-to-remember form of a computer's **machine language** (the binary 1s and 0s that the computer can execute). After conversion to machine language (by an **assembler**), programs typically run much faster and take less memory than BASIC language programs.

bit—A binary digit (either 1 or 0) usually grouped in larger units such as the **byte** (8 bits) used to represent a character or number. The amount of memory your computer has is measured in bytes (e.g., 16K bytes=16,000 bytes).

floppy disk—A mass-storage device using a flexible mylar disk to record information. It is a more sophisticated alternative (quick random access) to cassette tape storage (sequential access).

Starting from Square One
How To Buy A Computer

99'er Glossary
Vol. 1, No. 4

Command Module—A plug-in plastic cartridge from Texas Instruments with memory chips that contain a computer program (**software**).

RAM—Random-Access Memory found inside your computer (or Expansion Memory peripheral) that you can put a program or data into (write) and get it out from (read).

ROM—Read-Only Memory found inside your computer, its peripherals, and Command Modules. Unlike RAM, it can only be read from, and not written to. **GROM** (Graphics ROM) is special TI memory that contain programs written in Graphic Programming Language (**GPL**) used inside the computer and Command Modules.

Super Language:
A Screen Printing Utility
Part 1:
Design Considerations
4

Computer Gaming
1. Designer's Spotlight
2. Gameware Buffet
3. Spriter
4. Chuck-A-Luck: Part 2 2
5. Computer Chess Corner.
6. Adventure Registry
7. Arcade Arbiter Review

TI's New Mini Memory Module

Overland Flow

Brader's TIps

Getting Down to Business

LOGO TIMES
1. Avoiding Turtle Traps
2. Problem Solving with LOGO
3. TI LOGO with a Little T.L.C.

OnLoCAltion
1. Notes on a Computer Score Part 2 1
2. Name That Bone
3. Professor Holl's Pocket Programs
4. The Scott, Foresman School Management Applications Development Process 3

1 = *Notes on a Computer Score, Part 1;* Vol. 1, No. 4; p. 40

2 = *Chuck-A-Luck, Part 1;* Vol. 1, No. 4; p. 24

3 = *CAI: The State of the Art;* Vol. 1, No. 3; p. 64

4 = *From Dots to Plots;* Vol. 1, No. 4; p. 17

THE RULE OF 78

By Harley M. Templeton

"Why Mr. Templeton, you can't figure that!" said the lady at the finance company. I had merely asked her the formula for computing the payoff amount on the installment contract on my 1978 Datsun.

This emphatic "can't do" send me racing off to the library in my soon-to-be-liberated Datsun. And it was there that I discovered the existence of the "Rule of 78." So armed with this knowledge, I decided to write a program that applied the "Rule" to installment contracts and let my TI-99/4 do the figuring for me.

From the name of this article you might have expected some sort of game, but the Rule of 78 is no game. It determines the amount of money required to pay off the contract at any given time, or the amount to be re-financed when you trade in before making all the payments. Should you be so unfortunate and have to default, the Rule of 78 determines the balance that becomes due and payable—the amount the finance company would be entitled to recover by repossessing the car. This Rule also is the method recognized by the Internal Revenue Service for computing the portion of the finance charge deductible each year during the life of the contract.

The Rule of 78 defines the fraction of the total finance charge that is on the unused portion. The numerator of the fraction is the sum of the numbers of the remaining payments; the denominator is the sum of the numbers of all payments. The number of the first payment is equal to the number of payments of the contract—e.g., 48 payments for a four-year contract. The number of each succeeding payment is one less; the last payment is number 1. At the time the Rule got its name, 12-payment contracts were the usual type. The sum of 12, 11, . . . , and 1 is 78, the denominator of the fraction. A more appropriate name in our day would be rule of 1176, which is the sum of 48 through 1.

Many installment contracts allow an "acquisition charge" to be deducted from the finance charge before multiplying it by the fraction. This is almost a prepayment penalty, but not quite—because you usually pay only a *portion of* the acquisition charge. When applicable, the acquisition charge affects the payoff amount of the contract.

The Rule of 78 is also known as the "Sum of the Monthly Balances Method" and the "Sum of the Months Digits Method." According to the *Consumer and Commercial Credit Installment Sales*, a subscription service published by Prentice-Hall, Inc., it is widely used in installment contracts. From these volumes containing Federal and state law on the subject, I discovered that the Rule is required by law in some states and allowed by laws of all states. It applies to installment contracts on automobiles, furniture, and appliances, and to some types of loans. Internal Revenue Service Publication 545, *Interest Expense*, explains the Rule and its application to income tax deductions.

7807 Lazy Lane
Austin, Texas 78757

"The Rule of 78 is no game. It determines the amount of money required to pay off the contract at any given time..."

Running The Program

The program is shown in Listing 1. It is written in TI BASIC, but will also run in TI Extended BASIC. Copy the program into your computer and enter the RUN command.

Consult a copy of the contract. First, be sure it mentions the Rule of 78 or one of its aliases in the section on prepayment. Then locate the amounts requested in the initial display. All of the amounts are usually typed in except the acquisition charge; it is printed in the contract. The display is as follows:

```
   INSTALLMENT PAYMENTS
AMOUNT FINANCED: $
FINANCE CHARGE: $
ACQUISITION CHARGE: $
AMOUNT OF PAYMENT: $
NUMBER OF PAYMENTS:
FIRST PAYMENT DATE:
```

The prompts of the display are typical of the names used in contracts. The amount financed is the sum of the price of the merchandise, sales taxes, insurance, etc., less the down payment. The finance charge is the amount added to the amount financed to compute the total of payments. The acquisition charge is printed in the section on prepayment. It is $25 in many contracts. The amount of payment is the easiest figure to come up with. It is the amount you pay each month. The number of payments is typically 12 on appliances and 48 on new cars. Enter the date of the first payment expressed as three numbers separated by slashes. The first number represents the month, 1 through 12. The second is the day of the month, 1 through 31. The last number is the year, represented by the last two digits. For example, if the first payment were due December 23, 1980, you would enter 12/23/80.

After you enter the figures, the program lists the options as follows:

```
CHOOSE ONE
   1. CONTRACT SCHEDULE
   2. CONTRACT STATUS
   3. TAX DEDUCTION
   4. NEW CONTRACT
ENTER NUMBER:
```

The contract schedule option provides the date, total paid, balance, prepay amount, and amount saved by prepaying for the first payment. By pressing ENTER you request the next payment. By repeatedly pressing ENTER you can display these five items for each payment of the contract. On the display for December of each year the program also displays the tax deduction for the year.

When you specify the contract status option, the program requests a date. The program then displays the status of the contract on that date. If the date is during the period of the contract, the status display includes the date, total paid, balance, prepay amount, amount saved by prepay, and the tax deduction for the year if the contract is prepaid on that date. The status figures, of course, apply only if all payments have been made up to the requested date.

The tax deduction option shows you the allowable income tax deduction for each year of the contract. This is the same information provided in the contract schedule displays; by giving you only these figures, this option is much quicker. In many cases, prepayment is not possible, but deducting the proper portion of the finance charge is important.

The new contract option returns to the beginning of the program and requests the inputs previously described. If you were really into installment contracts, you could compute the figures for the contract for the car, then for the TV, etc. Option 4 would enable you to enter figures for each additional contract.

If you select option 1, 2, or 3, the program lists the values you entered at the top of the screen, as follows:

```
AMOUNT FINANCED: $2,545.73
FINANCE CHARGE: $     781.03
ACQUISITION CHARGE:$  25.00
AMOUNT OF PAYMENT: $ 92.41
NUMBER OF PAYMENT:      36
FIRST PAYMENT:      12/23/80
```

Below this display, the specific display for the selected option appears. For the contract schedule option, the following display is repeated for each payment:

```
CONTRACT SCHEDULE

AFTER PAYMENT ON        12/23/80
TOTAL PAID        $       92.41
BALANCE           $    3,234.35
PREPAY AMOUNT     $    2,558.92
SAVE BY PREPAY    $      675.43

DEDUCTION FOR 1980 $42.22
```

For the contract status option, the initial display requests the date, as follows:

```
CONTRACT STATUS
ENTER DATE:
```

Enter a date in the format previously described. If you enter a date before the month of the first payment, the following is displayed:

```
STATUS ON 11/30/80
 TOO EARLY
```

On the other hand, if you enter a date later than the last day of the month in which you will make the last payment, the following is displayed:

```
STATUS ON 12/1/83
PAID UP
```

When you enter a date during the period of the contract, the following is displayed:

```
STATUS ON 12/31/81:

TOTAL PAID        $    1,201.33
BALANCE           $    2,125.43
PREPAY AMOUNT     $    1,838.23
SAVE BY PREPAY    $      287.20
DEDUCTIBLE IN 81  $      451.61
IF PAID OFF ON 12/31/81
```

For the tax deduction option, the display is as follows:

```
IF YOU PAY ALL PAYMENTS
AS SCHEDULED, YOU MAY
DEDUCT FINANCE CHARGE
AS FOLLOWS:

YEAR              AMOUNT
1980        $       42.22
1981        $      415.14
1982        $      246.27
1983        $       77.40
```

At the bottom of each screen, the program displays the following message:

```
PRESS ENTER TO CONTINUE
OR 9 TO QUIT
```

For the contract schedule option, you get the figures for the next payment when you press ENTER. When all payments have been displayed, pressing ENTER displays the list of options previously described. For options 2 and 3, which have one screen each, pressing ENTER displays the option list.

The accuracy of the figures depends on the accuracy of the computer. Texas Instruments claims ten digits of accuracy for the TI-99/4. In the case of the contract on my 1978 Datsun, the finance company's figures were not exactly the same as mine. The differences were a penny or two, most likely due to differences in computer accuracy. Of course, I paid the amount their computer wanted.

Changing the Program

If you have a printer, you will want to change the program to print the data displayed on the screen. You will probably change the format. The contract schedule can be printed in tabular form, one line per payment, on an 80-column printer.

You may not want all the options. The program has a subroutine for each option; you can leave one or two out. The contract schedule subroutine begins on line 640, and ends on line 1500. The contract status option begins on line 1500. The contract status option begins on line 1520 and continues through line 2240. The tax deduction subroutine occupies lines 2260 through 2610. Each subroutine is independent of the other two; however, the driver, lines 100 through 620, and the miscellaneous subroutines from line 2630 to the end of the program, are required for all subroutines.

Running the Program in TI Extended BASIC

You can run the program in Extended BASIC as it is. But you can streamline it, exploiting some of the features of the more powerful language. The power of the DISPLAY statement of Extended BASIC is particularly valuable in this program.

Line 130 is a DEF statement that defines a rounding function. A format defined by an IMAGE statement automatically rounds fractions, and this function is used to align decimal points in the displays. The function is not needed if you use a specified format.

The subroutine beginning at line 2840 displays a string at a defined point on the screen. When you use a DISPLAY statement with the AT option this subroutine is not required. Similarly, the subroutine at line 2900 adds zeros to the right of the decimal point, when required. It also inserts a comma between the hundreds and thousands digit of numbers greater than 999.99. A defined format adds least significant zeros but does not insert the comma. If you want to use a format and give up the comma, omit this subroutine.

To incorporate these changes, modify the program shown in Listing 1 by performing the following steps:

1. Omit line 130 and modify line 140 as follows:
   ```
   140 IMAGE " #####.##"
   ```
2. Omit lines 450 and 460; modify line 470 as follows:
   ```
   470 PRINT USING
       "AMOUNT FINANCED :
       $ ##### . ##" : UB
   ```
3. Omit lines 480 and 490; modify line 500 as follows:
   ```
   500 PRINT USING
       "FINANCE CHARGE:
       $ #####.##":FC
   ```
4. Omit lines 510 and 520; modify line 530 as follows:
   ```
   530 PRINT USING
       "ACQUISITION CHARGE:
       $####.##":AC
   ```
5. Omit lines 540 and 550; modify line 560 as follows:
   ```
   560 PRINT USING "AMOUNT OF
       PAYMENT: $####.##":
       PMNT
   ```
6. Omit line 590 and modify line 600 as follows:
   ```
   600 PRINT USING "FIRST PAY
       MENT: ##/##/##":MO,
       DA,YR
   ```
7. Omit lines 770, 790, and 800; modify line 810 as follows:
   ```
   810 DISPLAY AT(14,17):USING
       " ##/##/$$":CMO,DA,
       CYR
   ```
8. Omit lines 840-870; modify line 890 as follows:
   ```
   890 DISPLAY AT(15,17):USING
       140:TOTPD
   ```
9. Omit lines 910-940; modify line 960 as follows:
   ```
   960 DISPLAY AT(16, 17):USING
       140: BAL
   ```

10. Omit lines 1040-1070; modify line 1090 as follows:

```
1090 DISPLAY AT(17, 17): USING
     140: PREPAY
```

11. Omit lines 1100-1130; modify line 1150 as follows:

```
1150 DISPLAY AT(18, 17): USING
     140: SAV
```

12. Omit lines 1240-1270 and 1290; modify line 1300 as follows:

```
1300 DISPLAY AT(20, 1): USING
     "DEDUCTION FOR 19##
     $####.##": CYR, ADED
```

13. Omit lines 1390-1420 and 1440; modify line 1450 as follows:

```
1450 DISPLAY AT(20, 1): USING
     "DEDUCTION FOR 19##
     $####. ##": CYR, ADED
```

14. Omit lines 1790 and 1800; modify line 1810 as follows:

```
1810 PRINT USING "TOTAL PAID
     $    #####.##": TOTPD
```

15. Omit lines 1840 and 1850; modify line 1860 as follows:

```
1860 PRINT USING "BALANCE
     $    #####.##": BAL
```

16. Omit lines 1930 and 1940; modify line 1950 as follows:

```
1950 PRINT USING "PREPAY
     AMOUNT $    #####.##":
     PREPAY
```

17. Omit lines 1960 and 1970; modify line 1980 as follows:

```
1980 PRINT USING "SAVE BY
     PREPAY $    #####.##":
     SAV
```

18. Omit lines 2100 and 2110; modify line 2120 as follows:

```
2120 PRINT USING
     "DEDUCTIBLE IN ##
     $ #####.##": SYR,
     DEDUCT
```

19. Omit lines 2400 and 2410; modify line 2390 as follows:

```
2390 PRINT USING "19##    $
     #####.##": DYR, DED
```

20. Omit lines 2550 and 2560; modify line 2540 as follows:

```
2540 PRINT USING "19##    $
     #####.##": DYR, DED
```

21. Omit lines 2650-2670 and modify line 2680 as follows:

```
2680 DISPLAY AT(23, 1): "PRESS
     ENTER TO CONTINUE"
```

22. Omit lines 2690 and 2700; modify line 2710 as follows:

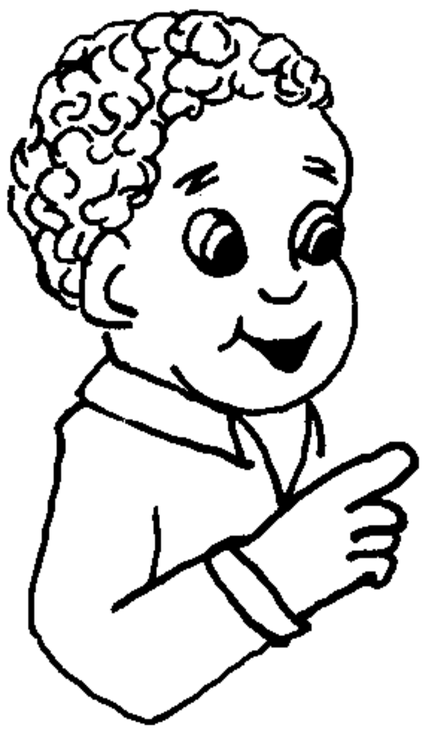
```
2710 DISPLAY AT(24, 1):
     "OR 9 TO QUIT"
```

23. Remove references to function RND2 in the following lines:

```
1010 SAV=RUL 78(AFC)
1230 ADED=DEDUCT/DEN*FC
1360 SAV=X
```

Continued on p. 83

Texas Instruments Home Computer



TI-99/4A Console	**325.00**
Peripheral Expansion System:	
Case	199.00
Disk Drive	269.00
32K Memory Expansion	229.00
UCSD p-System	189.00
Language Modules:	
Editor/Assembler	76.00
Extended Basic	76.00
Learning:	
TI LOGO	76.00
Early Learning Fun	24.00
Speak & Math™	24.00
Beginning Grammar	24.00
Fun:	
TI Invaders	33.00
Adventure	41.00
Munch Man	33.00

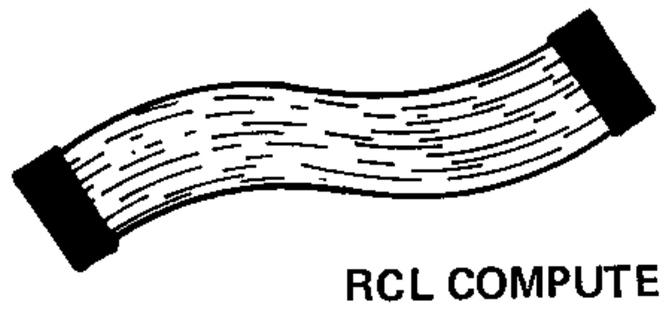
AND MUCH MORE

Write for our new catalog with lots of new TI and third party software and peripherals. Because we specialize in the TI-99/4A, we can answer your TI Home Computer questions!

KOMPUTAR WORKS

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

Now, You Can Separate Your TI 99/4A Computer From Your TI Peripherals



Send Order With Payment To

RCL COMPUTERS
359 Washington Ave.
P. O. Box 224
Oakmont, PA 15139
(412) 828-4301

ORDER FORM

SHIP THE FOLLOWING:

_____ **Unshielded 2' Cable At $31.95ea.**
_____ **Unshielded 3' Cable At $36.95ea.**

Shielded Cable Prices Upon Request

☐ Check No. _____ Amount _____

☐ Money Order _____ Amount _____

Add $3.00 for postage

PRICE F.O.B. Pittsburgh, PA.
PA. Deliveries add 6% Sales Tax
Allow 3-4 week delivery

SHIP TO: _____

NAME _____

ADDRESS _____

CITY _____ STATE _____ ZIP _____

NEW **NAME-IT** NEW

Mail Lists / Labels / Files

Capacity: Cassette — 30 to 150 records per file
Diskette — 300 records per file
Records: up to nine 28-character items per record.
Prompts: user designated record item prompts.
Both cassette and disk versions include:
Complete FILE SORT (50 in 2 mins, 100 in 5 mins)
DOUBLE SORT (ie: alphabetically by city).
SEARCH by any record item (ie: print labels for all members in one city or only paid members).
Record items may be PRE-SET (ie: L.A., CA appears but may be changed at input.)
PRINT: ALL, SELective (search) or INDIV. choice.
Print STYLES: 4 standard label styles;
columns or row series (items or order selectable);
or easily added do-it-yourself program lines.
Safe utilization of maximum console RAM.
CONVERT existing files using guidelines in instruction booklet.
Generate FORM LETTERS with TI-PWRITER that use NAME-IT record items in test (ie: Dear ****,).

CASSETTE: $32.00/**DISKETTE** $35.00

TI-PWRITER

a complete WORD PROCESSOR

FULL upper & lower case (including 99/4) — even on screen print.
ANY Input/Output storage of text — disc, cassette, cassette input/disc output, or vice versa.
Holds 3000 characters (before storage) — 50,000 characters per disc or 60 minute cassette.
COMPLETE Software Control of Printer (depending upon its capabilities) — for enhanced print, underlining, formatting.

CASSETTE: $32.00/**DISKETTE**: $35.00

Both NAME-IT & TI-PWRITER:

ANY Printer — T.P., RS232C (or screen only).
Input typing SPEED — over 100 words per minute.
ANY Line Length — 28 to 254 characters per print line.
Complete text EDITING — by cursor control; including insert & delete lines, partial print, printer halt or abort without text loss, page FWD & BKWD.
No Special Equipment — monitor, console, Extended Basic module, storage device, printer (or a friend with one).
Each comes with a 20 page instruction booklet.

GAMES

Extended Baseball — full sprite action, dual joystick control.
Extended Hangman — 580 words, graphics, (speech optional).
Gorfia Pestulitis — fight invading space monsters (joysticks).
Tic-Tac-Toe — 4 levels of play (speech optional).

CASSETTE: $9.95 for 2/$16.95 for all 4
DISKETTE: $12.95 for 2/$19.95 for all 4

• *More programs coming soon.* •

• *Extended BASIC Module Required for ALL Programs* •
• *All are 99/4 & 99/4A compatible* •

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

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

TI's New MINI MEMORY MODULE:

There's More There Than Meets the Eye

By Ira McComic

You know, looks can be deceiving. For example, who would ever suspect that a bespectacled, mild-mannered reporter for the *Daily Planet* could actually hop over a skyscraper with a single bound? In the same way, there's a lot more to TI's new Mini Memory Command Module than meets the eye. It appears to be just a normal, garden-variety Command Module—the same size, same shape, the same black, unassuming case. But lying inside that plain cover are the ingredients to convert your TI Home Computer from a good BASIC programming machine into a trim and efficient assembly language instrument.

The name itself is a clever disguise. "Mini" Memory, indeed! If you believe that all there is inside that plastic case is just a little bit of memory, then you'd probably believe the Trojan Horse was nothing more than an overgrown child's riding toy! There's actually 14K bytes of memory sitting in there: 4K bytes of RAM, 4K bytes of ROM, and 6K bytes of GROM.

RAM (read/write) memory is the kind of memory in which your programs are stored. You know how it works. You write a program and it's stored in RAM. When you turn off the computer, the program you wrote is gone, unless you had the foresight to save it on a cassette or diskette before you flipped the switch on the computer. Now, here's the first surprise from the Mini Memory Module: You can store a program or data in its RAM and your program or data is not lost when you turn off the computer. In fact, you can store a program or data in the Mini Memory Module's RAM, turn off the computer, remove the module from the console and carry it around with you, just as you can a cassette or diskette. This seeming miracle is possible because the module contains its own life-support system. A battery inside the module keeps the contents of the RAM alive. The RAM components are CMOS devices and are power misers. It only takes a small trickle of current to keep the little critters alive when they are not being used. The battery inside the module should last you a couple of years. When you finally need a new battery, you can exchange the module for a new one.

Battery-backed RAM can be real handy, if you think about it. Just imagine—you can write a new program, store it in the Mini Memory Module, turn off the computer, yank out the module, zip over to your friend's house where you plug in the Mini Memory Module, and instantly load your new program into your friend's computer so he can admire your handiwork. No cassettes, no disks, no messy cables, and no long waits.

Besides the battery-backed RAM, the Mini Memory Module also has 6K bytes of GROM (Graphics Read Only Memory) and 4K bytes of ROM (Read Only Memory). Inside the GROM and ROM are seven additional TI BASIC subprograms, including PEEK and POKE [See *Brader's TIps* in this issue for an explanation of PEEK and POKE—Ed.] This same GROM and ROM also gives you access to many of the computer system's routines from assembly language programs. There's still more. As you develop your assembly language programs, you've got some help. The ROM contains a powerful program debugger called EASY BUG which you can use to exterminate those pesky "logic vermin" which sometimes infest programs.

At this point, you may be saying to yourself, "What good does all this assembly language access and debugging stuff do me, anyway, without an assembler?" Glad you asked. Here's the next bit of exciting news. The Mini Memory Command Module comes with a cassette which contains an assembler. You can load this assembler into memory, enter assembly language statements, and have the assembler translate the statements into 9900 object code for you.

See what I mean? There's a lot more to this Mini Memory Module than its alliterative name . . . The Mini Memory Module is, indeed, full of pleasant surprises. Here's what we've discovered so far: First, it's got 4K bytes of battery-backed RAM where we can save programs and data. Secondly, it's got 10K bytes of ROM and GROM which contain additional TI BASIC subprograms, provide a passport into the computer system's routines, and hold the EASY BUG debugger. Thirdly, the Mini Memory Module comes with an assembler so you can create your own assembly language programs. That ought to be enough to even coax a smile out of a mother-in-law, but there's still one more surprise: On the same cassette with the assembler is a bonus. It's a demonstration program which produces a fascinating colorburst of patterns on the monitor or TV display.

Let's explore these items one at a time.

FILE STORAGE

Probably most persons will use the Mini Memory Module most often for temporary storage of programs and data. You can think of the Mini Memory Module as a very fast-access storage device. [Also see *Brader's TIps* for a general explanation of "files," and *Getting Down to Business* for a tutorial on "random access" files—Ed.]

Additional Files

When you have the Mini Memory Command Module plugged in, the 4K-byte RAM has the file name MINIMEM for TI BASIC program and data storage. The RAM occupies physical addresses 28672 through 32767 (hexadecimal 7000 through hexadecimal 7FFF). You can save programs in this file and load programs from it. (For example, to save a TI BASIC program, just enter the command SAVE MINIMEM.) You can also store data in this file using the file specifications available for any TI BASIC file. For example, the following statements open the Mini Memory file and store four data values in the file.

```
OPEN #3:"MINIMEM",RELATIVE,FIXED,UPDATE,
INTERNAL
PRINT #3: A,B,C,D
```

With the Mini Memory Module you can also access a second new file. EXPMEM2 is the name of a 24K-byte memory file located in the 32K Memory Expansion unit. EXPMEM2 is only available, however, if you have the Memory Expansion unit connected to your computer and turned on.

ROM AND GROM PROGRAMS

Inside the Mini Memory Command Module's 10K bytes of "firmware" (that's computer talk for ROM and GROM) are additional TI BASIC subprograms, hooks to connect assembly language programs to the computer system's programs, and a program debugger for machine language programs.

Additional TI BASIC Subprograms

Seven additional TI BASIC subprograms are yours with the Mini Memory Module. These subprograms are PEEK, PEEKV, POKEV, CHARPAT, INIT, LOAD, and LINK.

Continued on p. 22

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.

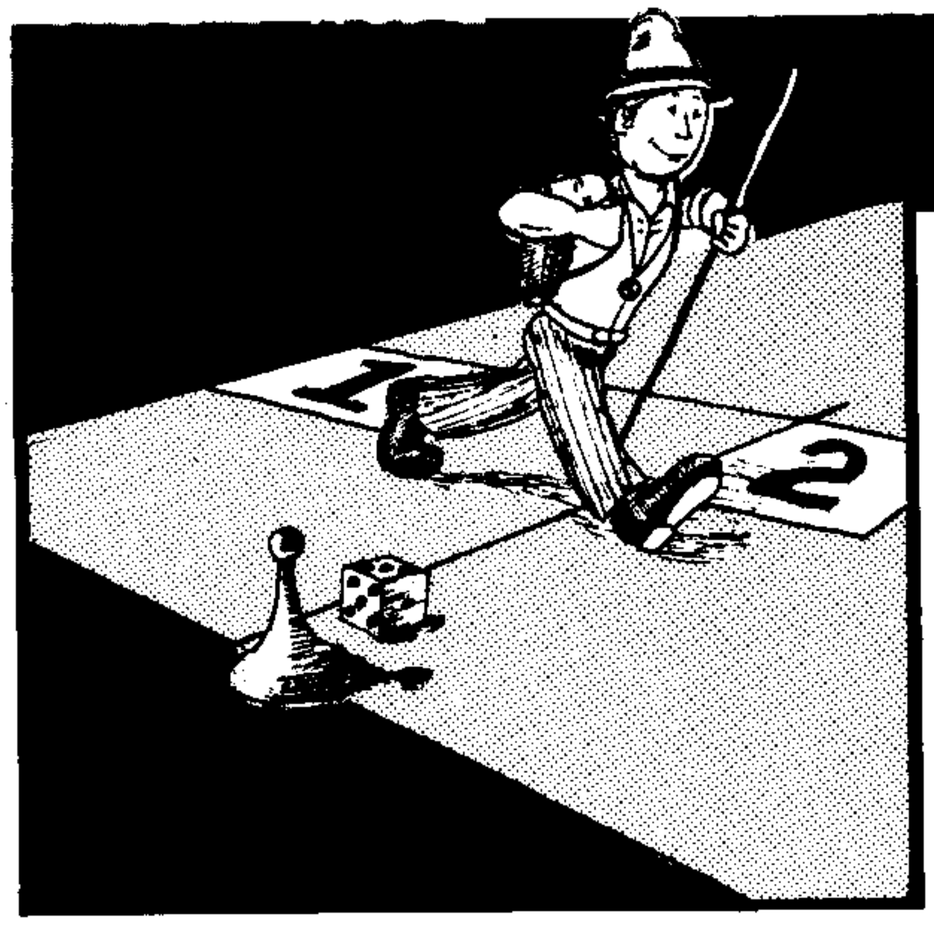
What's next? Wait and see. We're already expecting.

EPSON
EPSON AMERICA, INC.

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

FEATURE	ORIGINAL MX-80	GRAFTRAX-80*	ORIGINAL MX-100	MX-80 with GRAFTRAX-Plus	MX-80 F/T with GRAFTRAX-Plus	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.



How to Buy A Computer

> "... be assured that you are embarking on an exciting adventure ... and realize that ownership is not only exciting but helpful and productive too."

By Samuel L. Jenkins

In this article I will offer some suggestions to help those of you shopping for your first personal computer. I will not directly compare brand names, nor will I attempt a technical critique of the TI-99/4A home computer—but I will point out some of the TI machine's exceptional features.

What follows is a general discussion of computer shopping techniques written by and for the computer novice who is experiencing the bewilderment of trying to make a wise computer purchase in a market exploding with new products. I offer these suggestions from the perspective of a writer who is not a computer professional. I have owned a TI-99/4 for a year and a half and also recently bought a competitive brand computer. In addition, I plan to purchase a third brand during 1982. Therefore, I am not dedicated to a single brand of computer although I am impressed with the 99/4A capabilities. All of my comments apply equally to the 99/4 and the new 99/4A unless otherwise noted.

My computers are used to develop computer-assisted instruction (CAI) for applications in the field of rehabilitation. The following suggestions result from the actual experiences of a beginner faced with the task of learning about computers—one who has spent literally hundreds of hours pouring over manuals and magazines, and peering into a monitor screen.

Since my background is in psychology and counseling I can't resist beginning with some general, facilitative remarks. First of all, no matter which brand of computer you eventually buy, you have a high probability of regretting your choice at times. No single computer will have all the features you want (or grow to want). Like any other machine, your computer will have some compromise features; there's also the matter of the grass always looking greener...etc. Typically, after a major purchase like a car or a computer, we set about to convince ourselves that we made the best possible choice even if there is considerable evidence to the contrary. So be aware that your buyer's anxiety may not totally disappear the instant you take possession of your new computer. Secondly, regardless of how impressed you are with your new computer's gee-whiz features, you will quickly adjust your expectations upward. What seems incredibly exciting during the honeymoon can be routine after your computer relationship has matured. Whatever you buy now, you will probably soon want to expand either with more hardware (machinery/gadgets) or software (programs which tell the machine what to do). Thirdly, start now! Don't wait for computers to come down to $9.98—they probably never will. The manufacturers will just keep on making them more sophisticated for about the same money. Oh sure, you can save a couple hundred dollars with wise shopping but don't expect to get a computer for the cost of a cheap calculator in the near future. In the meantime, there is a lot to learn and a lot to do with your computer. And lastly, don't expect your friends, spouse, etc., to be as thrilled as you are about your computer. It is up to *you* to educate *them*.

907 6th Ave. East
Menomonie, WI 54751

Who Buys Personal Computers

Rumor has it that someone once tried to profile the "typical" personal computer buyer so that marketing strategies could be more effective. After all the survey data was analysed, the only factor shared by the majority of people was the desire to write a very successful program and market it to become rich. Otherwise, the world of personal computer owners is populated by every conceivable type of individual—people using their computers for myriads of different purposes.

Just so you know that you have a lot of company out there in your pursuit of purchasing a computer, see if you don't recognize yourself in one or more of the following categories:

Type 1—The electronics amateur who is intrigued by all the latest technology. This person enjoys fiddling with the equipment. He doesn't have to have a reason for getting a computer other than "it's there." We should be greatful to these people because they started the home computer craze a few years ago when they began buying kits by the thousands and started tinkering around with various applications.

Type 2—The aware parents who want the family to be up on "the latest." The family can play games and learn about computing as well as do the budget, and so on. This is the home computer market which has yet to reach its full potential. (How many people do you know who own a computer?) The average family will want a flexible computer capable of many different functions at a low price. Their machine should be "friendly" and easy to use. It should be expandable so that later exhancements can be added as the family's needs grow in sophistication.

Type 3—The small business owner or professional person who wants to automate the office. These individuals will agonize over the question of how much computer to buy. If they don't get "enough," it could well end up at home for the kids to play with; but why buy a $10,000 system if a $3500 package will do the job. This system should be capa-

ble of storing and retrieving large amounts of data and possibly do word processing in the office.

Type 4—The educator interested in computer-assisted instruction (CAI). He or she will need a computer capable of displaying eye-catching color graphics and animations along with text, speech, and sound.

Type 5—The scientist or engineer who will use the machine at work or home because it is easier than standing in line to get on the company's big mainframe computer. Even companies that own big "brainiac" computers are buying micros to spread around to key people.

The list could go on and on, but I hope I have made my point that the "typical" microcomputer buyer is anything but typical. We all have one thing in common, however. We have all been bitten by the computer bug and the only known cure is to take the plunge and get our very own microcomputer!

Types of Sellers

If you as a buyer are feeling overwhelmed by all the computer choices available, pity the typical salesperson. He may be a store clerk, more at home with stereos and televisions, who has been reassigned to the new computer department. Or at the other end of the spectrum, he may be a programmer or technician who also sells computers. Odds are that you will be met by the former more often than the latter when you visit your local computer outlet. Just as buyers exist at every level of sophistication, so do sellers. It is my opinion that absolute knowledge about computer technology is less important than the *attitude* of the salesperson and their willingness and desire to help *you* learn about computers. After all, almost all of us are new to the world of personal computing.

If you haven't already, you will shortly encounter at least one of the following salespeople:

Type 1—The sincere young man or woman who introduces themselves with a nervous smile and confesses, "I only started in this department yesterday; let me see, where is the power switch on this little beauty . . . " Don't leave too soon though. If you have the time and patience, you and the trainee can learn a lot about the computer in an hour.

Type 2—The equally sincere salesperson who introduces himself and says, "What can I show you . . . we have a 48K whiz-banger with double density DOS and CP/M on special . . . " This individual will joyously prattle on until your glazed eyeballs communicate either lack of interest or comprehension. (They are equivalent in the clerk's opinion.) You can then leave the store with a handfull of pamphlets and a heart full of doubt—and possibily a car full of computer.

Type 3—The merchandizing expert who moves computers the same way he used to move TV sets, stereos, etc. This type cannot refrain from knocking the competition by saying things like, "Brand X is almost out of business, that's why we don't handle'em . . . what'd ya' say you do for work . . . I sell a lot of Number Crunchy 100's to people in your field." This individual may be able to tell you a lot about his computer since he will be shrewd enough to read up on all the features of his machine; you may actually learn something if you have the confidence and patience to endure a barrage of irrelevancies.

Type 4—The skilled and sensitive sales professional who has developed a good knowledge of computing, or vice versa, the computer professional who has developed basic competence as a salesperson. This person will ask you right off what you want to do with your computer and help you with the answer if you aren't sure. You will appreciate this individual's patience and willingness to find out information for you. He or she will consult with a superior or even call the manufacturer without fear of appearing ignorant. When you meet people like this, respect their time and effort and show your appreciation. We don't want them to get discouraged and switch jobs. There will be little danger of this, however, since they will probably be making a lot of sales with many happy customers!

How To Shop

Be careful not to equate the amount of advertising you see for a computer with it's technical sophistication or suitability for your needs. The advertising of a well known imported car used to extoll the virtues of the air-cooled, rear-engine design until it could no longer meet emission standards and deliver competitive gas mileage. Other companies had been marketing water-cooled, front-wheel drive models for years and had been ridiculed as stone-age engineers by the air-cooled company. But when time came to switch over to front-wheel drive, water-cooled cars, the air-cooled company would have you believe that *they* practically invented this new and radical design . . . So much for advertising. I am continually impressed with the features of the TI-99/4 as I grow more familiar with it and other brands, but very little of this knowledge came from advertising, or for that matter, from sales people. It came from *use* of the machine.

So take the time to go beyond mere advertising when you shop. Try to talk to some people who own a computer, or visit a local computer club. But remember, as pointed out earlier, people try to convince themselves that their choices are best, so expect some very prejudiced views. Also, be cautious about believing everything you read in magazine reviews about various brands of equipment. I am especially leary of articles which evaluate a couple dozen brands on 10 or 15 variables. They look impressive with charts and diagrams but sometimes have outright erroneous data. I have read such articles that declared that the TI machine had no high-resolution graphics or memory expansion capabilities. Well, TI has one of the best high-resolution color graphic capabilities on the market and can be expanded to a 48K system. I have noticed similar errors on other brands, too. So visit several stores, read a few computer magazines like *99'er*, and get your confidence up so you won't be intimidated by the salesperson.

You may also need to know a little computer jargon, although the better salespeople will avoid trying to impress

you with their vocabulary. If you don't already have one, pick up a glossary of terms while you are out for your first visit to the computer store. For starters, you should know the meaning of terms like the following:*

BASIC	**Integrated circuit**
Bit	**Input/Output**
Byte	**Microcomputer**
Chip	**Modem**
CRT (monitor)	**Peripheral**
Disk Operating System (DOS)	**Printer**
	Program
Disk Drive	**RAM**
Diskettes	**ROM**
Firmware	**RS232C**
Hardware	**Software**

With just a few of these terms tucked away in your memory banks, you can walk into the computer store with more confidence and less quiver in your voice when you ask to see the "Brainiac 3000" computer.

Ask to see a demonstration of each computer you think you can afford. But be aware that many demonstration programs you see are written in a program language other than BASIC—i.e. the language available to the user on most small computers. Consequently, the demo may be super impressive with lots of color graphics, animation, and sound, but find out if you can duplicate these effects readily with the BASIC programming language available to you. If you are interested in having good color graphics in your programs, ask the salesperson to enter some simple statements in BASIC to illustrate the computer's ability to perform the following:

A. Clear the screen.
B. Change the screen color.
C. Plot color shapes on the screen. Try to place a "duck" or a "car" on the screen. Find out if the user can create his own shapes or is he limited to pre-defined shapes stored in the computer's memory.
D. Place a graphic shape and text on the same screen. Some computers can do one or the other but cannot do both without elaborate and difficult programming.

Happily, the 99/4A does all of the above with ease. You can program in 16 colors with simple, easy-to-use BASIC statements. If graphics are important to you, check out the TI Extended BASIC graphics capabilities. They are sensational and compete with computers costing as much as a thousand dollars more. If you want sound capabilities in your programs, ask for a demo of the following:

A. Play a three note chord.
B. Play a simple scale.
C. Demonstrate the highest and lowest frequency programmable.
D. Demonstrate the loudest and softest volume of sound possible.
E. Create sound effects like a "choo-choo" or an "explosion."

Speech synthesis adds an exciting dimension to computing, especially in educational programs. Texas Instruments makes it easy to integrate speech into BASIC programs with its speech synthesizer and *Speech Editor* Command Module or the *Terminal Emulator II* Command Module. The *TE-II* will synthesize any English word typed into the computer; the *Speech Editor* will allow you to choose from a resident vocabulary of over 300 words. By all means get a demonstration of speech synthesis if you are interested in computer-assisted instruction—it is well worth the added cost.

The Editor

Regardless of the type of use you plan for your computer, you will definitely need a good editor. This feature is not discussed much when you are shopping, but it is extremely important. However, if you can think and type without errors, you can skip this section and not worry about editing. Otherwise, pay special attention to the following discussion:

Good, you are honest! I found out the importance of an editor the hard way. Not one salesperson mentioned this feature in any of my shopping except to say that you could correct errors. From the typical treatment of this subject in the stores, you might conclude that all editors are alike. Nothing could be further from the truth. There is a galaxy of differences between computer brands and their editing capabilities. Those differences can either make your computer a joy or a pain to use.

So, what is an editor? Somewhere inside your computer, buried in all that fabulous circuitry, is a component which interprets all of the instructions you type into the computer. It turns your instructions into the one's and zero's that the computer understands. It doesn't understand words--just one's and zero's. This component is said to interpret the program for the computer. It also will do the work of editing or changing program statements after they have been entered into the computer. When you are writing and debugging (removing errors from) BASIC programs, you are bound to make typing errors. To correct these, you could type the offending line over completely and all would be well. But it is very time consuming and aggravating to retype a line having perhaps 20-50 characters just to correct one or two mistakes! It is much better to go into the program line and edit the mistakes without disturbing the rest of the line.

A good editor will permit you to modify a line of program code by inserting, deleting, or erasing characters and words while displaying the changes on the monitor screen exactly as made in the program. A poor editor will require complicated key strokes and fail to display the corrections on the screen as they are made. It will make you pound on many keys and ultimately resort to retyping the whole thing over. The 99/4A's editing capabilities are far superior to my Number 2 computer. In fact, the TI editor is equivalent to a good word processor in its capability to correct a line of program code. (I am writing this article on my 99/4 using a simple word processing program that I wrote myself. It uses all the editing features resident in the computer and works very well for editing text.)

I cannot over emphasize the importance of the editor, and strongly recommend that you evaluate it carefully before you make a decision about a brand of computer. The surest way to test the editor is for *you* to sit down at the keyboard and have the salesperson walk you through some editing functions. Don't be satisfied with having the clerk do it for you because he'll probably make it look easy by doing something exceedingly simple. You might try the following exercise.

Enter this program line:

```
100 PRINT "NOW IS THE TIME FOR
    ALL GOOD MEN TO COME TO
    THE AID OF THERE COUNTRY."
```

(If you are totally new to programming I should point out that this BASIC statement will cause the words inside the quotes to be displayed on the monitor if you RUN the program.)

Notice that the word *THERE* is misspelled; so correct the spelling without retyping the entire line, then insert the word *BEST* before the word *TIME*. If you can't accomplish this by the store's closing time, ask the salesperson to do it; if he can't do it with ease, give serious thought to buying another brand of computer.

While you are at this, ask the salesperson to demonstrate *resequencing* for you. Resequencing is a simple but valuable (and frequently unavailable) feature with which your computer should be equipped. It permits you to renumber your program line numbers so that you can insert additional lines into an existing program. For example, you might type in this simple BASIC program:

```
10 PRINT "HELLO"
11 PRINT "WHAT IS YOUR NAME?"
12 INPUT N$
13 PRINT "THANK YOU";N$
14 END
```

Notice that you don't have any room between lines for additional lines. If you later decide to change the program, you either have to type the program over or resequence the line numbers to provide

*See my simple definitions at the end of the article.

Texas Instruments TI-99/4A Home Computer Products

			Your Cost
CONSOLE			
PHC	004A	TI-99/4A Home Computer	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	User's 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	5038	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

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

APPLICATION PROGRAMS CONTINUED

Education/Personal Enrichment Continued

		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

		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

*Available in First Quarter 1982.
**Available in Second Quarter 1982
†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	$159.95
TI-58C	87.95
PC100C	148.95
Programmer (LCD-*new*)	59.95

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 7 Days a Week!
7:00 AM to 7:00 PM Mon-Sat.
SUNDAYS NOON to 5:00 PM
Order Desks: (213) 739-1130
TOLL-FREE (outside Calif.) 800-421-8045
TOLL-FREE (within Calif.) 800-252-2153

All goods subject to availability; this ad supersedes all previous ads; we are not responsible for typographical errors; we will meet or beat almost any advertised price if the competition has the goods on hand.

Minimum shipping & handling $4.95
All orders subject to verification & acceptance.
Send $2.00 in U.S./$5.95 foreign request of for our 112-pg catalog-great prices!

space. Normally, you don't intentionally get yourself into corners where it is necessary to resequence your programs, but it does frequently happen (courtesy of Murphy's Law). On the TI machine, resequencing is easily accomplished by typing **RES** and pressing the **ENTER** key. Presto! The program looks like this:

```
100 PRINT "HELLO"
110 PRINT "WHAT IS YOUR NAME?"
120 INPUT N$
130 PRINT "THANK YOU";N$
140 END
```

Now you can add additional lines between the original ones. Many computers do not have the resequencing function built in so you have to load in a separate program from a disk or tape. This function is important enough that it should be built into the machine as it is in the TI-99/4A.

General Considerations

Regardless of the sophistication of the system, you should expect certain fundamental "creature comforts." First, it is mandatory that the character display on the monitor screen be clear and easy on the eyes. You may not fully appreciate this during a brief demonstration in the store, but spend an hour or two peering at the screen in your basement, and you'll know what I mean. Note: Characters displayed on a 9-inch monitor in the store may not appear so sharp when you hook up to the 19-inch TV at home. On big screen displays the characters become more ragged due to the fact that the dots composing the characters are larger and spread out over a greater area. Some computers display white characters on a black background and vice versa. The TI has exceptionally sharp black characters formed by an 8 x 8 dot matrix with a very pale blue background. Or you can change the character and background colors to any of 16 colors. This is especially useful for educational programming.

My only criticism of the TI display capabilities is that with TI BASIC it is limited to a line of 28 characters for text or 32 for graphics. [With TI's Editor/Assembler Command Module, you have a 40 character "window" which automatically scrolls horizontally across an 80-column "page." The Video Display Processor chip inside the computer actually has a 40-column "text mode," and the software produces the doubling effect—Ed.] Some computers display less, but many are capable of displaying lines up to 80 characters or more. My Number two computer displays 40, but I see little practical difference between it and the TI machine. However, the 80-character display is desirable if you plan to do extensive word processing (letters, reports, etc.). Having lower-case characters is also a nice feature if you plan to do word processing. The new TI-99/4A has a type of lower-case which is actually a compressed upper-case text; it works very well. Please note that you can do word processing with the 28 character line format, but the display on the screen will not be spaced exactly as the text will appear on the printed page that the printer puts out. The advantage of having an 80-column screen display is that you can see what your printed output will look like before it gets to the printer. In any case, your printer (one of those hardware enhancements you'll be wanting as mentioned earlier) will easily format output from any computer so that it fits on the printed page just as it should. The printer should also have the capability of printing both upper and lower case characters with the proper program, so that you need not worry about having lower-case resident in your computer (but it is nice to have).

Another "creature comfort" to consider is the computer keyboard. The original TI-99/4 was criticised for having a keyboard smaller than a conventional typewriter. Actually it is very easy to use and one can touch type on it very efficiently. But TI has modified the keyboard on the new TI-99/4A so it is more like a standard typewriter and added some function keys and a repeat key to improve the computer's flexibility.

If you select a disk system for program and data storage rather than a cas-

sette tape system, you will have the advantages of speed and convenience but you will sacrifice something too. In addition to the higher cost of the disk system (maybe 10 times the cost of a tape recorder), you will also lose some of your program space (random access memory, known as RAM) inside the computer. Some systems will have a 2K overhead (2000 bytes) while others may require 10K or more. It is desirable to have a low overhead in the system if at all possible so that your valuable program memory space will be available for programs. The 99/4A disk system digests about 2K of your RAM leaving a nominal 14K for programs (on the standard 16K system). To put this into perspective, one page of typed, double-spaced material with liberal margins is equivalent to about 2K of information. If you make the mistake of buying a 16K computer which has a 10K overhead for the disk system, you would only have about 6K of program space remaining after you turn on the disk system. And one of the very popular computer brands actually has a 10K overhead! So when you look at one in a store, the salesperson will probably insist that you get (and pay for) at least 32K of RAM. Moral: 16K memory in computer "A" does not necessarily equal 16K memory in computer "B". Texas Instruments gives you a lot of memory for the money.

How much memory will you actually need in your computer? Well, a 16K computer is generally considered satisfactory for most home use. For business and educational applications you will probably need more memory—48K is satisfactory in most cases. That covers your program memory requirements inside the computer. For permanent storage of large amounts of data such as student grade records and inventory reports, you will use disk or tape. Such storage is relatively cheap. A diskette (called a "floppy disk" because they are flexible plastic) can store 90K or more of information on a 5¼ inch surface costing a mere four or five dollars. You can store the equivalent of about 50 typed pages of information on one such disk. Cassette tape is okay for home use and for back-up copies of your disk data, but is generally too slow for serious business or educational applications.

Service

Check out the service policy on your computer before you buy. Some manufacturers will exchange defective components, and others want to repair and return the original unit. If downtime is critical to you, choose the system which can be replaced in the shortest time. My 99/4 developed intermittent problems after more than a year of very heavy use, and TI exchanged it for a factory rebuilt unit for only $45.00 with same-day service and no questions asked. If trouble develops during warranty, the exchange charge is minimal. When I thought I had a defective disk system during the third month of ownership, the service center would have exchanged the entire disk system for about $3.50, but as it turned out, I had a bad diskette instead.

Where To Buy

Deciding where to buy your computer can be difficult. Should you buy from a local computer store, a department store, or perhaps from a mail-order outlet? You can get some terrific bargains from a mail-order firm. You'll see dozens of ads in any computer magazine and nearly all will accept credit cards making it very easy to buy. I saved nearly 40% on my TI machine buying from a firm in another city across the state; my Number 2 computer cost almost $500 less from out of state compared with the local store. The argument for buying from a local dealer and paying more is that you can count on better personal service if your machine goes on the blink. This may or may not be true depending on your dealer's integrity and quality of service. You could buy locally and still have problems with service. In my opinion, the overhead of the local computer store justifies the higher

Continued on p. 82

Memory . . . from p. 13

The PEEK subprogram reads bytes of CPU RAM data and copies the data directly into TI BASIC variables. For example, the statement:

CALL PEEK (8192,A,B,C(8))

reads three bytes of data from address 8192 and up, and assigns the values read to the variables A, B, and C(8).

The PEEKV subprogram reads bytes from VDP RAM. It works exactly like PEEK, except PEEKV accesses VDP RAM instead of CPU RAM.

The POKEV subprogram stores data values into VDP RAM. For example,

CALL POKEV(784,30,30,30)

writes the value 30 to VDP RAM locations 784, 785, and 786.

The CHARPAT subprogram reads a 16-character pattern identifier that specifies the pattern of a character code. For example,

CALL CHARPAT(68,D$)

places the pattern defining character code 68 in the string variable D$.

The three TI BASIC subprograms INIT, LOAD, and LINK interface assembly language programs and TI BASIC programs.

The INIT subprogram initializes the CPU memory for assembly language programs. The LOAD subprogram serves two purposes: it loads assembly language object files into CPU memory; and it loads data into CPU memory.

There are two forms of the LOAD subprogram. One form is used to load an object file from a storage device into memory and the second form is used to load data directly into CPU memory. For example, the statement

CALL LOAD ("DSK1. DEMO")

loads the file DEMO from the diskette in Disk Drive 1.

The second form of the LOAD subprogram is a POKE function for CPU RAM. For example, the statement

CALL LOAD (8197,85,40)

loads the value 85 into memory location 8197 and the value 40 into memory location 8198.

The LINK subprogram passes control and, optionally, a list of parameters from a TI BASIC program to an assembly language program. For example, the statement

CALL LINK ("PROG1",A,E(9))

passes control from a TI BASIC program to an assembly language program named PROG1 and passes the variables A and E(9) to the program.

Access to System Routines

The utility routines resident in the Mini Memory Command Module can be called from an assembly language program to access machine resources and interface with the TI BASIC interpreter. It's fair to warn you that the use of these routines requires a knowledge of the routines themselves and the organization of data used by the routines. You can get additional information about these routines from the Editor/Assembler owner's manual. (This is an excellent book, by the way. It's packed with inside information on the Home Computer's architecture.)

Two types of access programs are resident in the Mini Memory Command Module. One program contains a collection of system utilities with which to link to ROM/GROM routines, perform a keyboard scan, access the VDP, etc. The individual utility programs are classified as either a "Standard Utility" program or an "Extended Utility" program.

A second program contains TI BASIC interface utilities with which an assembly language program can access variables passed through a CALL LINK statement in a TI BASIC program. This program also contains an error-handling utility to return exceptions to a TI BASIC program.

Standard Utility Programs

Here's a list of the standard system utilities which become accessible with the Mini Memory Command Module.

- VDP Single Byte Write – write a single-byte value to a specified VDP RAM address.
- VDP Multiple Byte Write – write multiple bytes from CPU RAM to VDP RAM.

- VDP Single Byte Read – read a single byte from a specified VDP RAM address.
- VDP Multiple Byte Read – read multiple bytes from VDP RAM into CPU RAM.
- VDP Write to Register – write a single-byte value to any of the VDP RAM registers.
- Keyboard Scan – scan the keyboard and return a key-code and status. This routine can also read the position of the Wired Remote Controller.

Extended Utility Programs

Extended utilities are provided to access routines in the console GROMs and ROMs. These utilities are GPLLNK (link to GPL routines in GROM), XMLLNK (link to routines in ROM), and DSRLNK (link to Device Service Routines).

The Mini Memory Module manual (try to say that three times real fast) advises that you ought to know what you're doing before you access these routines.

GPLLNK Routines

The GPLLNK routines are as follows:

- Load Standard Character Set – load the standard character set into VDP RAM.
- Load Small Character Set – Loads the small character set (for the 40-column "Text Mode") into VDP RAM.
- Execute Power-Up Routine – Initializes the system as if the computer had just been turned on.
- Accept Tone – Issues an accepting tone for input.
- Bad Response Tone – Issues a bad-response tone warning.
- Bit Reversal Routine – Provides a mirror image of a byte of information. (This is often handy to form a mirror image of a character definition.)
- Cassette Device Service Routine – Accesses a cassette tape recorder/player as a storage device.
- Load Lower Case Character Set – Loads the lower-case character set into VDP RAM.

A number of floating point routines are also available through GPLLNK. Here they are:

- Convert a floating-point number to an ASCII string.
- Compute the greatest integer contained in a value.
- Raise a number to a specified power.
- Compute the square root of a number.
- Compute the inverse natural logarithm of a value.
- Compute the natural log of a number.
- Compute the cosine of a number.
- Compute the sine of a number.
- Compute the tangent of a number.
- Compute the arctangent of a number.

XMLLNK Routines

Routines in the console ROM can be accessed through the XMLLNK routine. The following routines can be called from an assembly language program using XMLLNK.

- Floating-point addition.
- Floating-point subtraction.
- Floating-point multiplication.
- Floating-point division.
- Floating-point compare.
- Floating-point stack addition.
- Floating-point stack subtraction.
- Floating-point stack multiplication.
- Floating-point stack division.
- Floating-point stack compare.
- Convert a string to a number.
- Convert a floating-point format number to integer.
- Push a value onto the value stack.
- Pop a value from the value stack.
- Convert an integer number to floating-point format.

Continued on p. 52

- VDP Single Byte Read – read a single byte from a specified VDP RAM address.
- VDP Multiple Byte Read – read multiple bytes from VDP RAM into CPU RAM.
- VDP Write to Register – write a single-byte value to any of the VDP RAM registers.
- Keyboard Scan – scan the keyboard and return a key-code and status. This routine can also read the position of the Wired Remote Controller.

Extended Utility Programs

Extended utilities are provided to access routines in the console GROMs and ROMs. These utilities are GPLLNK (link to GPL routines in GROM), XMLLNK (link to routines in ROM), and DSRLNK (link to Device Service Routines).

The Mini Memory Module manual (try to say that three times real fast) advises that you ought to know what you're doing before you access these routines.

GPLLNK Routines

The GPLLNK routines are as follows:

- Load Standard Character Set – load the standard character set into VDP RAM.
- Load Small Character Set – Loads the small character set (for the 40-column "Text Mode") into VDP RAM.
- Execute Power-Up Routine – Initializes the system as if the computer had just been turned on.
- Accept Tone – Issues an accepting tone for input.
- Bad Response Tone – Issues a bad-response tone warning.
- Bit Reversal Routine – Provides a mirror image of a byte of information. (This is often handy to form a mirror image of a character definition.)
- Cassette Device Service Routine – Accesses a cassette tape recorder/player as a storage device.
- Load Lower Case Character Set – Loads the lower-case character set into VDP RAM.

A number of floating point routines are also available through GPLLNK. Here they are:

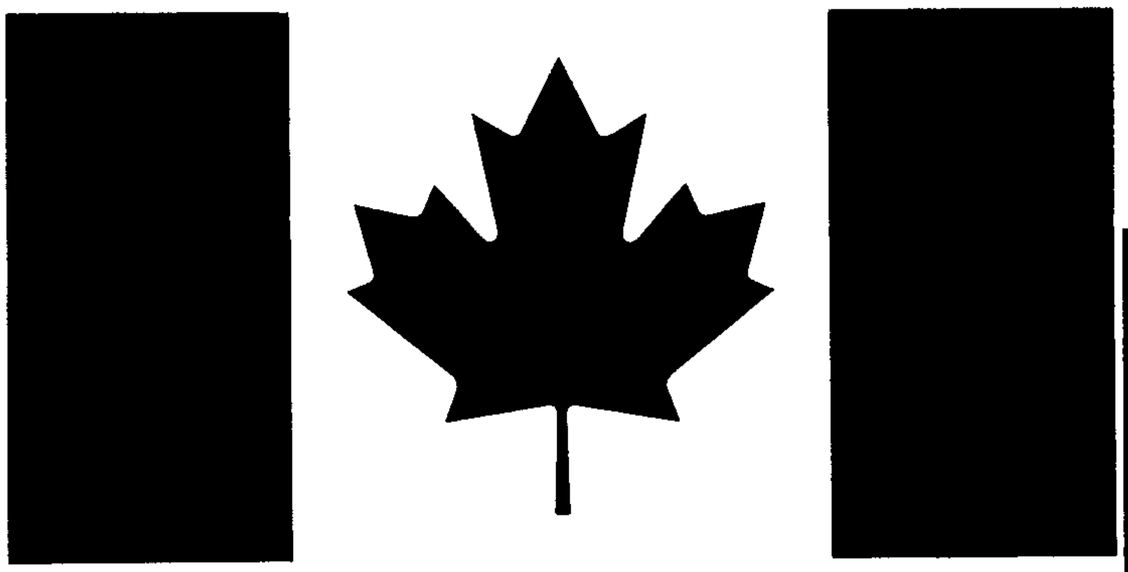
- Convert a floating-point number to an ASCII string.
- Compute the greatest integer contained in a value.
- Raise a number to a specified power.
- Compute the square root of a number.
- Compute the inverse natural logarithm of a value.
- Compute the natural log of a number.
- Compute the cosine of a number.
- Compute the sine of a number.
- Compute the tangent of a number.
- Compute the arctangent of a number.

XMLLNK Routines

Routines in the console ROM can be accessed through the XMLLNK routine. The following routines can be called from an assembly language program using XMLLNK.

- Floating-point addition.
- Floating-point subtraction.
- Floating-point multiplication.
- Floating-point division.
- Floating-point compare.
- Floating-point stack addition.
- Floating-point stack subtraction.
- Floating-point stack multiplication.
- Floating-point stack division.
- Floating-point stack compare.
- Convert a string to a number.
- Convert a floating-point format number to integer.
- Push a value onto the value stack.
- Pop a value from the value stack.
- Convert an integer number to floating-point format.

Continued on p. 52



CANADIANS!

TEXAS INSTRUMENTS
TI-99/4A Home Computer

CONSOLE
PLUS
rf modulator

This package consists of 1 PHC 004A Home Computer; 1 PHA 2100 RF Modulator; 1 PHA 2000 Cassette Cable; 1 PHA 2602 Basic Manual and 1 PHA 2603 Users Guide.

$399.95
CANADIAN FUNDS

DELIVERY AFTER JUNE 1 1982

DISK DRIVE
PLUS
controller

This package consists of 1 PHP 1850 Disk Memory Drive; 1 PHP 1800 Disk Drive Controller; 1 PHM 3019 Disk Manager Command Module and two Manuals.

$839.00
CANADIAN FUNDS

GAMES
IN COMMAND MODULES

PHM 3009	Football	
PHM 3018	Video Games I	
PHM 3023	Hunt the Wumpus	
PHM 3024	Indoor Soccer	
PHM 3025	Mind Challengers	
PHM 3030	A-Maze-Ing	
PHM 3031	The Attack††	
PHM 3032	Blasto††	
PHM 3033	Blackjack and Poker††	
PHM 3034	Hustle††	
PHM 3036	ZeroZap††	
PHM 3037	Hangman††	
PHM 3038	Connect Four††	
PHM 3039	Yahtzee††	
PHM 3052	Tombstone City	
PHM 3053	TI Invaders	
PHM 3054	Car Wars	

$39.95
CANADIAN FUNDS

DELIVERY AFTER JUNE 1 1982

PERIPHERALS

PHP 1500	Solid State Speech™ Synthesizer	125.95
PHP 1600	Telephone Coupler (Modem)	209.95
PHP 1700	RS-232 Accessories Interface ††	295.95
PHP 1800	Disk Drive Controller †† (One Disk Manager packed with each Disk Controller)	339.95
PHP 1850	Disk Memory Drive	499.95
PHP 1900	Solid State Printer	869.95
PHP 2200	Memory Expansion †† (32K RAM)	339.95
PHA 2100	R.F. Modulator (TV Adapter)	42.95
PHA 4100	10" Color Monitor	699.95

OPTIONAL ACCESSORIES

PHP 1100	Wired Remote Controllers (Pair)	19.95
PHA 1950	Thermal Paper (2 Pack)	12.95

APPLICATION PROGRAMS

Home Management/Personal Finance

Command Modules

PHM 3006	Home Financial Decisions	41.95
PHM 3007	Household Budget Management (Data storage system is required)	41.95
PHM 3012	Securities Analysis	69.95
PHM 3013	Personal Record Keeping (Data storage system is recommended)	59.95
PHM 3016	Tax/Investment Record keeping (Disk system is required)	84.95
PHM 3022	Personal Real Estate (Data storage system is recommended)	84.95
PHM 3044	Personal Report Generator (Data storage system is recommended)	59.95

Diskette

PHD 5001	Mailing List	29.95
PHD 5003	Personal Financial Aids	29.95
PHD 5021	Checkbook Manager	29.95
PHD 5022	Business Aids Library — Finance Management (Extended BASIC Command Module is required)	45.95
PHD 5024	Business Aids Library — Inventory Management (Personal Record Keeping or Statistics Command Module is required)	29.95
PHD 5027	Business Aids Library — Invoice Management (Personal Record keeping or Statistics Command Module is required)	29.95
PHD 5029	Business Aids Library - Cash Management (Extended BASIC Command Module is required)	59.95
PHD 5038	Business Aids Library — Lease/Purchase Decisions	29.95

Cassette

PHT 6003	Personal Financial Aids	19.95
PHT 6038	Business Aids Library — Lease/Purchase Decisions	69.95

ENTERTAINMENT

Command Module

PHM 3041D	Adventure (Pirate Adventure Diskette Game Included)	69.95
PHM 3041T	Adventure (Pirate Adventure Cassette Game Included)	69.95

Diskette

PHD 5002	TI-Trek (with speech)	29.95
PHD 5010	Mystery melody	29.95
PHD 5015	Oldies But Goodies — Games I	29.95
PHD 5017	Oldies But Goodies — Games II	29.95
PHD 5025	Saturday Night Bingo (Solid State Speech™ Synthesizer is required)	42.95
PHD 5037	Draw poker	33.95

Adventure Series

PHD 5043	Pirate Adventure	
PHD 5046	Adventureland	
PHD 5047	Mission Impossible	
PHD 5048	Voodoo Castle	
PHD 5049	The Count	
PHD 5050	Strange Odyssey	ALL 39.95
PHD 5051	Mystery Fun House	
PHD 5052	Pyramid of Doom	
PHD 5053	Ghost Town	
PHD 5054	Savage Island I & II	
PHD 5056	Golden voyage	

Cassette

PHT 6010	Mystery melody	21.95
PHT 6015	Oldies But Goodies — Games I	21.95
PHT 6017	Oldies But Goodies — Games II	29.95
PHT 6025	Saturday Night Bingo (Solid State Speech™ Synthesizer is required)	42.95
PHT 6037	Draw Poker	33.95

Adventure Series

PHT 6043	Pirate Adventure	
PHT 6046	Adventureland	
PHT 6047	Mission Impossible	
PHT 6048	Voodoo Castle	
PHT 6049	The Count	
PHT 6050	Strange Odyssey	ALL 39.95
PHT 6051	Mystery Fun House	
PHT 6052	Pyramid of Doom	
PHT 6053	Ghost Town	
PHT 6054	Savage Island I & II	
PHT 6056	Golden Voyage	

EDUCATION/PERSONAL ENRICHMENT

Comand Module

PHM 3002	Early Learning Fun	41.95
PHM 3003	Beginning Grammar	49.95
PHM 3004	Number Magic	33.95
PHM 3005	Video Graphs	41.95
PHM 3008	Video Chess	69.95
PHM 3010	Physical Fitness	49.95
PHM 3015	Early Reading† (Solid State Speech™ Synthesizer is required)	59.95
PHM 3020	Music Maker (Data storage system is recommended)	59.95
PHM 3021	Weight Control and Nutrition (Data storage system is recommended)	75.95
PHM 3027	Addition and Subtraction I† (Solid State Speech™ Synthesizer is recommended)	59.95
PHM 3028	Addition and Subtraction II† (Solid State Speech™ Synthesizer is recommended)	59.95
PHM 3029	Multiplication I† (Solid State Speech™ Synthesizer is recommended)	59.95
PHM 3040	TI LOGO (Memory Expansion is required)	49.95
PHM 3043	Reading Fun	59.95
PHM 3059	Scholastic Spelling — Level III† (Solid State Speech™ Synthesizer is required)	59.95
PHM 3060	Scholastic Spelling — Level IV† (Solid State speech™ Synthesizer is required)	59.95
PHM 3061	Scholastic Spelling — Level V† (Solid State Speech™ Synthesizer is required)	59.95
PHM 3062	Scholastic Spelling — Level VI† (Solid State Speech™ Synthesizer is required)	59.95
PHM 3064	Typing Tutor ††	59.95

Diskette

PHD 5007	Teach Yourself BASIC	47.95
PHD 5009	Music Skills Trainer	39.95
PHD 5011	Computer Music Box	29.95
PHD 5018	Market Simulation	29.95
PHD 5019	Teach yourself Extended BASIC (Extended BASIC Command Module is required)	39.95
PHD 5020	Music Maker Demonstration (Music Maker Command Module is required)	21.95
PHD 5023	Basketball Statistics (Extended BASIC Command Module is required)	39.95
PHD 5026	Bridge Bidding I	39.95
PHD 5030	Speak & Spell™ Program (Solid State Speech™ Synthesizer is required)	39.95
PHD 5031	Speak & Math™ Program (Solid State Speech™ Synthesizer and Terminal emulator II are required)	39.95
PHD 5039	Bridge Bidding II	39.95
PHD 5041	Bridge Bidding III	39.95
PHD 5042	Spell Writer (Terminal Emulator II Command Module and Solid State Speech™ Synthesizer are required)	39.95
PHD 5067	Teach Yourself beginning BASIC	39.95

Cassette

PHT 6007	Teach Yourself BASIC	39.95
PHT 6009	Music Skills Trainer	39.95
PHT 6011	Computer Music Box	21.95
PHT 6018	Market Simulation	24.95
PHT 6019	Teach yourself Extended BASIC (Extended BASIC Command Module is required)	29.95
PHT 6026	Bridge Bidding I	39.95
PHT 6031	Speak & Math™ Program (Solid State Speech™ Synthesizer and Terminal Emulator II are required)	39.95
PHT 6039	Bridge Bidding II	39.95
PHT 6041	Bridge Bidding III	39.95
PHT 6042	Spell Writer (Solid State Speech™ Synthesizer and Terminal Emulator II Command Module are required)	39.95
PHT 6067	Teach Yourself Beginning BASIC	39.95

OTHER APPLICATION PROGRAMS

Command Modules

PHM 3000	Diagnostic	41.95
PHM 3001	Demonstration	69.95
PHM 3011	Speech Editor (Solid State Speech™ Synthesizer is required)	69.95
PHM 3014	Statistics (Data storage system is recommended)	69.95
PHM 3026	Extended BASIC	124.95
PHM 3035	Terminal Emulator II	69.95
PHM 3055	Editor Assembler	124.95
PHM 3058	Mini-Memory	124.95

Diskette

PHD 5004	Programming Aids I	39.95
PHD 5005	Programming Aids II	29.95
PHD 5006	Math Routine Library	39.95
PHD 5008	Electrical Engineering Library	39.95
PHD 5012	Programming Aids III	29.95
PHD 5013	Graphing Package	29.95
PHD 5016	Structural Engineering Library	39.95

Cassette

PHT 6004	Programming Aids I	16.95
PHT 6006	Math Routine Library	39.95
PHT 6008	Electrical Engineering Library	39.95
PHT 6013	Graphing Package	19.95
PHT 6016	Structural Engineering Library	39.95
PHT 6044	SMU Circuit Analysis I	249.95

BUY ANY THREE COMMAND MODULES BEFORE JUNE 30 1982, AND TEXAS INSTRUMENTS WILL SEND YOU A MUNCH MAN COMMAND MODULE AT NO EXTRA COST.

WRITE FOR AVAILABILITY OF NEW TEXAS INSTRUMENTS ITEMS

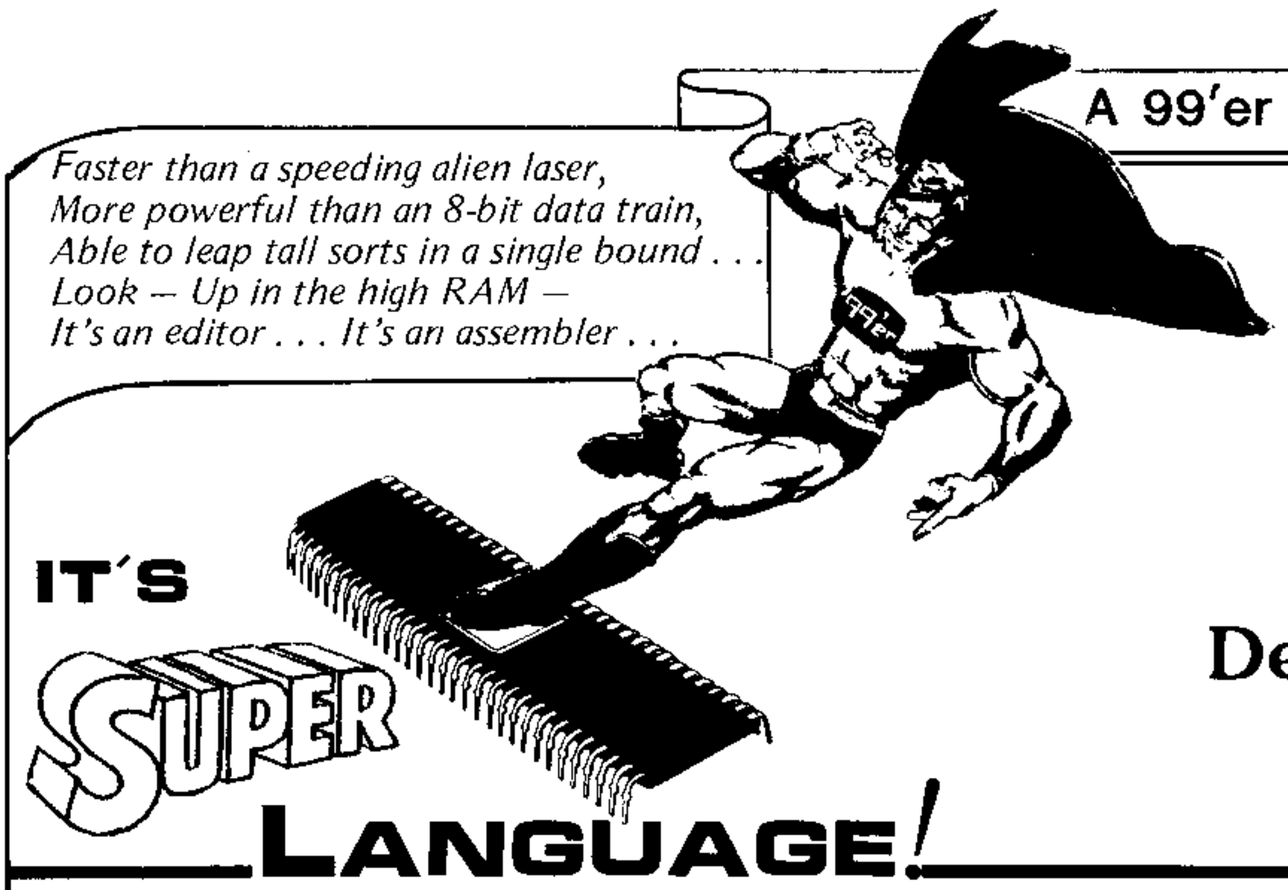
PAYMENT AND DELIVERY: All prices quoted are in Canadian Funds. For faster delivery, pay by Certified Cheque or Money Order (personal cheques take two weeks to clear). At these prices, we cannot afford Credit Card purchases. Add $5.00 postage and handling to all orders. We can provide any item on the Texas Instruments Canadian price list. All prices subject to change.

WRITE FOR PRICES ON ALL TI99/4A PRODUCTS AVAILABLE IN CANADA

403 426 2152

Canadian Micro Works

3724 91 Street Edmonton Alberta Canada T6E 5M3



A Screen Printing Utility

Part 1: Design Considerations

By Patricia Swift
Assembly Language Editor (The Human One)

One of the best features of the TI-99/4(A) computer is its graphics capability. The programmer can create a huge variety of screens by using the simple character-definition commands of TI BASIC. Wouldn't it be nice to dump those screens to your non-thermal printer? This two-part article presents a method for doing this. We will be using an Epson MX-100 printer, but many of the principles are applicable to any printer capable of bit-map (dot-addressable) graphics such as the MPI 88G, Okidata Microline series (with graphics option), and the EPSON MX-80 equipped with GRAPH-TRAX 80. Part I gives the theory behind the screen dump. In the next issue, Part II will give the Assembly Language subroutine itself.

I should mention that the 99/4A has an improved video processor (TMS9918A) which allows you to define up to 768 unique characters on the screen. However, this bit-map mode requires an extra 12K of memory to hold the larger tables needed. We'll limit ourselves to the Graphics I, or standard mode, in this discussion.

Approach—in English

The video screen contains 768 character positions, arranged in 24 rows of 32 characters. Each character is composed of an 8 x 8 dot matrix, giving you a screen of 256 x 192 dots. The screen dump program will reproduce the screen dot-for-dot on the printer.

With bit-image mode selected, the MX-100 prints "characters" which are one dot wide and 8 dots high. (See W.K. Balthrop's article, "From Dots to Plots", in the last issue of *99'er Magazine*.) Since the screen characters are also 8 dots high, each screen character can be represented by 8 MX-100 bit-image characters, for a total of 64 possible dots per screen character.

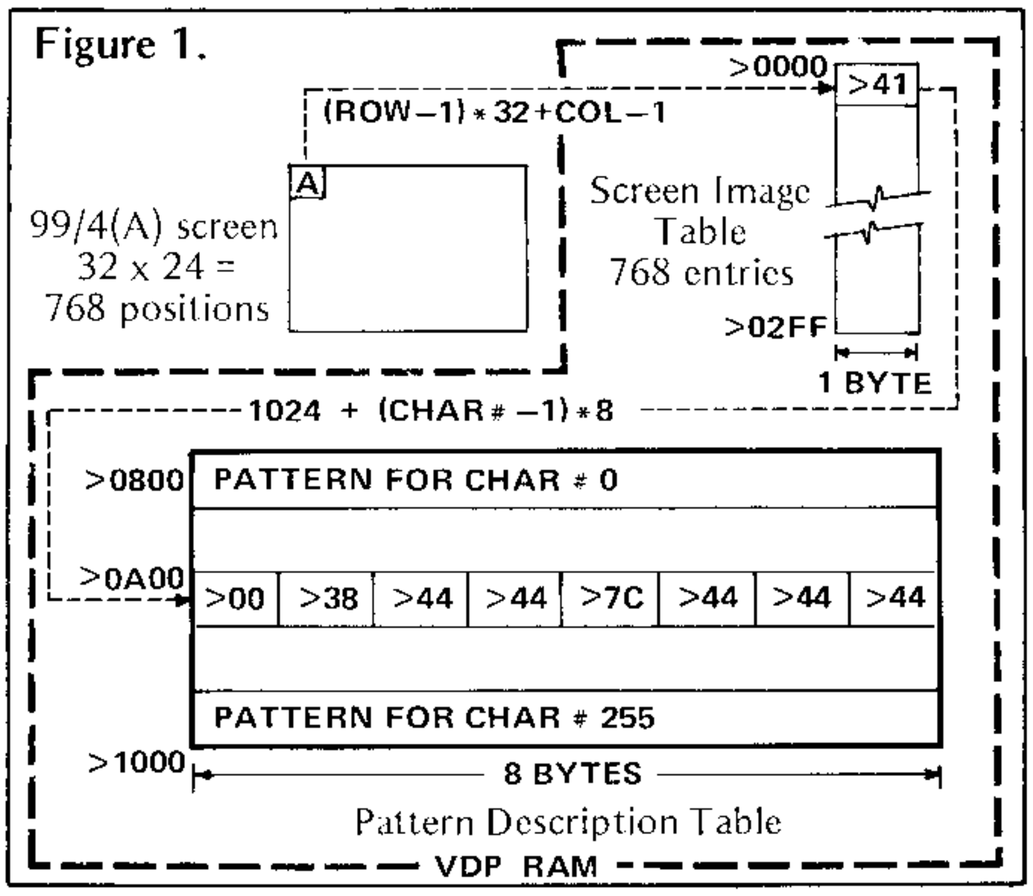
Accessing the Screen Image

The contents of the screen are stored in VDP RAM. Since we are not concerned with color here, only two of the screen tables in VDP RAM are of interest. The first is the Screen Image Table, which starts at default address >0000 and contains 768 bytes. Each byte corresponds to a character position on the screen, and contains the character number occupying that screen position. VDP RAM addresses >0000 through >001F correspond to the first row of screen characters, VDP RAM addresses >0020 through >003F correspond to the second screen row, and so on. Since each character number is contained in one byte, you can see that the character numbers must be between >00 and >FF, or decimal 0 through 255.

The second table we'll need is the Pattern Description Table, which starts at VDP RAM address >0800 by default. This table contains the dot patterns for each of the 256 characters which can be in use. The BASIC subprogram CHAR, which is used to define dot patterns for characters, stores patterns in this table. Since a character pattern takes 8 bytes to define, and there can be up to 256 different characters, the Pattern Description Table occupies 2048 bytes of VDP RAM.

Figure 1 shows the relationship between these two tables. For a given screen ROW and COLUMN, the VDP RAM address of the corresponding character number is given by (ROW-1) * 32 + COLUMN-1. Once you have obtained this character number, you can use it to index to the correct spot in the Pattern Description Table. The offset in this table is just (CHAR#-1) * 8, since each pattern description is 8 bytes long. Figure 1 shows an example of finding the pattern for the home postion (ROW 1, COLUMN 1) on the screen. The character number resides in the Screen Image Table at address 0. If the home character on the screen is "A", then VDP RAM address 0 contains the value 65 or > 41. The offset in the Pattern Description Table is then (65-1) * 8 = 512 or >200. Adding this to the base address of the Pattern Description Table, we get VDP RAM address >800 + > 200 = >0A00. The eight bytes starting at >0A00 in VDP RAM contain the pattern for the character "A". You can see that for our purposes, the contents of the Screen Image Table are just intermediate, though necessary, data. The character pattern is what we're really after.

Figure 1.



The 8-byte character pattern represents the dot pattern which appears on the screen in what I'll call "row-wise" form. The top portion of Figure 2 illustrates this for the character "A". The first byte of the pattern represents the first row of dots which comprise the character. The hexadecimal notation is just a shorthand way to group four bits at a time, with bits of value 1 standing for dots which are turned on in the character.

Translating the Characters to MX-100 Format

The Epson printer constructs its bit-image output in a different way. It uses what I'll call "column-wise" form. It still takes 8 bytes to produce the same character, but each byte of data passed to the printer represents a column (rather than a row) of dots in the finished character. The bottom of Figure 2 illustrates this. If we think of the character's dot pattern as an 8 x 8 matrix, then the translation from TI internal format to MX-100 (or MX-80) bit-image format is equivalent to transposing the matrix. We can't really treat each character pattern as a 64-bit matrix because 9900 Assembly Language does not have a BIT data type, but we can base the logic of the program on this idea.

Figure 2.

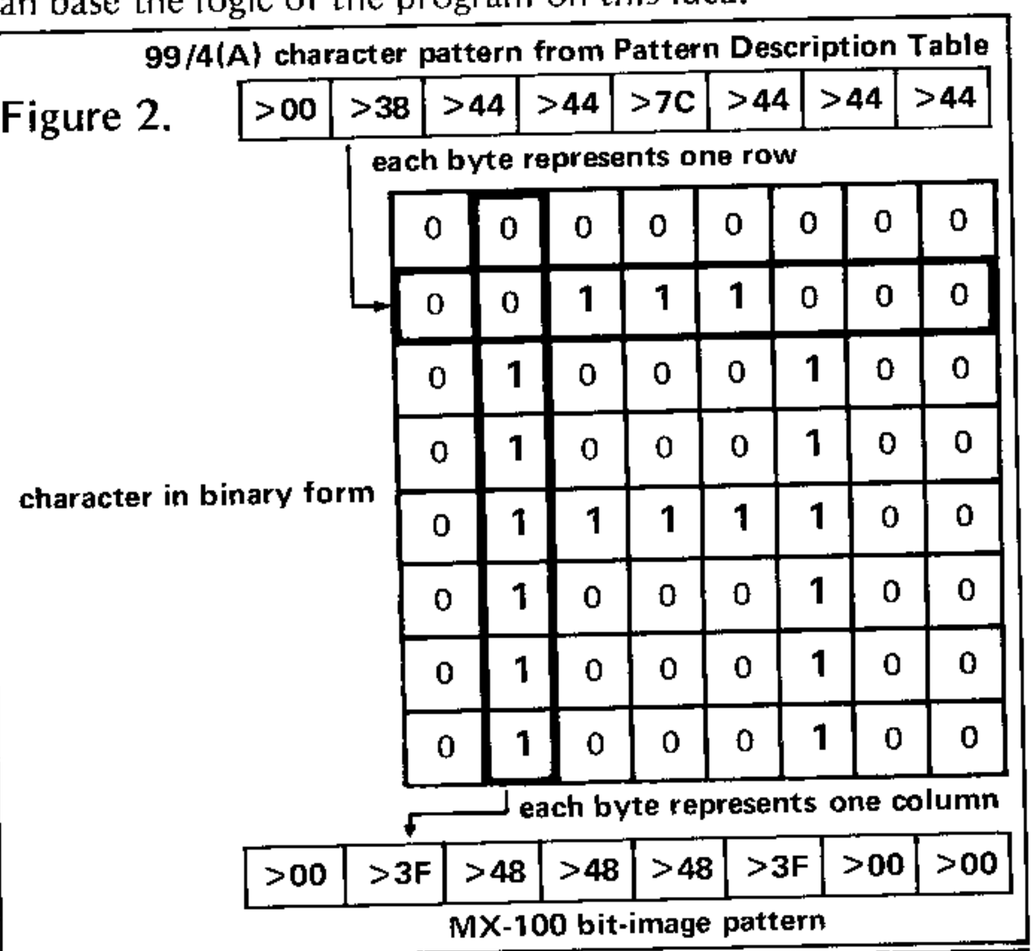
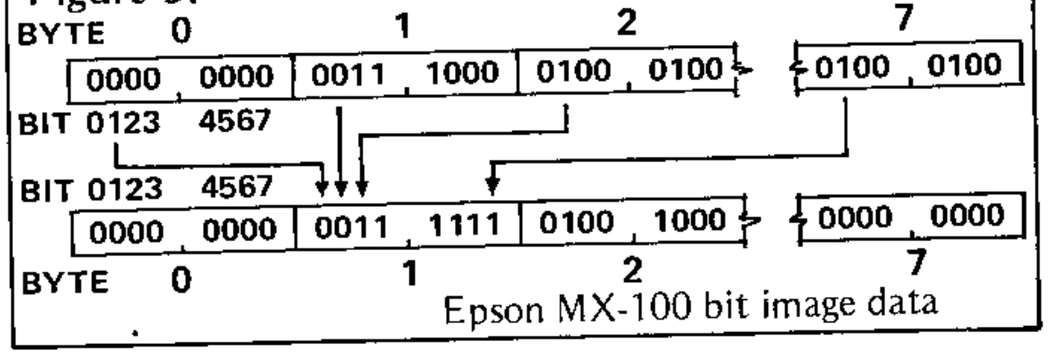


Figure 3. TI-99/4(A) character in binary form:



Program Outline

The screen dump program reads the Screen Image Table one byte at a time starting at the top (VDP RAM address 0). The value of each byte is used to calculate the position of the character pattern, and the 8-byte pattern is obtained from the Pattern Description Table. These 8 bytes will be manipulated to produce 8 bytes of information encoded for the MX-100 printer. Figure 3 shows how the bits of the 99/4(A) character pattern are rearranged to form bit-image data for the MX-100. Notice that the data at byte M, bit N is moved to byte N bit M—or transposed. The program will also have to send certain control characters for bit-image mode to the printer.

Next Time

Part II of this article will present a method of rearranging the bits in the character patterns, and show an Assembly Language implementation of these ideas.

Thank you for volunteering your interesting solution, Rick. One additional comment: When the contents of a disk will not be changed frequently, the following sequence of commands may be entered to obtain a LOAD file:

```
RUN "DSK1.LOADER"
NEW
MERGE DSK1.CAT
SAVE  DSK1.LOAD
```

The menu will then automatically appear upon entering Extended BASIC.

Dear Sir:

Your magazine couldn't be any better. Some programs published carry the "X'd out" disk. How do we store these programs for future use?

C. M. Katterjohn
Decatur, GA

These full-memory programs may be stored on cassette tape. Note: If you have a disk system, do not use it to SAVE segments of your type-in work for later dumping to cassette. The program might not RUN. It's safer to keep the disk controller off, and SAVE code segments directly to cassette tape.

Dear Sir:

As a joyful owner of the TI-99/4A computer, I would first like to compliment you on the *99'er Magazine.* I have found the articles to be very informative and, in many cases, useful for my purposes. Keep up the good work, and take a shot at J. C. Penny's to carry your magazine as they are a large retailer of the 99/4A system.

Now the reason I'm prompted to write you: Over the past couple of years I have acquired a rather extensive home remote control system from BSR Corporation—specifically, the X10 model. I read articles everywhere on how to make your home computer operate this marvelous system. In every case but *one,* the hardware is expensive ($100+) or requires a degree in electrical engineering to build. The one exception is made by Radio Shack and sells for $39.95 as a complete unit. It is packaged to connect directly to the cassette port of most TRS80 systems, includes two cassette tapes with the BASIC programs to operate the system under an 8 to 16K environment, and a printed listing of the program. I'm sorry to report that I bought this unit and could find no simple way to interface it to my 99/4A.

Interfacing this device seems to have a number of problems.

1. PEEK and POKE instructions are used.
2. Even if the 99/4A had PEEK and POKE I wouldn't know what to POKE or PEEK, and where . . . . .
3. The Radio Shack controller is sold as a black box and I am finding it difficult to get any kind of schematic prints for it.
4. Even if the TI cassette port could be used, the cables are incompatible.

Help . . . . . Is there a reader who might shed some light on my project or maybe you would be interested in this useful and popular system as an article in upcoming *99'er* issues.

Elden Kerr
Cupertino, CA

The TI-99/4A **does** *have the ability to PEEK and POKE if you have either the stand-alone Editor/Assembler Command Module or the Mini Memory Command Module. Also, with Extended BASIC and the Expansion Memory Peripheral you have the ability to PEEK and POKE into this additional memory, but not into the console's 16K RAM. What you suggest, Elden, would indeed make an interesting and useful article. Are there any readers out there who'd like to take the ball and run with it?*

Letters . . . from p. 7

variety of topics. Keep up the great work. (And I am looking forward with great anxiousness to the day when the *99'er* becomes a monthly.)

And speaking of praises, I would like to thank you for your kind words [regarding their *TI-ASTEROIDS* program] which appeared in your article *The Joys of Computer Gaming* (Vol. 1, No. 4).

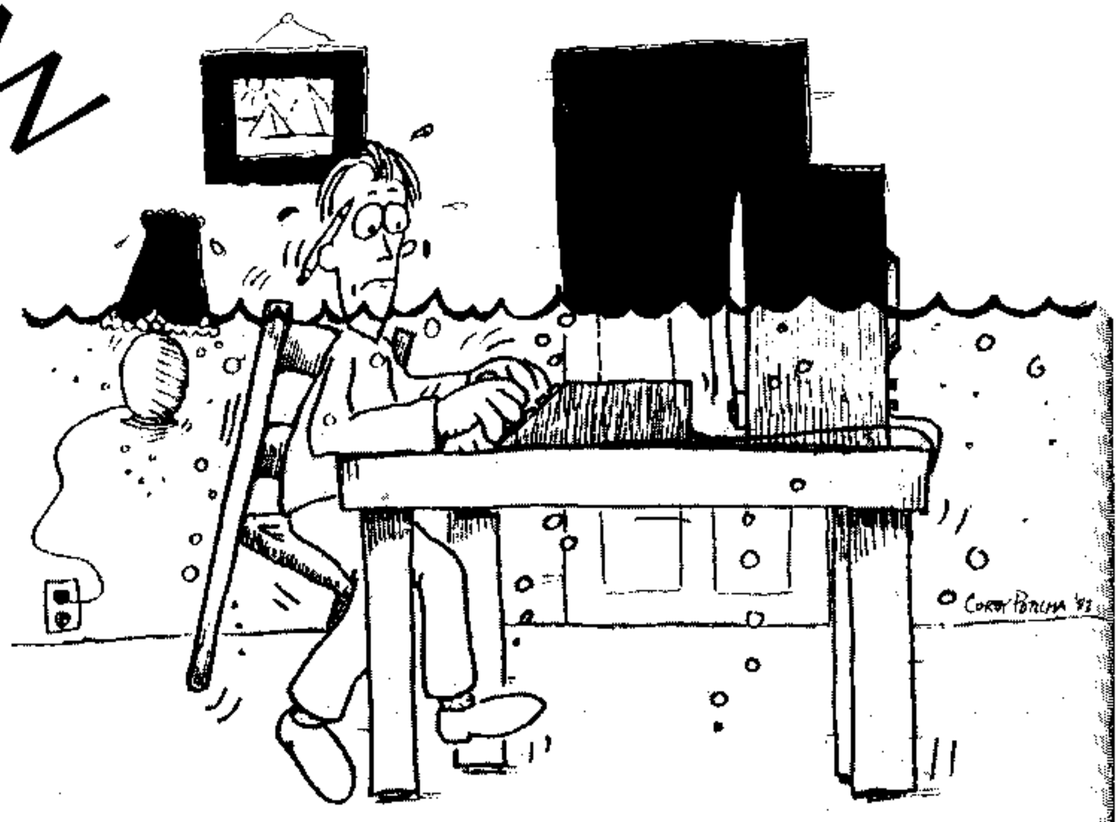
Rick Rothstein
FFF Software
Trenton, NJ

```
100 REM   ******************
110 REM   *                *
120 REM   *   POOR MAN'S   *
130 REM   *                *
140 REM   * PROGRAM LOADER *
150 REM   *                *
160 REM   *       BY       *
170 REM   *                *
180 REM   * RICK ROTHSTEIN *
190 REM   *                *
200 REM   ******************
210 REM
220 CALL CLEAR :: PRINT
    "PROGRAM STATUS.......WORKING" :: CL$="CLEAR" :
    : DIM A$(20):: OPEN #1:"DSK1.",INPUT ,
    RELATIVE,INTERNAL
230 DEF LN$(N)=CHR$(0)&CHR$(N)
240 DEF DI$(R)=CHR$(162)&CHR$(240)&CHR$(183)&CHR$
    (200)&CHR$(LEN(STR$(R)))&STR$(R)&CHR$(179)&CHR$
    (200)&CHR$(1)&STR$(COL)&CHR$(182)&CHR$(181)
250 DEF IF$(N)=CHR$(132)&"K@"&CHR$(190)&CHR$(200)
    &CHR$(2)&STR$(N)&CHR$(176)&CHR$(169)&CHR$(199)
    &CHR$(LEN(A$(I-64))+5)&"DSK1."&A$(I-64)
260 FOR I=0 TO 20
270 J=J+1 :: INPUT #1:A$(I),B,C,D :: IF I=0
    THEN 280 ELSE IF J>=127 OR LEN(A$(I))=0
    THEN 290 ELSE IF ABS(B)<>5 OR A$(I)="LOADER"
    THEN 270
280 NEXT I
290 CLOSE #1 :: EN$=CHR$(181)&CHR$(199)&CHR$(28)&
    "PRESS <ERASE> TO END PROGRAM"&CHR$(0):
    : COL=1 :: L=I-1 :: OPEN #2:"DSK1.CAT",
    VARIABLE 163
300 PRINT #2:LN$(1)&CHR$(157)&CHR$(200)&CHR$(5)
    &CL$&CHR$(0)
310 PRINT #2:LN$(2)&DI$(1)&CHR$(199)&CHR$(28)&
    "CATALOG"&RPT$(" ",12-LEN(A$(0)))&
    "DISKNAME-"&A$(0)&CHR$(0)
320 COL=9 :: FOR I=1 TO L :: PRINT #2:LN$(1+2)&DI$
    (12+I-INT(L/2))&CHR$(199)&CHR$(3+LEN(A$(I)))
    &CHR$(I+64)&"--"&A$(1)&CHR$(0):: NEXT I
330 PRINT #2:LN$(L+3)&CHR$(162)&CHR$(240)&CHR$(183)
    &CHR$(200)&CHR$(2)&"24"&CHR$(179)&CHR$(200)
    &CHR$(1)&"1"&CHR$(182)&CHR$(238)&EN$
340 PRINT #2:LN$(L+4)&CHR$(157)&CHR$(200)&CHR$(3)&
    "KEY"&CHR$(183)&CHR$(200)&CHR$(1)&"0"&CHR$(179)
    &"K@"&CHR$(179)&"S@"&CHR$(182)&CHR$(0)
350 PRINT #2:LN$(L+5)&CHR$(132)&"S@"&CHR$(190)&CHR$
    (200)&CHR$(1)&"0"&CHR$(176)&CHR$(201)&LN$(L+4)
    &CHR$(0)
360 FOR I=65 TO L+64 :: PRINT #2:LN$(L+I-59)&IF$(I)
    &CHR$(0):: NEXT I
370 PRINT #2:LN$(2*L+6)&CHR$(132)&"K@"&CHR$(190)
    &CHR$(200)&CHR$(1)&"7"&CHR$(176)&CHR$(157)&CHR$
    (200)&CHR$(5)&CL$&CHR$(130)&CHR$(139)&CHR$(0)
380 PRINT #2:LN$(2*L+7)&CHR$(134)&CHR$(201)&LN$(L+4)
    &CHR$(0):CHR$(255)&CHR$(255)::: CLOSE #2 :
    : DISPLAY AT(23,21)BEEP:"COMPLETE" :: END
```

OVERLAND FLOW

By Flavian Stellerine

134 Division St.
Trenton, NJ 08611



When rain falls on a surface, part of it passes into the soil (unless the surface is impervious such as concrete or asphalt) and the remainder disappears over a period of time either by evaporation or by runoff (overland flow) or by both. In most engineering drainage systems, the amount of water lost by evaporation is negligible; thus, drainage must be provided for all rainfall that does not infiltrate the soil or is not stored temporarily in surface depressions (lakes, swamp, etc.) within the drainage area. Until recently, the Rational Method was used for calculating design discharges for storm drains. This method has various drawbacks and is of limited applicability. The method used for this program is the Izzard dimensionless hydrograph. This method has been verified in laboratory tests and gives computed overland flow hydrographs agreeing closely with the measured hydrographs. The results of Izzard's method can be used by engineers in the design of drainage facilities for parking lots, airports, highways, etc. (See sample problem.)

Program Description

Input to the Overland Flow Program consists of two elements. The first consists of rainfall data and the second consists of physical characteristics.

Standard curves (see fig. 1) may be developed to express rainfall intensity-duration relationships with an accuracy sufficient for drainage problems. Rainfall intensity-duration data have been published by the National Weather Service.

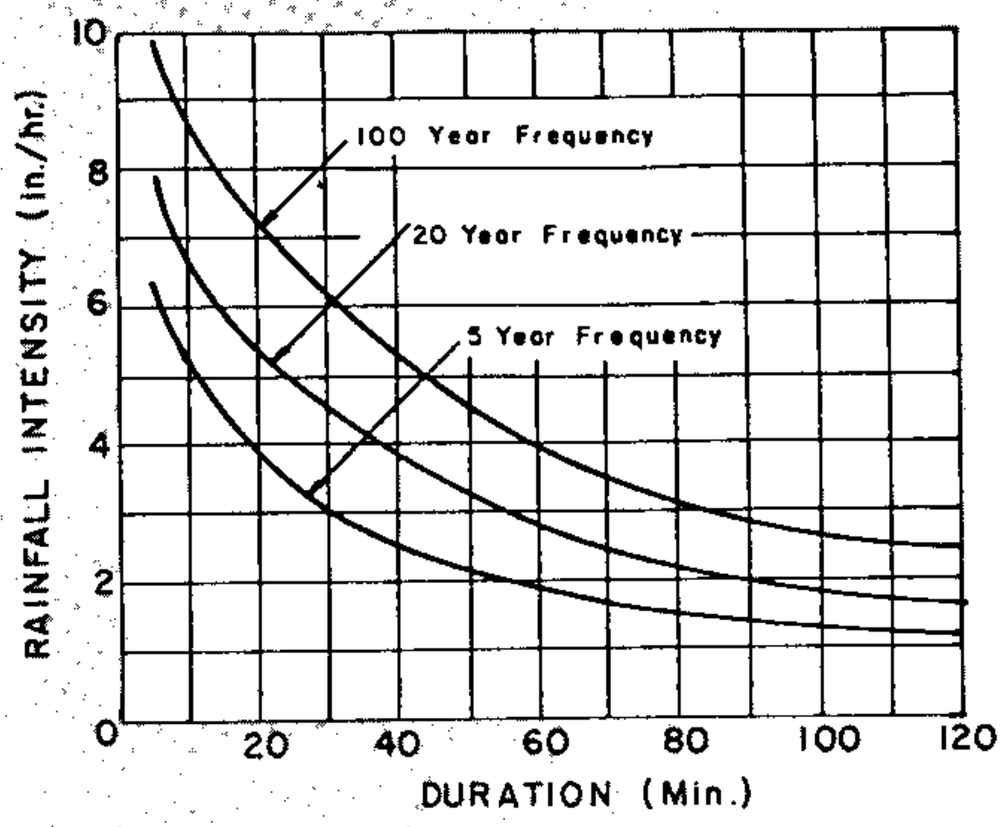


Figure 1.
Typical Intensity-Duration-Frequency Curve

The physical characteristics needed are; length, width, and slope of the area of interest, and a coefficient to describe the type of surface in question. The computer program contains a table from which the surface coefficient can be determined.

Output from the program can be displayed on the monitor screen or TI thermal printer. The program displays input data and the overland flow hydrograph in tabular and/or graphic format. The program has the capability of calculating and displaying two hydrographs at any one time. Thus, it is possible to vary the input data and compare the results.

Definition of Terms

Depression Storage:	Rainwater retained in puddles, ditches, and other depressions in the soil surface.
Equilibrium:	Occurs when the intensity of effective rainfall is equal to the outflow discharge. See Figure 2.
Equilibrium time (Te):	The time in minutes when the equilibrium condition is reached. See Figure 2.
Infiltration:	Passage of water through the soil surface into the soil.
Intensity:	Effective rainfall intensity in inches per hour. Effective rainfall is that which occurs after depression storage and infiltration capacities are met. See Figure 2.
Maximum Discharge:	The discharge, in cubic feet per second, when equilibrium is reached. See Figure 2.
Roughness Factor:	A coefficient that characterizes the resistance to flow of a particular surface type.
Length:	Distance, in feet, in the direction of slope, on which overland flow occurs. See Figure 3.
Slope:	See Figure 3.
Width:	Distance, in feet, perpendicular to the length, on which overland flow occurs. See Figure 3.

Operating Instructions

Step 1: Insert the cassette into a recorder, type: OLD CS1 and press ENTER. The computer then displays directions for loading the tape.

Step 2: When the cursor appears, type RUN, and press ENTER. When the title screen appears, press any key. Then select the screen or thermal printer as the device for output from the program.

Sample Problem 1

A parking lot 300 ft. long in the direction of the slope and 900 ft. wide has a tar and gravel pavement on a slope of .0025. Assuming a uniform rainfall intensity of 2.75 in/hr for 30 minutes, what is the maximum discharge that a gutter should be designed for.

Type RUN, then press ENTER.
Select the thermal printer as the output device.
Enter the following data:

Intensity	2.75
Duration	30
Length	300
Width	900
Slope	.0025
Roughness Factor	.017

Select option 1.

The gutter should be designed for a maximum discharge of 17.2 cfs.

DATA--HYDROGRAPH #1

GIVEN:

INTENSITY(IN/HR)=	2.75
DURATION(MIN)=	30
LENGTH(FT)=	300
WIDTH(FT)=	900
SLOPE(FT/FT)=	.0025
ROUGHNESS FACTOR=	.017

CALCULATED:

EQUILIB. TIME(MIN)=	19.52
MAX DISCHARGE(CFS)=	17.188

HYDROGRAPH #1

TIME(MIN)	DISCHARGE(CFS)
2.0	0.172
3.9	1.031
5.9	3.094
7.8	6.531
9.8	9.453
11.7	12.031
13.7	14.266
15.6	16.469
17.6	16.156
19.6	17.188
20.0	17.187
20.3	15.469
20.7	13.750
21.2	12.031
21.6	10.312
22.1	8.594
22.6	6.875
23.7	5.156
25.4	3.438
28.4	1.719
46.0	0.859
57.9	

Sample Problem 2

If the parking lot described in problem 1 is resurfaced with asphalt, what effect will this have?

After problem 1 is complete select option 3.
Enter the same data as for problem 1 with the exception of the roughness factor. Enter .007.
Select options 1 and 2.

DATA--HYDROGRAPH #2

GIVEN:

INTENSITY(IN/HR)=	2.75
DURATION(MIN)=	30
LENGTH(FT)=	300
WIDTH(FT)=	900
SLOPE(FT/FT)=	.0025
ROUGHNESS FACTOR=	.007

CALCULATED:

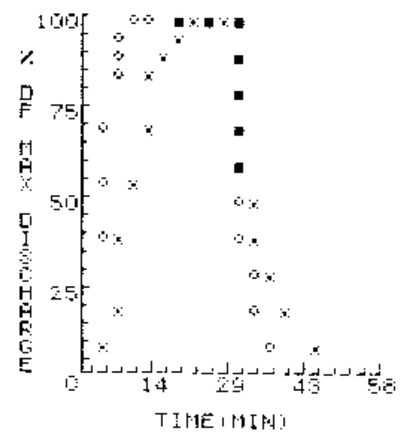
EQUILIB. TIME(MIN)=	9.2
MAX DISCHARGE(CFS)=	17.188

HYDROGRAPH #2

TIME(MIN)	DISCHARGE(CFS)
0.9	0.172
1.8	1.031
3.0	3.094
4.5	6.531
6.5	9.453
8.5	12.031
10.5	14.266
12.4	15.469
14.4	16.156
16.3	17.188
17.0	17.187
17.1	15.469
17.3	13.750
17.5	12.031
17.7	10.312
18.1	8.594
18.5	6.875
19.5	5.156
21.5	3.438
25.5	1.719
41.5	0.859

LEGEND

\ = HYDROGRAPH #1
0 = HYDROGRAPH #2
■ = COINCIDENCE OF 1 & 2
MAX DISCHARGE(CFS)= 17.188



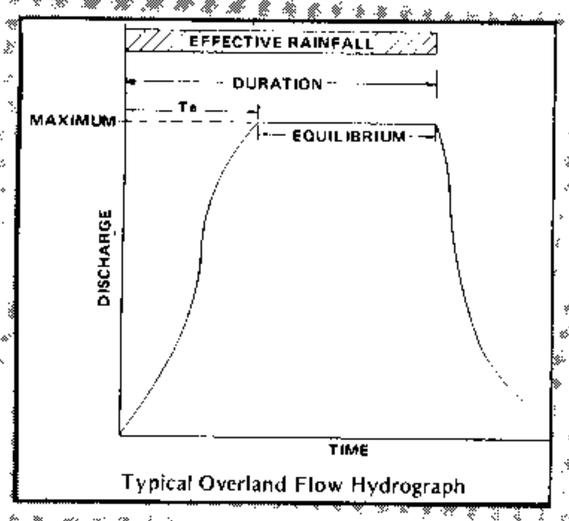
Step 3: After choosing the output device, the computer asks for the input data needed to compute the overland flow hydrograph. Type in the data requested and press ENTER.

Step 4: After all data is entered, the computer will generate a hydrograph and display the menu. Select one of the following options:

1. DISPLAY DATA (GIVEN AND CALCULATED)
2. DISPLAY HYDROGRAPH
3. COMPUTE ANOTHER HYDROGRAPH AND COMPARE
4. REDIRECT OUTPUT
5. ENTER NEW PROBLEM
6. END PROGRAM

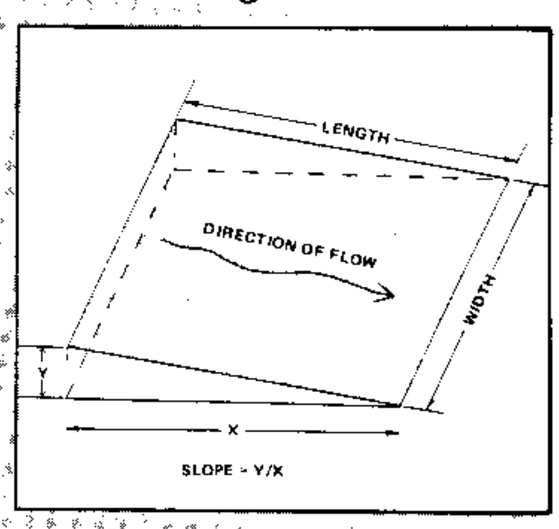
After completing any of options 1 through 5 the computer returns to the menu.

Figure 2.



Typical Overland Flow Hydrograph

Figure 3.



OPTION 1: DISPLAY DATA (GIVEN AND CALCULATED) – If you select option 1, the computer will display the input data that you entered and also the calculated values for equilibrium time and maximum discharge.

OPTION 2: DISPLAY HYDROGRAPH – If you select option 2, the computer asks you if you want the hydrograph displayed in table or graphic form or both. The graphic form plots the hydrograph points as percent of maximum discharge versus time. When two hydrographs are plotted the maximum discharge is the greater of the two hydrograph maximums.

OPTION 3: COMPUTE ANOTHER HYDROGRAPH AND COMPARE – If you select option 3, the computer asks you to enter another set of data in order to calculate another hydrograph. Since the first hydrograph is retained by the computer, this option can be used to vary any of the input data and examine the result (see sample problems). This option can be used as many times as the user wishes. The computer always compares to the original hydrograph computed when the program was initially run. If a subsequent hydrograph is preferred to the original, select option 5 and enter the new hydrograph as the original. Thus, all other hydrographs computed via option 3 will be compared to the new hydrograph.

OPTION 4: REDIRECT OUTPUT – If you select option 4, you change the device to which the output is displayed.

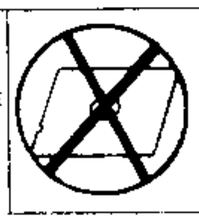
OPTION 5: ENTER NEW PROBLEM – If you select option 5, the program begins again. This option is used to rerun the program without having to type RUN. Also, use this option in conjunction with option 3 to compare several hydrographs and select one that is best suited to the problem.

OPTION 6: END PROGRAM – This option returns the computer to TI BASIC.

Program listings begin on p. 30

EXPLANATION OF THE PROGRAM
Overland Flow

Line Nos.	
160-530	Program initialization: Character assignments and array dimensioning.
540-1080	Data entry.
1090-1490	Calculation of Overland Flow Hydrograph.
1500-1800	Display hydrograph, in tabular form, on video monitor or TI Thermal printer.
1810-1990	Display menu and go to portion of program according to option selected.
2000-3200	Display hydrograph, in graphic form, on video monitor or TI Thermal printer.
3210-3460	Subroutine to align numbers on display.
3470-3590	Display given and calculated data.
3600-3710	Prepare program to accept and calculate a second hydrograph.
3720-3790	Subroutine to blank and restore screen when displaying information on video monitor.
3930-4220	Scale and label axes of graph.
4230-4250	Common subroutine to check keyboard entry.
4260-4510	Select drive for program output.



If using Thermal Printer; otherwise CALL FILES(1) will allow RUNing.

```
100 REM  ******************
110 REM  * OVERLAND FLOW *
120 REM  ******************
130 REM  BY FLAVIAN STELLERINE
140 REM  99'ER VERSION 1.5.1
150 REM
160 CALL CLEAR
170 CALL SCREEN(15)
180 GOSUB 3720
190 PRINT TAB(8);"OVERLAND FLOW";:TAB(14);"BY";:TAB(6);
    "FLAVIAN STELLERINE";;;;;;;;;;
200 PRINT TAB(4);"PRESS ANY KEY TO BEGIN";
210 CALL SOUND(150,600,1)
220 GOSUB 3760
230 GOSUB 4230
240 CALL CLEAR
250 CALL SCREEN(8)
260 PRINT TAB(9);"INITIALIZING";;;;;;;;;;;;
270 OPTION BASE 1
280 DIM H(2,2,22),IN(2),DU(2),LE(2),WI(2),SL(2),CR(2),TE(2),QW(2),
    QE(2),TPP(2,2)
290 DEF RD(X)=INT(100*X+.5)/100
300 DEF RD2(X)=INT(1E3*X+.5)/1E3
310 DEF RD4(X)=INT(1E4*X+.5)/1E4
320 DEF RD6(X)=INT(1E6*X+.5)/1E6
330 RESTORE 380
```

```
340 FOR I=1 TO 42
350 READ CODE,CH$
360 CALL CHAR(CODE,CH$)
370 NEXT I
380 DATA 99,E080808081818FF,100,E0808080808080,101,F880808080808080,
    102,00000000010101FF
390 DATA 103,0000010101010FF,104,60909060,105,06090906,106,
    0000000060909060,97,000024181824
400 DATA 107,0000000006090906,111,90606090,112,09060609,113,
    0000000090606090,96,0018242418
410 DATA 114,0000000009060609,121,F0F0F0F0,122,96696996,123,
    9060609060909060,64,FFFFFFFFFFFFFFFF
420 DATA 124,9060609006090906,131,69969669,132,0F0F0F0F,133,
    0906060960909060
430 DATA 134,0906060906090906,141,6090906090606090,142,0609090690606090,
    143,00000000F0F0F0F0
440 DATA 144,0000000096696996,151,6090906009060609,152,0609090609060609,
    153,0000000069969669
450 DATA 154,000000000F0F0F0F,155,007840404444FF,156,00F0404040404040,
    157,00F8404040404040,93,0906
460 DATA 158,000000000404FF,159,000004040404FF,145,,98,00003C3C3C3C,91,
    0000000000006090,92,91620408102046B9F
470 GOSUB 4260
480 HY=1
490 FOR I=1 TO 2
500 FOR J=1 TO 22
510 H(1,I,J)=0.0
520 NEXT J
530 NEXT I
540 CALL CLEAR
550 CALL SCREEN(8)
560 PRINT " HYDROGRAPH #";STR$(HY):
570 PRINT " ENTER DATA:"::
580 INPUT "  INTENSITY(IN/HR) ":IN(HY)
590 IN(HY)=RD2(IN(HY))
600 IF IN(HY)<1E4 THEN 630
610 GOSUB 4470
620 GOTO 580
630 PRINT
640 INPUT "  DURATION(MIN)    ":DU(HY)
650 DU(HY)=RD(DU(HY))
660 IF DU(HY)<1E5 THEN 690
670 GOSUB 4470
680 GOTO 640
690 PRINT
700 INPUT "  LENGTH(FT)       ":LE(HY)
710 LE(HY)=RD(LE(HY))
720 IF LE(HY)<1E6 THEN 750
730 GOSUB 4470
740 GOTO 700
750 PRINT
760 INPUT "  WIDTH(FT)        ":WI(HY)
770 WI(HY)=RD(WI(HY))
780 IF WI(HY)<1E6 THEN 810
790 GOSUB 4470
800 GOTO 760
810 PRINT
820 INPUT "  SLOPE(FT/FT)     ":SL(HY)
830 SL(HY)=RD6(SL(HY))
840 IF SL(HY)<=.04 THEN 910
850 CALL SOUND(200,120,1)
860 PRINT :" FOR ACCURATE RESULTS THE";" VALUE FOR SLOPE SHOULD";" BE
    LESS THAN 0.04.":
870 PRINT " DO YOU WISH TO RE-ENTER";" A VALUE FOR SLOPE?(Y/N)"::
880 GOSUB 4230
890 IF (KEY=89)+(KEY=78)=0 THEN 880
900 IF KEY=89 THEN 820
910 PRINT
920 FOR I=1 TO 350
930 NEXT I
940 CALL CLEAR
950 PRINT "   ENTER ROUGHNESS FACTOR":
960 PRINT "   USING TABLE AS A GUIDE"::
970 PRINT "SURFACE TYPE          FACTOR":
980 PRINT "------------          ------"
990 PRINT "SMOOTH ASPHALT PAVE..0.0070"::"TAR AND SAND PAVE....0.0075"::
    "CRUSHED-SLATE PAPER..0.0082"::
1000 PRINT "CONCRETE PAVEMENT....0.0120"::"TAR & GRAVEL PAVE....0.0170"::
1010 PRINT "CLOSELY CLIPPED SOD..0.0260"::"DENSE BLUEGRASS TURF.0.0600":::
1020 INPUT "   ROUGHNESS FACTOR= ":CR(HY)
1030 CR(HY)=RD4(CR(HY))
1040 IF CR(HY)<1 THEN 1070
1050 GOSUB 4470
1060 GOTO 1020
1070 FOR I=1 TO 250
1080 NEXT I
1090 CALL CLEAR
1100 CALL SCREEN(3)
1110 PRINT "          COMPUTING":::::::::::
1120 QE(HY)=IN(HY)*LE(HY)/43200
1130 KX=((0.0007*IN(HY))+CR(HY))/SL(HY)^(1/3)
1140 DE=KX*LE(HY)*QE(HY)^(1/3)
1150 TE(HY)=DE/(30*QE(HY))
1160 KNT=0
1170 FOR J=1 TO 10
1180 IF (J/10*TE(HY))>DU(HY)THEN 1190 ELSE 1210
1190 KNT=J
1200 GOTO 1250
1210 H(HY,2,J)=J/10*TE(HY)
1220 NEXT J
1230 H(HY,2,11)=((DU(HY)-TE(HY))*0.5)+TE(HY)
1240 H(HY,2,12)=DU(HY)
1250 QW(HY)=QE(HY)*WI(HY)
1260 RESTORE 1270
1270 DATA .01,.06,.18,.38,.55,.7,.83,.9,.94,1,1,1
1280 FOR J=1 TO 12
1290 IF (J>KNT)+(KNT<>0)=-2 THEN 1340
1300 READ MULT
1310 H(HY,1,J)=MULT*QW(HY)
1320 NEXT J
1330 H(HY,1,12)=QW(HY)
1340 IF KNT=0 THEN 1360
1350 QW(HY)=H(HY,1,KNT-1)
1360 DO=(CR(HY)/SL(HY)^(1/3))*LE(HY)*QE(HY)^(1/3)
1370 FOR I=1 TO 9
1380 H(HY,2,I+12)=(DO/(120*QW(HY)/WI(HY)))*(((1-I/10)^(-2/3))-1)
1390 H(HY,2,I+12)=H(HY,2,I+12)+DU(HY)
1400 H(HY,1,I+12)=(1-I/10)*QW(HY)
1410 NEXT I
1420 H(HY,2,22)=(DO/(120*QW(HY)/WI(HY)))*6.36B064+DU(HY)
1430 TPP(HY,1)=0
1440 TPP(HY,2)=0
1450 H(HY,1,22)=.05*QW(HY)
1460 IF KNT<>0 THEN 1810
1470 TPP(HY,1)=.25*(DU(HY)-TE(HY))+TE(HY)
1480 TPP(HY,2)=.75*(DU(HY)-TE(HY))+TE(HY)
1490 GO TO 1810
1500 CALL CLEAR
1510 IF FILE=0 THEN 1530
```

Note to Programmers:
The techniques used in this program are not limited to hydrology. You will observe a good example of plotting graphs, scaling the parameters for the graph, and plotting two different graphs with possible coincident points.

Continued on p. 85

GETTING DOWN TO BUSINESS

By George Struble

RANDOM ACCESS: When Random Does Not Mean By Chance!

Random-access files are extremely important in any conversational application that requires a data base of some kind. This includes any kind of *business* information system, but also includes a lot of others as well. But unfortunately, the concept of what "random access" actually is frequently gives rise to misunderstandings and even fear--that is, the fear that using random access is too complex to be attempted. In this article I will try to correct some of the misunderstandings, and start to show you how to use random-access files.

The dictionary I took to college told me that random meant "going, made, occuring, etc., without definite aim, purpose, or reason." Synonyms given are haphazard, chance, casual, aimless. Thus, when I first heard of random access in reference to computer data, it didn't sound like anything I would want. The good people didn't *mean* haphazard or chance, or any of those other things; they meant access *directly* to a piece of data specifically wanted, *without* having to pass sequentially by a lot of other unwanted data to get there. To me, this is much better described by the phrase "direct access," and I have been using direct access and talking against the term "random access" for years. But enough. The terminology *random-access* appears in my newer dictionary, and is generally accepted in computer circles as "permitting access to stored data in any order the user desires." From the standpoint of the storage unit, access *is* random in the older sense, since the sequence of access requests is not at all predictable (compared with sequential access, where access is entirely predictable). So this is the point—direct (I still like that word) access to whatever data we want, in any sequence.

Why is this important? Suppose you are using an inventory system. You have a transaction for product 539. Your last transaction was for product 762. What must you do to retrieve, update, and rewrite the record for product 539? If your inventory file is an ordinary sequential file, you must start at the beginning of the file, reading all the records up to product 539, rewriting each to a new file. Impossibly slow, yet it gets worse: After you do your thing with product 539, you either have to finish copying the rest of the records to the new file, or postpone that, hoping the next transaction will be for a product after 539 so we can save a trip through the whole file. What we clearly need is the ability to go directly to record 539, read it, and write the updated record back in the same place. Direct-access files permit you to do just that, and the savings in time are what make a data-based system feasible—not only for inventory, but for accounts payable or receivable, general ledger, etc.

Implementation in TI BASIC

In a random-access file, in TI BASIC and in every other system I know, all records must be the same length. The operating system knows the length of each record, knows where the file begins on disk, and therefore can calculate the exact location of the 367th record, or any record. This calculation is used whenever we ask to read or write a particular record.

Let's look at the statements we use on random-access files. They are the same statements we use on ordinary (sequential) files, but some parameters are different. First, when we OPEN a random-access file, we must declare:

- file organization is RELATIVE
- file type is DISPLAY or INTERNAL
- open mode is INPUT, OUTPUT, or UPDATE
- record type is FIXED.

Don't ask why the word RELATIVE is chosen to specify random access, but it may have something to do with the address calculation: the location of each record is computed relative to the beginning of the file. You may well want to construct your random-access files as INTERNAL, to save space and time required for converting DISPLAY (ASCII) files for internal use. An INTERNAL file cannot be listed directly, but you probably need a program to list a random-access file anyway. An "open mode" of UPDATE allows you to read and write records in your file, and this is what you want most of the time. UPDATE is also the default if you don't specify an open mode. As you specify FIXED record type, you may specify the record length too, and I recommend that you do. As an example,

```
OPEN #1:"DSK1.INVENTORY",
RELATIVE,INTERNAL,UPDATE,
FIXED 92
```

opens the INVENTORY file on your DSK1 as your #1 file; the file has 92-byte records in internal format, for random-access reads and writes. When you first create a file, you can and should specify the number of records to be allocated initially; the number follows the word RELATIVE. For example, the program that first established this file could have used

```
OPEN #2:"DSK1.INVENTORY",
RELATIVE 150,INTERNAL,
OUTPUT,FIXED 92
```

To read a particular record, include the record number (the first record is numbered zero) in the INPUT statement; if N = 119, for example,

```
INPUT #2,REC N: PN,D$,Q,PR
```

reads the 119th record from the file into the variables PN, D$, Q, PR. The PRINT statement similarly includes the

Continued on p. 54

About the Author

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

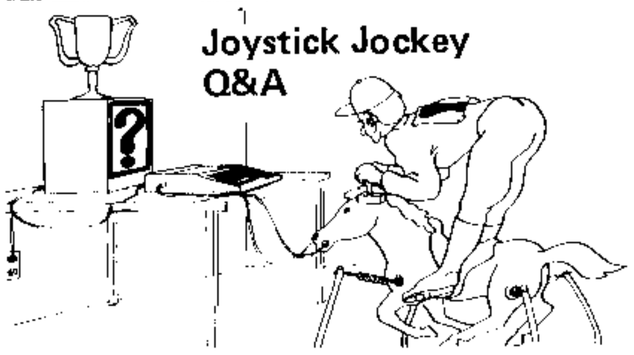
COMPUTER GAMING™

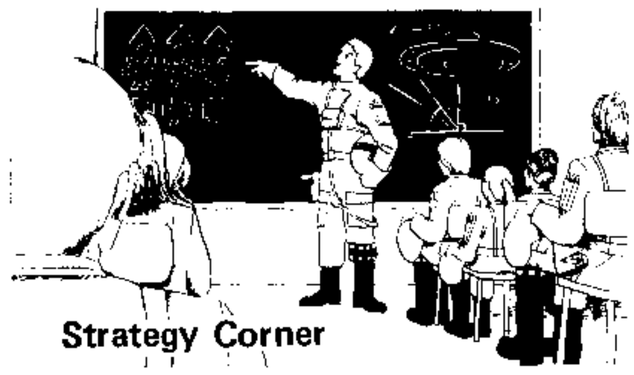
Vol. 1, No. 1
$2.00 in U.S.A.

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 . . .





An Interview with John C. Plaster

Designer and Programmer of "Tombstone City: 21st Century"

By Gary M. Kapla

GMK: *Have you programmed many games?*

JCP: Tombstone City was the first. I've since programmed one other, and started work on a third.

GMK: *What experiences have you had that you think contributed to your desire and ability to design computer games?*

JCP: Nothing special, really. I just had a regular childhood . . . played things like checkers, Old Maid, and poker. I never had played any arcade games before I designed *Tomstone City.*

GMK: *Are you an artistic or creative person in other fields or activities?*

JCP: I don't know that I'm that creative in other areas. I have a pure math background—that's what I majored in in school . . . so maybe I have more of a logical-type mind than a creative mind.

GMK: *What jobs have you had before coming to work at TI?*

JCP: I guess I've only had three jobs in my life. My first job was working for my dad, a farmer in Canyon, Texas. I worked with him on the farm up until the time I went to college. I got my B.A. at Rice University, and spent a year at the University of Arizona doing graduate work. I completed my Masters and PhD in math at Texas Tech. That was where I worked as a TA [teaching assistant]. I also spent one semester at Abilene Christian University.

GMK: *How did you start getting involved in computer games, and where did the idea for Tombstone City come from?*

JCP: I just started playing around in 9900 [Assembly Language for the TI-99/4A] code. I really wasn't intending to write a game when I started . . . I just drew a spaceship on the screen and played with it. I was probably influenced by the fact that most of the arcade games I was familiar with were space oriented. As far as the background of *Tombstone*, I spent a year at the University of Tucson in Arizona. The saguaro cactus found in the game comes from my living in Arizona.

GMK: *How about specifics of the game . . . how did these develop?*

JCP: It was really what you'd consider spontaneous. I started putting the game together in April, and got a working version of it in May. There really wasn't a whole lot of forethought to it. I started putting it on the screen as fast as I thought of it. So, from the conception of the idea, to completion of the basics, we're talking about a four-week period.

GMK: *From that whirlwind timetable, it would appear that you have had quite a bit of experience in 9900 Assembly Language coding.*

JCP: Actually not. I first started on 9900 coding about two weeks before I began the game, and that was because of another project—a numeric expression interpreter. Before that, I was involved in two major projects. The first was the *Personal Real Estate* done in GPL [Graphic Programming Language]; the second was also written in GPL, and that was the Milliken Math Series.

GMK: *Did you find that programming in GPL was easier or more difficult than in 9900 Assembly Language?*

JCP: Some things are easier in GPL, but overall, it's probably easier to learn and program in 9900 code. 9900 to me is very straightforward, easy-to-learn Assembly Language. It was easier than the IBM Assembler I learned in college. I don't think a person would really have any problem learning it. You have to get down the four or five basic routines that allow you to interface with the screen and sound. That sort of thing, I think, is fairly well explained in the Editor/Assembler manual. Once you've got that down, then it is a very straightforward process.

GMK: *What steps were involved in the actual development of the game?*

JCP: First, I very quickly started putting things up on the screen. I was thinking that if you start with a spaceship, then you've got to have something attacking the ship . . . so I next put a couple of monsters on the screen. Then once you have the monsters, you've got to determine how they are going to be generated. That was probably the main element in the game—how the monsters were to be generated. One of the hurdles I had to overcome was keeping track of the monsters on the screen while generating new monsters. I kept track of the monsters in a linked-list type structure, and because of the limitations in CPU memory, I had to restrict the number of monsters that could appear on the screen to about nine. Then I had to resolve how the monsters would actually be generated. I ended up doing this with a screen search, where every so often the screen is actually searched to find a generating pair of saguaro cactus.

GMK: *Were there any other major hurdles in the game's design?*

JCP: That was about it, except for how the spaceship actually started out on the screen. This was another design element that had to be resolved. That's how the "safe area" [inside the grid] came about.

GMK: *Was your game design seriously constrained by any memory limitations imposed on you?*

JCP: No, the game is really a pretty simple model. It's not a very complicated game, and I didn't utilize all of the system capabilities in it. I didn't really come up with a feature that I wanted to implement but couldn't because of the system. Everything that I wanted I was able to implement with very little trouble. The only thing I could call a limitation was the amount of CPU memory, but the limits that were placed on me were probably good anyway.

GMK: *How so?*

JCP: We only have about 180 bytes of usable CPU memory. This is where I kept my linked list for my monsters. Because of its size, I was limited to maybe nine monsters. I said that was probably good meaning that if I had fifteen monsters that would have probably made the game too difficult. As far as chip utilization goes, the game has one GROM and one ROM inside the Command Module. The GROM chip was used for the graphics, and the ROM chip for the 9900 code—the actual programming of the game. I used all the memory available in the ROM, and had quite a bit left over in the GROM. If I needed to, I could have put some of the data I had in the ROM into the GROM.

GMK: *Did the basic game scenario change any as the programming evolved?*

JCP: First of all, there wasn't any real scenario at the beginning. It was just a capture-type game—monsters attack ships, and ships shoot monsters. When the game was just about completed, a full scenario was developed to fit the game elements. Management really had some problems rationalizing my Western-type background with a space theme. At first, my scenario consisted of an old ghost town being taken over by the government for a nuclear power site, with the inevitable nuclear accident oc-

Continued on p. 47

GAMEWARE BUFFET

Tex-Thello

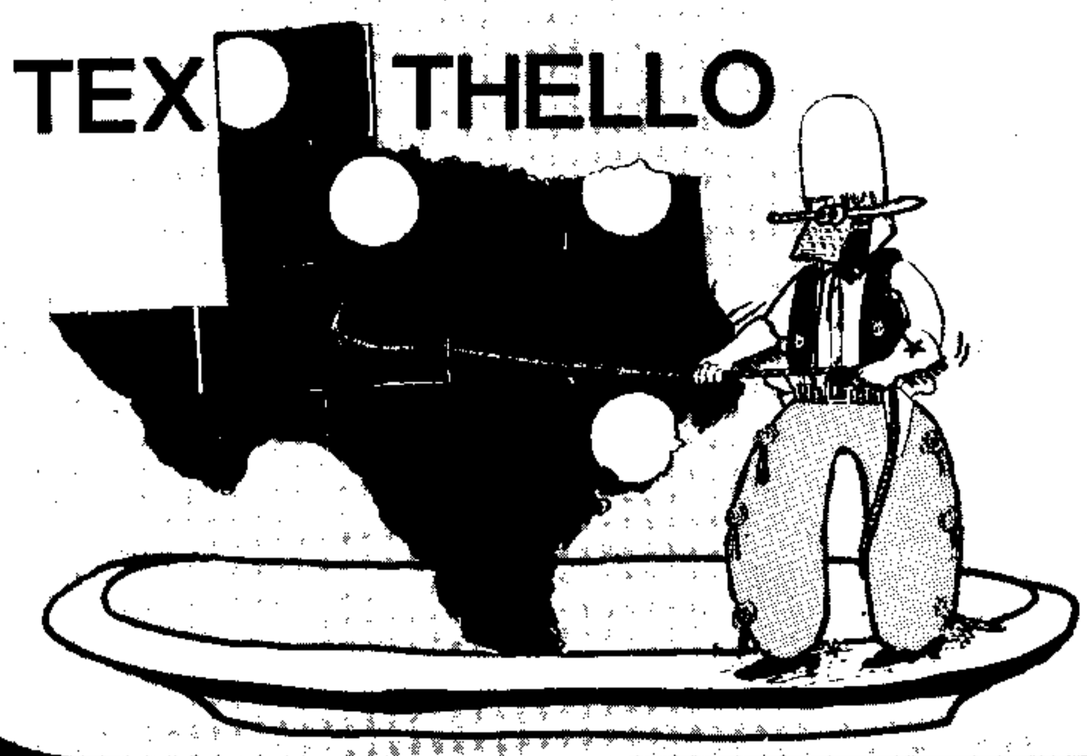
By J. Crawford Cook II
41727 Greenwood
Canton, MI 48187

Tex-Thello is a microcomputer version of the popular *Othello* (a trademark of Gabriel, Industries, Inc.) board game. The program is written in TI BASIC for either the TI-99/4 or TI-99/4A. It pits the human player against the computer for an exciting game on three levels of difficulty: On Level 1, the computer just tries to capture the most markers. On Level 3 (the highest level), the computer takes into account the edge squares and corner squares—thus providing it with more of a theoretical advantage. Level 2 is an intermediate level. The program will check for illegal moves (sounding a warning tone within 30 seconds) and change the color of "captured" markers according to the moves.

Game Rules

1. Since the first four squares in the middle of the board must be occupied (in "checkboard fashion") first, the program automatically provides this initial set-up.
2. The player alternates turns with the computer by entering the grid coordinates for a move. A move consists of placing a color square so that it "captures" (by completing the outflanking of) one or more of the opposite color squares. The computer will then change all the captured squares to the opposite color.
3. A move must always consist of capturing at least one square.
4. If a legal move cannot be made, it then becomes the opponent's turn to move.
5. Capturing may be accomplished horizontally, vertically, or diagonally in one or more rows or directions.
6. The game is over when the board is filled with color squares, or when it is not possible for either opponent to move, or when the board is filled (or partially filled) with all one color. The opponent with the most squares is the winner.

Listing on p. 67



Force 1

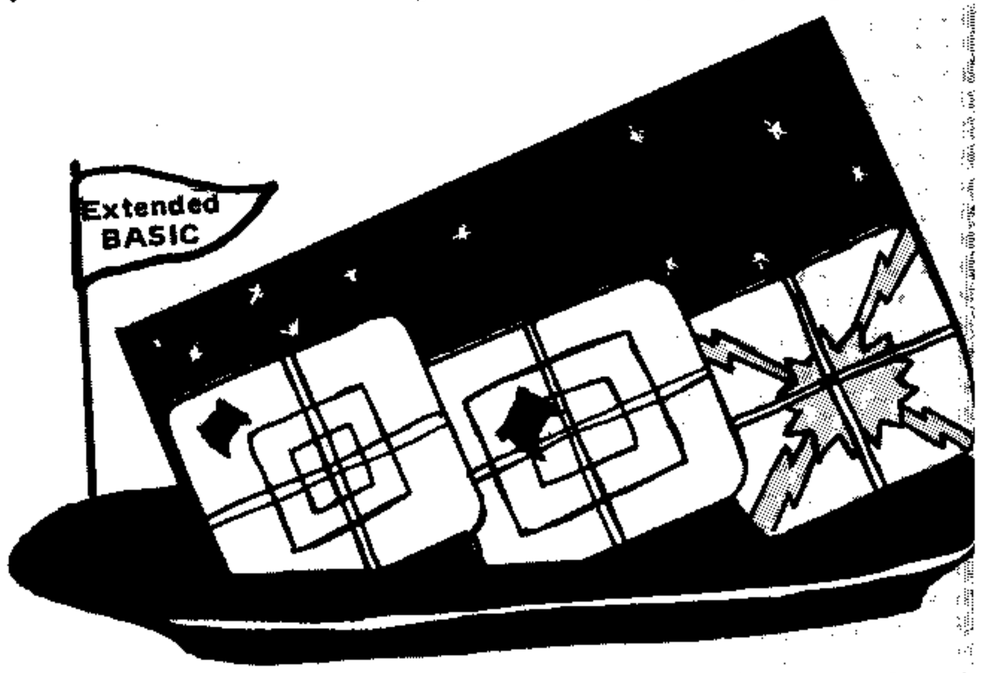
By W. K. Balthrop
Contributing Editor

You are the Captain of the "Force 1," a United Fed ation of Planets police cruiser. A message has just co in that a large number of alien bandits have ente your sector and are planning an attack on your home pla The bandits cannot be taken alive and therefore must be d stroyed. The job won't be easy, so you'd better stay alert.

Since the bandits are armed with short-range laser c nons, they should be encountered when beyond their fir range. As you become a better pilot, you may choose increase your ship's speed with higher levels of difficul This means that the alien craft will be approached mu more rapidly, and more accuracy on your part is needed.

On first sighting, your radar screen will show the alie be no larger than the background stars, and very difficult pick out among them. As you approach the ship, it w become larger and larger, until the alien is either in range fire its laser cannon, or slightly out of range and flies ri past you, instead.

To manuever your ship in order to set your gun sights the alien bandit, you must use the four arrow keys. If y hold a key down continually, your ship will keep acce ating in that direction. This will, of course, cause the s field and the alien ship to move more in the *opposite* di tion. For example, if the alien ship were moving off to right of the screen and you want to bring him back into center, you would hold the "→" key down until the a started moving toward the center. Then to halt all movem by the alien and keep him from going to the left of screen, press the "←" key until the alien either stops slows way down to a minimum speed. The idea is to slow horizontal and vertical speed to a minimum and position h in the center of your gun sight. To fire your laser blas press "ENTER." Getting the alien in your gun sight may



be as easy as it sounds, for the alien is intelligent and per cally shifts course like all skillful space bandits. So just w you think you have him, he's off in another direction . . .

You have 1000 units of time to complete your mis before the strike on your planet. If 25 or more bandit s are destroyed, you will gain an extra 1000 units of tim attack the second wave of aliens.

The Program

The program is written in Extended BASIC. I dec here to make use of the magnify commands to create a s of space ships that start off very small and gradually bec larger. This gives a more realistic view of an object co closer. I gave the ship a random speed—slow at first when at a great distance, and accelerating as it gets closer. I gave the ship the ability to change directions randomly of the time. The ability to use sprites for both the ship the star field made it possible to create the illusion of a motion—not just changing the alien's direction in refer to yours, but also with respect to every star in the star f

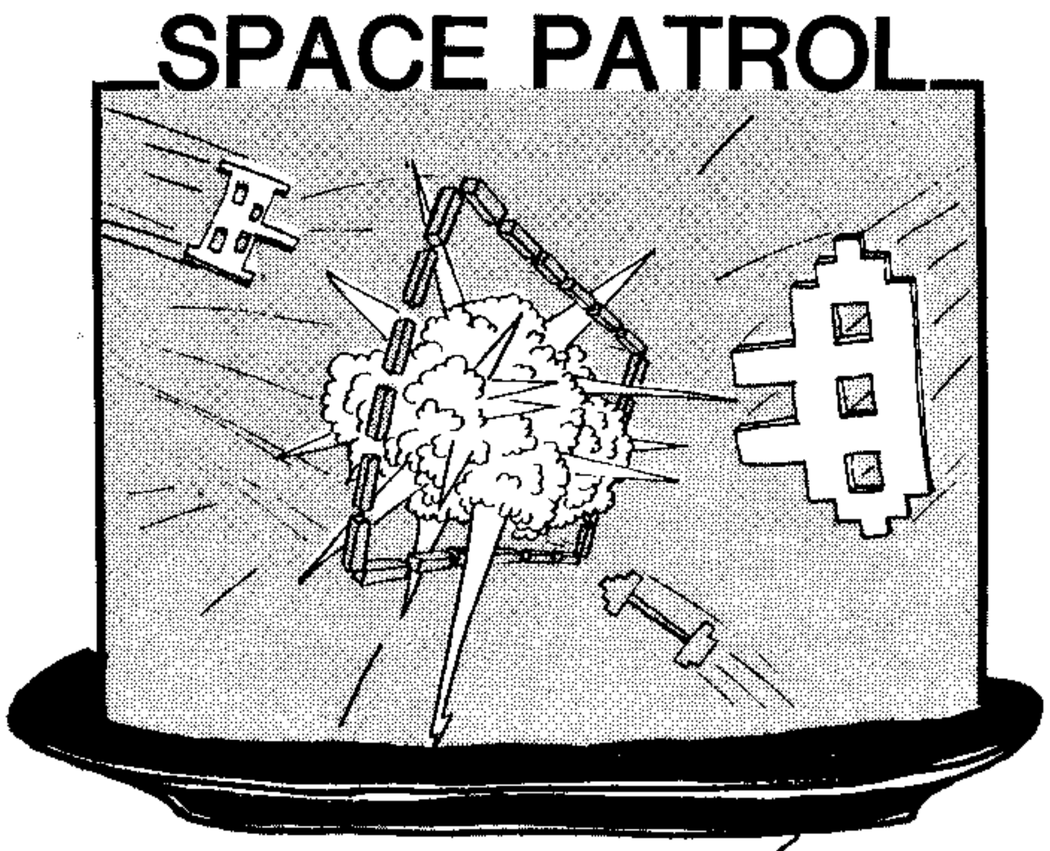
For example, take the case of the alien ship traveling to the right of the screen and all of the stars not moving. If you press the "→" key until the alien stops moving, all of the stars will now be moving to the left, and the alien will be still. This works the same way vertically.

By using the coincidence statement and the tolerance option, I was able to make it more difficult to hit a ship at a greater distance (where it needs to be a direct hit) than to hit one that is nearby. There is however a slight time delay from the time you press the ENTER key until the laser fires. This makes it almost impossible to hit a moving target. So the challenge will be to get the alien in your gun sights and hold him there long enough to make a successful strike.

The laser bolts that you fire at the alien are there all of the time, but kept invisible. I then use the CALL COLOR statement *twice*—once to turn on the bolts, and once to turn them back off.

If the alien ship is still in your gun sight when it reaches maximum size, you will be within range of his laser cannon and be fired upon. WARNING: Laser cannons never miss at short range!

Listing on p. 90



Space Patrol

By Dean Cleveland
2908 South 11th St.
Milwaukee, WI 53215

Extended BASIC

The Earth is at war! Another planet is trying to gain control of our solar system. You are the Captain of a patrol ship armed with high-powered lasers. Your mission—destroy a fleet of 15 enemy supply ships en route to their Battle Star. But be careful, because the supply ships are armed with "killer satellites." When launched, the satellites will move in on your ship and self-destruct unless you destroy them quickly.

Your ship has a supply of 400 energy units, and energy is depleted by 10 units each time you fire your lasers. You also have a deflecter shield that is automatically activated when a "killer" gets past your lasers and explodes near you. This will deplete your energy by 50 units. Your on-board computer will warn you if a "killer" has been launched.

At the start of the game, your gun sight will appear in the center of the screen. You may use a joystick or the arrow keys to position this on your target (depending on the option chosen at the start of the game). Then press either the "fire button" or the "Y" key to fire your lasers.

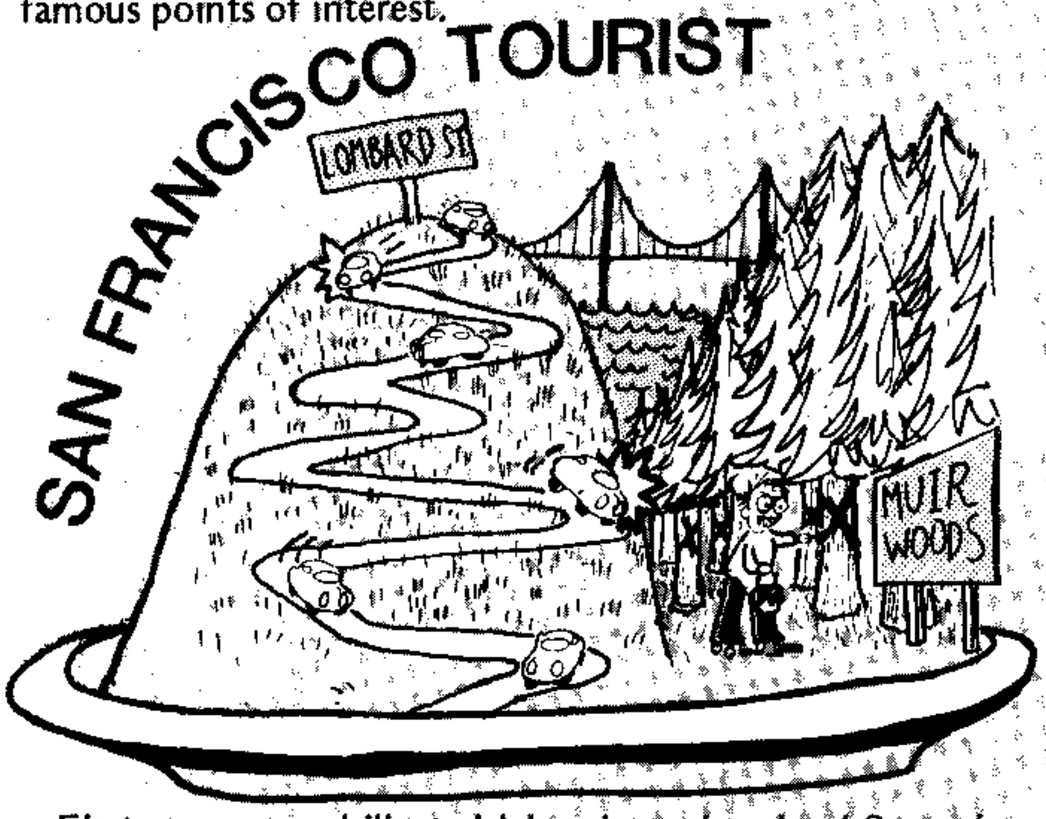
GOOD LUCK AND GOOD SHOOTING, CAPTAIN!!

Listing on p. 91

San Francisco Tourist

By Regena
Contributing Editor

"I left my heart in San Francisco . . ." But I'll be going back to San Francisco in October for the 99'er TI-Fest. "San Francisco Tourist" is a game written in TI BASIC to whet your appetite for all the sights that abound in and around this beautiful city. Actually, this program is two games in one, and is built around a couple of San Francisco's famous points of interest.



First, try your skill at driving down *Lombard Street* between Hyde and Leavenworth. It's on a steep hill and is known as the world's crookedest street. Use the left and right arrow keys (S and D) to steer down the red brick road without bumping into the white concrete sides—or onto someone's green lawn.

Now drive north across the Golden Gate Bridge to *Muir Woods*, a beautiful, peaceful forest of some of the tallest living trees. Start at the upper left cornet of the screen and take a quiet walking tour through the trees. Use the arrow keys to go the right direction, then press ENTER to mark the trees you've seen on your map. You'll have a limited time because you'll have to get back to the city for TI-Fest.

Programming Techniques

This game program implements many of the features I discussed in my article, "Fun and Games", printed in Issue 4. The title screen includes the choice of games, and the player needs to just press the key of his choice (wrong keys are ignored). The program will then branch to the appropriate game and a screen of instructions is printed. The player may just press a key when he is ready to start the game—the screen stays on the instructions only as long as the player wishes.

"Crookedest Street" uses the scrolling during printing to simulate the road going past. A DEFinition statement near the beginning of the program on Line 170 defines a random coordinate R between -3 and +3. A line of road is printed offset R from the previous line. Lines 820 to 850 make sure the line stays on the screen.

Both games move an object (one graphics character for simplicity and speed) by using the arrow keys. In "Crookedest Street" only the left and right arrow keys are used. The car always is drawn on Row 7, and the arrow keys determine whether the car is drawn in the same column, two columns to the left, or two columns to the right. Lines 930-980 keep the car on the screen. In "Muir Woods" the person may move up, down, left, or right, but will not wrap—staying at the edge, instead. The person will also continue to move in one direction until another arrow key is pressed; the character is moved in each CALL KEY loop.

CALL GCHAR(x,y,G) is used in both games. In "Crookedest Street" you need to know if the new position of the

Continued on p. 49

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

TI-994A Keyboard $300

TEXAS INSTRUMENTS
INCORPORATED

Model		Name	Mfr. Sugg. Ret.	Elek-Tek Price
CONSOLE				
PHC	004A	TI-99 4A Home Computer	525.00	**300.00**
PERIPHERALS				
PHP	1500	Solid State Speech™ Synthesizer	149.95	**120.00**
PHP	1600	Telephone Coupler (Modem)	224.95	**180.00**
PHP	1700	RS-232 Accessories Interface	224.95	**180.00**
PHP	1800	Disk Drive Controller (One Disk Manager module packed with each Disk Controller)	299.95	**225.00**
PHP	1850	Disk Memory Drive	499.95	**375.00**
PHP	1900	Solid State Printer	399.95	**300.00**
PHP	2200	Memory Expansion (32K RAM)	399.95	**300.00**
PHP	2400	P-Code Peripheral	399.95	**300.00**
PHA	2100	R F Modulator (TV Adapter)	49.95	**40.00**
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	**9.00**
PHA	2000	Dual Cassette Cable	14.95	**12.00**
PHA	2010	Monitor Cable	19.95	**16.00**
PHA	2020	Audio Adapter (Headphone Jack)	19.95	**16.00**
PHA	2605	Blank Overlays (4 Pack)	7.95	**7.00**
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 required)	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**
		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 (Personal 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	39.95	**32.00**
		Cassette		
PHT	6003	Personal Financial Aids	14.95	**12.00**
PHT	6038	Business Aids Library — Lease/Purchase Decisions	34.95	**28.00**
Education/Personal Enrichment				
		Command Modules		
PHM	3002	Early Learning Fun	29.95	**24.00**
PHM	3003	Beginning Grammar	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	3015	Early Reading† (Solid State Speech™ Synthesizer is required)	54.95	**44.00**
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	3027	Addition and Subtraction II† (Solid State Speech™ Synthesizer is recommended)	39.95	**32.00**
PHM	3028	Addition and Subtraction III† (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	3040	TI LOGO (Memory Expansion is required)	149.95	**75.00**
PHM	3043	Reading Fun†	54.95	**44.00**
		Diskette		
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	Spell Writer (Terminal Emulator II Command Module and Solid State Speech™ Synthesizer 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**

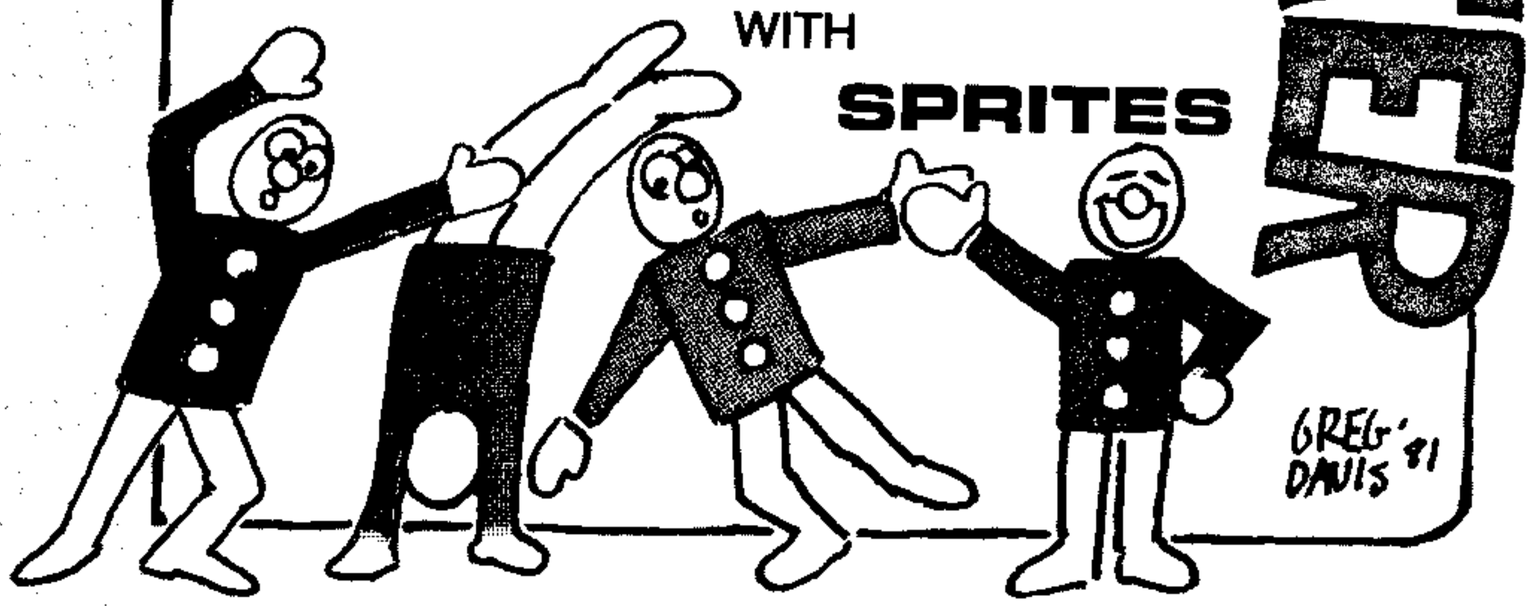
Model		Name	Mfr. Sugg. Ret.	Elek-Tek Price
APPLICATION PROGRAMS CONTINUED				
Education/Personal Enrichment Continued				
PHT	6039	Bridge Bidding II	24.95	**20.00**
PHT	6041	Bridge Bidding III	24.95	**20.00**
PHT	6042	Spell Writer (Terminal Emulator II Command Module and Solid State Speech™ Synthesizer are required)	24.95	**20.00**
Entertainment				
		Command Modules		
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	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**
PHM	3041D	Adventure (Pirate Adventure Diskette Game Included)	49.95	**40.00**
PHM	3041T	Adventure (Pirate Adventure Cassette Game Included)	49.95	**40.00**
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 (available 2nd quarter)	39.95	**32.00**
		Diskette		
PHD	5002	TI-Trek (with 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	24.95	**20.00**
		Adventure Series		
		The following adventure diskettes require the PHM 3041 D Command Module		
PHD	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		
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 Series		
		The following adventure cassettes require the PHM 3041 T Command Module		
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	3000	Diagnostic	29.95	**24.00**
PHM	3001	Demonstration	69.95	**58.00**
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**
		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**
		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**
SOFTWARE LIBRARIES				
PHL	7001	The Home Financial Manager	139.95	**105.00**
PHL	7002	The Family Entertainer	89.95	**70.00**
PHL	7003	The Elementary Educator	99.95	**75.00**
PHL	7004	The Music Educator	64.90	**52.00**
PHL	7005	The Super Programmer	119.00	**95.00**
PHL	7006	The Speaking Math Teacher	119.85	**95.00**
PHL	7007	The Speaking Reading Teacher	109.90	**85.00**
PHL	7009	The TI Arcade Game Series	119.85	**95.00**
PHL	7010	The Milton Bradley Game Series	114.75	**90.00**
PHL	7011	The Computer Introductory Package	119.85	**95.00**

†Developed by Scott Foresman ††Developed by Milton Bradley The Attack, Blasto, Hustle, ZeroZap, Connect Four and Yahtzee are trademarks of Milton Bradley

ELEK-TEK, inc.
6557 N. Lincoln Ave., Chicago IL 60645
(800) 621-1269 (312) 677-7660

CALL TOLL FREE 800-621-1269 (EXCEPT Illinois, Alaska, Hawaii)
Mastercharge or Visa by mail or phone. Mail Cashier's Check, Money Order, Personal Check (2 wks to clear). Add $4.00 1st item; (**Alaska, Hawaii, Puerto Rico, Canada add $8.00 first 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.

HIGH SPEED SPRITER ANIMATION WITH SPRITES



By Fernando Caracena

TI Extended BASIC lets you fill the screen with rapidly moving sprites of many colors. See for example "Sprite Chase" by Ron Binkowski in the July/August issue of *99'er Magazine*. Although the smooth and rapid motion possible with sprites is indeed quite impressive and arcade-like, think how much more spectacular these screen displays would be if we *animate* the moving sprites: After all, why just move a man-shaped sprite when you can also move his arms and legs? Picture the visual impact of a bird-sprite flying across the screen flapping its wings. How about a circus parade with clowns tumbling, animals walking, and elephants moving their trunks . . . All of this, and more, is possible with sprite animation.

The technique of animation is old and well known. First we draw a series of figures with each figure in a slightly different position and posture. Then, we rapidly flash the figures one after the other on the screen, and "persistance of vision" goes to work—fooling our eyes and causing us to see the figure move as if alive. And now with sprites, we can duplicate this movie animation technique on the TI-99/4 and 99/4A through simple commands in Extended BASIC. [This is all made possible by the TMS 9918 and 9918A Video Display Processor chip. See related article in a forthcoming issue—Ed.]

The usual trouble with computer animation is the tedious series of tasks that must be performed after drawing the figures—you have to determine, keep track of, and key in those long pattern identifiers. If you have chosen to work with sprites that are four characters large, these codes then become 64 hexadecimal characters long! This situation prompted me to write *SPRITER* (Listing 1), a program that does much of the work for you, and leaves you free to concentrate on the fun—drawing the figures for the animation sequence. Spriter automatically computes, files, and saves an array of four-character pattern identifiers that define sprites of magnification 3 or 4 (figurines). After you draw each figurine you can output a model of it to the thermal printer (optional) and when you are finished, you can save the whole file on cassette tape or disk.

> **"The usual trouble with computer animation is the tedious series of tasks that must be performed after drawing the figures . . .**
>
> **[BUT]**
>
> **Spriter automatically computes, files, and saves an array . . . that defines sprites . . . and leaves you free to concentrate on the fun . . ."**

When you run SPRITER, it presents you with a 16 x 16 character work area in the screen's character display field. Under your direction, the computer generates an enlarged model of the figurine within the work area. The image is made up of dark and bright character squares, each of which has a counterpart in the figurine. Changes in the display field are automatically converted into changes of the figurine's pattern identifier. The figurines in the computer's memory (RAM) can be stored permanently on tape or disk and later accessed by either SPRITER or any other program with animation recall capability. See for example, the animation demonstration program in Listing 2. SPRITER thus allows you to generate new figurines, transfer any figurine that you have stored on tape into RAM, and rework any figurine that is present in RAM.

How to Run SPRITER

Instructions are almost self contained: A series of prompts guides you through much of the program. First, you are asked, if you have a thermal printer, and if you want to input a file of characters from tape or disk (and the corresponding file name). Afterwards, the work area is framed on the screen. If you have chosen to show an existing figurine by reading in an old file, SPRITER copies that figurine to the work area. When the cursor appears in the upper left-hand corner of the work area, you are ready to draw a new figurine or redraw an existing one. You can move the cursor anywhere within the work area with the motion keys—the arrow keys for horizontal and vertical motion, and with the **W**, **R**, **C**, and **Z** keys for diagonal motions. When the cursor is moved, it automatically leaves a trace as determined by the polarity keys—bright if the **A** key was pressed and dark if the **F** key was previously pressed. When the cursor first appears, the polarity of the trace is dark. Afterwards, by using the motion and polarity keys you can draw and erase portions of the model until you are satisfied with the results. Then press the **Q** key to exit the drawing mode. A new series of prompts will guide you through the rest of the program.

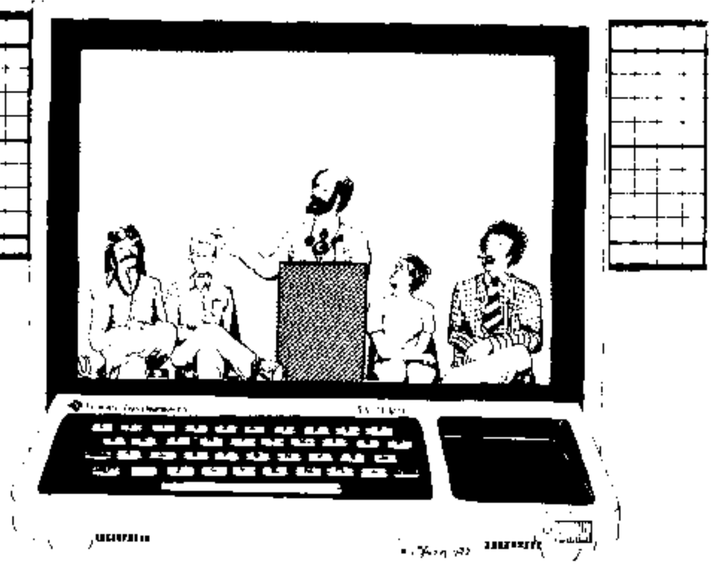
How SPRITER works

Space does not allow a line-by-line description of SPRITER (see Listing 1). But for those interested in exploring the intracacies of the program, I have provided a road map in the form of a structure diagram (Figure 1). Functons identified within the main program are depicted as boxes above the dashed line;

Continued on p. 45

4412 Pali Way
Boulder, CO 80301

PROS ON PROGRAMING



CHUCK-A-LUCK
PART 2
By Samuel D. Pincus

Building a good program is a lot like building a good house. First, you need a good design. Then, you need good tools, good materials, and good work habits to use them all properly. In the last issue, we discussed a way to develop a good design by using a technique commonly called "Structured Design" or "Top-Down Design." Now, we"ll talk a little about how to get the tools, materials and cultivate the habits that we need.

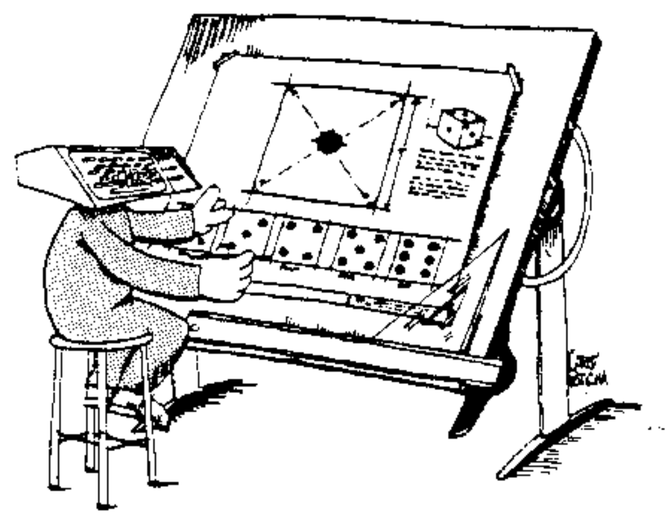
After completing our design effort, you might expect the next step is to begin coding the program. But this, in fact, is not the case. Just having a good design doesn't mean that the code in your program will be correct or that you will write the best code for the job. In every task, there are two things to remember: The first is that you want to do the right job. The second is that you want to do the job right. To do the right job means that your code has to follow the design that you came up with. To do the job right, you have to create the best code for the job. And like anything else, these both require planning. That's right! We still have some planning to do. Only this time, we must plan our code.

The first thing to do is refresh your memory on the design we came up with to play CHUCK-A-LUCK. Take a look at the last issue of *99'er Magazine*. Notice how we developed the modules that tell us *what* we have to do, but not *how* to do it. The purpose of planning our code is to figure out *how* we want to do it in the best possible way. At the same time, we want the *hows* that we develop to be easily coded and debugged, to execute quickly, and to be easily modifiable so that we can make future improvements.

Starting UP with START_UP

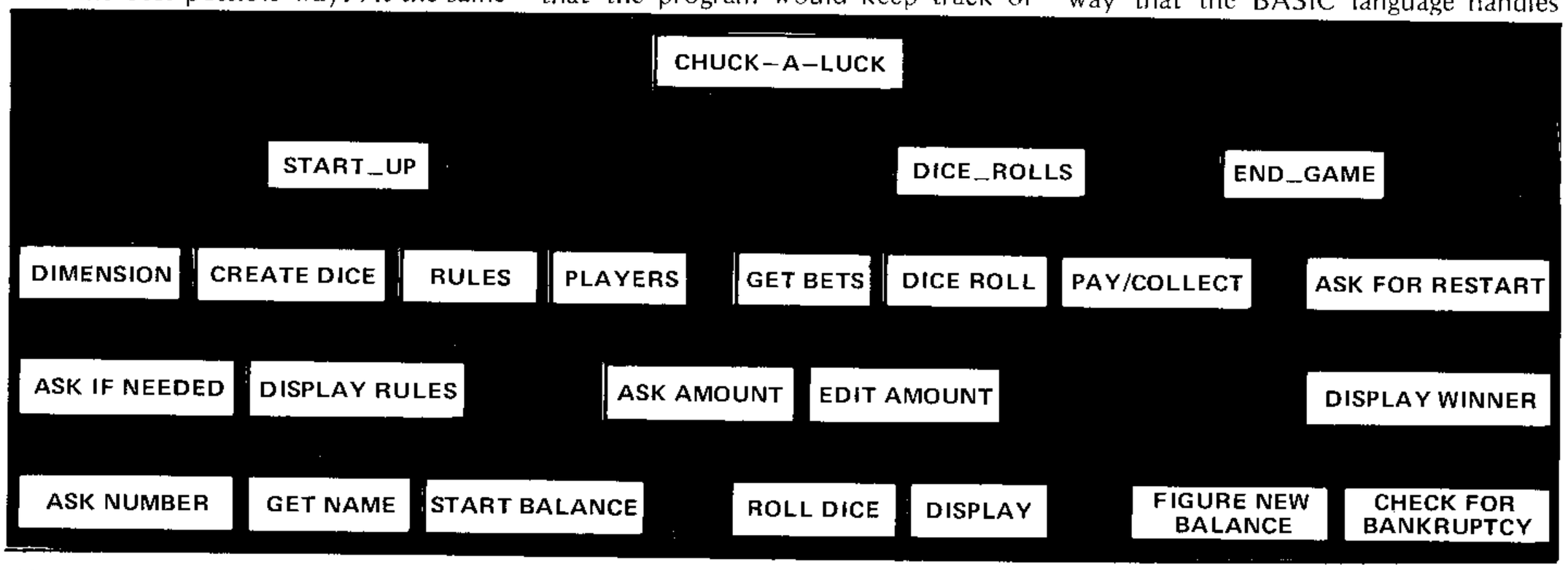
Let's start with the module called START_UP. One of its top-level components was DIMENSION. That module is needed to set up the dimensioning of any arrays needed in the program. Although it is not absolutely necessary to code the DIMension statement at the very beginning of your program, I have found that its always best to put it right up front. So, when I plan my code, the DIMENSION module will be my very first line of code. One good coding habit to get into is to start your programs at line 100, which leaves you room in the front of your program in case you have to add an extra statement to start off your program. I will reserve lines 100-140 for any dimensioning of data that I will need. But before I go on with the remaining design of the code, I think that we had better take time out to talk about the DIM statement and what it is used for.

When I was doing the design, I knew from my original plan that the program was going to have to handle 4 players. That meant that every time I did something concerning a player, I would have to know which player I was dealing with. For example, each player was going to make a bet and win or lose. That means that the program would keep track of these things (called variables) for each player. There are two ways of doing this. The first way is to give a different name to each one of these variables for each player. That is, I could keep track of each player's bet by having one variable called BET-1 and another variable called BET-2, etc. This way, I would know at a glance what was contained inside the variable. The only problem with this way of doing things is that the program needs separate code for each player. This means that you would have to key in more lines of code. It means more chances for data entry errors. It also means the possibility that you could



accidently write the code for each player a little bit differently, which in turn means that you would need to debug your code for each player.

Suppose, however, that you did not need to give each player a different variable name. Suppose that you could just call the variable by the single name of BET. Then the code for each player would be the same. As a matter of fact, you would have to code the logic only once, because it could be re-used for each player. As you can see, this would be a great improvement. There is only one problem with this. You still have more than one player so you would have to be able to say which player's BET you wanted to deal with. Well, the way that the BASIC language handles



this is to allow you to set up an array called BET. This array has only one name but contains multiple slots. Imagine an apartment building called BET containing only one floor with a lot of rooms in it. The room numbers start with 0 and increment by one. Then the computer can put the betting information for player number 1 into room number 1, the information for player number 2 into room number 2, etc. Now, in order to look at the bet of player number 3, all we have to do is to tell the computer to look at room number 3 of BET. We do this by saying BET(3). The value 3 is called a subscript of the array called BET.

This is an improvement over saying BET-3 but not much. But if the computer can be told which subscript (room number) to use via another variable, then you can realize a great improvement. Suppose all you had to do was tell the computer to look at something called X, and that X had the value of the subscript in it. Now you just put the room number inside X and tell the computer to look at BET(X). How do you put the room number into X? The same way you put any number inside any variable. You can say things like X = 3 or X = A + B or set X to a range of values in a FOR-NEXT loop. The important thing is that you do not have to know in advance what is in X before you execute the code. By the way, I used the name X just as an example of my subscript name. We could have called it PLAYER-NUMBER, or I, or any other legal variable name. Also, just because a variable is being used as a subscript in one part of your program, it doesn't mean that the variable can only be used as a subscript. Any variable can be used as a subscript. It is also possible for two (or more) variables to be used as subscripts for the same array depending on what you were trying to do.

"Roomy" Arrays in TI BASIC

Now, two questions should be running through your head. The first question should be, "How many rooms can TI-BASIC build for a given array?" The answer is that it depends on what (if anything) you tell it to do. If you don't tell it anything, it will automatically set aside 8 rooms (slots) for any array it may meet in your code. It will do this the first time it sees the array. If you need more that 8 slots, you have to tell it how many you do need. If you need less than 8, you may not want to waste space on unused slots. In these two cases, you tell it how many slots you need by using the DIM statement.

The second question should be, "What about room 0?" In my game, it is always empty. In some programs, however, room 0 may very well be used. If room 0 is not going to be used at all, you can tell this to TI-BASIC so that it won't waste computer memory with a room 0. This is done by putting a statement with OPTION BASE 1 in front of the DIM statement.

Of course, just as an apartment building can have rooms on more than one floor, an array can have more than one level of slots. But since we don't need multiple levels in our program, we'll leave a discussion of this to a later column. For now, let's look at our variable list and see what variables are going to be arrays so that we will put them in our DIM statement. As stated in my previous article, we will need to keep track of information for 1 to 4 players. In addition, there will be 3 dice and each die will have a value. Look at line 100 to see how I coded the DIM statement for these arrays. Notice that you cannot use a "-" as part of a name in TI-BASIC. The best way to set up a name like DICE-VALUE is to code it using the underline character in place of the dash. That means that I coded the array as DICE_VALUE in the program.

Leaving Out the Difficult is Easier

We are now ready to begin planning the code for the rest of the START-UP module. Because (by definition) this code will only be used once for each game, I like to keep it up far away from the main logic of my program. For this reason, I usually begin coding these one-shot modules at line number 20000. In order to give myself a lot of leeway in case I leave out a line of code or have to add another line during debugging, I always increment my statement numbers by 10 or more. In addition, I also make sure that there is plenty of unused statement numbers between the end of one module and the start of another.

The first module within START-UP to be coded is responsible for creating the graphics for the dice. Naturally, you now expect me to give you the code. But I won't! You see, it's not really important that I do this right away. I can always create the dice later after I am sure that the rest of the program is working correctly. This is one of the important advantages of designing and planning your program in advance.

When you plan your code, don't rush right into figuring out how the code in all your modules will look like. First decide what modules or parts of modules can be left out without affecting the program logic. For example, the code to display instructions can be added as the very last part of your programming effort. A program usually will also contain whole modules requiring complex code that can be replaced by easy code the first time through. After you are sure that the program as a whole is working correctly, you can gradually replace the easy code with the complex code. Why? Because it's easier to find your mistakes in an easy program! So an important "rule of thumb" is to always start out with an easy version of your program.

Then, as you add the difficult pieces, you at least know where to look if you hit a snag in your debugging.

Well, if I leave out all the graphics for now, what can I substitute in its place? I can simply display the number of each die instead of graphically showing the dice themselves. After the program is running, I will go back and add the graphics as well as any sound routines. Look at what I am trying to accomplish this way:

1. By leaving out unnecessary code, I can get the program up and running faster. This means that I can begin debugging my program earlier. This in turn means *easier* debugging because there is less code to go through.

2. By using easy code in place of complex code in some modules, I make it easier to debug the "guts" of my program. After knowing that the program runs correctly, I can begin replacing the easy code with the hard parts a little at a time. Then I will test only one or two new parts at a time. This means easier debugging because any problems will probably be due to the new code.

3. After ensuring that the main portions are running correctly, I can "fool around" with the hard portions without worrying that I will hurt the program's logic. For example, after I know that the program is running correctly by displaying the dice numbers, I can now experiment with how I want the dice themselves to be displayed. I can even come up with two versions—one for TI-BASIC and a different one for Extended BASIC using sprites! I won't have to worry that adding different versions of this code will destroy my program.

4. By getting a version up early, I can see if my program is worth continuing to work on. After seeing it in action, I may decide that it just isn't worth the effort to continue with the coding.

So for right now, I won't code the CREATE DICE routine but I will set aside lines 20000-20500 for the code later. The next module is called RULES and will be responsible for giving the rules when asked. One part asks if the rules are wanted; another displays them. Like the CREATE DICE module, the entire code for this module isn't needed now. But if I do code in the part asking if the rules are wanted, I can test this part of the logic. If the size of the program you are writing is large enough, you may decide to leave both of these parts out on your very first try.

Since I have decided to code part of this module, I will lay it out in lines 21000-22000. The first thing I want to do is clear the screen.This will attract the players' attention and remove any "clutter" that may be on the screen from any previous program. It's always a good idea to start out your program with a CALL CLEAR statement. Notice that in my code in lines 21010-21050, I am asking the players for information *and telling them what form I expect the answer to be in!* Too often, a programmer will code his program so that he is expecting a particular answer, but never tells the person using his program what form the answer should be in. There is nothing more frustrating to a user than trying to figure out what the person who coded the program means when the program displays a message like CODE?, and what the valid values of the input are. You should try to develop the habit of explaining what data you are looking for, and what the legal values the program will allow as part of your code of an INPUT statement.

The next thing the module does is make sure that only the first character is going to be looked at. This is done by using the SEG$ function to strip off the rest of A$. One of my programming "rules of thumb" is to minimize the chance that the user of a program can enter bad data. If I am only expecting a Y or N, I want to look at only the *first* character of the input. If the wrong answer was given, an appropriate message is displayed and the original question is asked again. The code to display the message will be eventually located in lines 21100-21990, but I'll just put in a REM statement to show where the code will be added later.

The next module (called PLAYERS in our design) is very important and easy to code, so I will code it in full the first time out. This is done in lines

22000-22330. Notice that it prompts the player for the required data in each case, and edits the input to insure that only valid data gets in. One of the main differences between a well written program and a poorly written one is the amount of editing done on input. The *hows* of an edit for an alphanumeric field should always include a test for an empty field (called a "null string"). TI-BASIC allows a null string to be entered in response to an INPUT statement. This kind of string data can cause a number of problems in your program, especially if you want to display the data on the screen. I always test for an empty field whenever I INPUT a string variable. That is what I am doing in line 22140.

It may also be necessary to limit the size of an alphanumeric field depending on how you want to display it on the screen. For example, you may want to limit the size of a player's name so that it fits on the same line as his cash balance. The best way to handle this is to check its length (using the LEN function) as part of your edit. If the player enters a name that is too long, you can tell him so, and ask that he enter a shortened version of his name. There is also, however, another way: You may shorten the field yourself by using the SEG$ function—as I do in line 22250.

Whenever numeric fields are entered into your program, there are always four things you must edit for. First, you have to make sure that numeric data was entered. Luckily, TI-BASIC will do this for you automatically so you don't have to write any code to test for this. You should get in the habit of immediately testing your input as follows: (1) check to make sure it isn't too large, (2) check to make sure it isn't too small, and (3) especially check to make sure it is a whole number (if that's what you are expecting). Look at my code in statements 22010-22030 to see what I mean. Also note that if the answer is illegal, I ask for the item to be re-entered. If you don't make it obvious that you want the data entered again, it is possible that the person using your program may not even know that he or she made a mistake and get confused on what to enter next.

The main portion of our design is called DICE_ROLLS; it is responsible for actually playing the game. First, it gets the bets from each player. Then it rolls the dice. Finally, it makes the payments to or collects the losings from each player who is still in the game. Since this code is executed a lot, I will place it in the front of my program. The three main components are called GET BETS, DICE ROLL and PAY/COLLECT. The first two components will be coded as subroutines called from DICE_ROLLS. Line 210 calls GET BETS and line 230 calls DICE ROLL. The third module, PAY/COLLECT, will be coded as part or the DICE_ROLLS module.

Save the Unimportant for Mañana

Why did I set them up this way? The answer to this question requires a little background in the style of coding that I have adopted. As you know by now, I have a number of rule-of-thumb methods that I follow. One of these rules of thumb is that if I get to a module that I will be expanding or replacing later, I set it up as a subroutine to be coded later. I just code in a GOSUB statement and keep going. If it is a module that has to be coded fully the first time around, I usually code it right then—unless it looks like something that is hard to code. In that case, I code in the GOSUB statement and hold off coding it in until I have to. I write my programs this way because I never want to tackle any code that will destroy my trend of thought. After all, one of the reasons we did a design in the first place was to make sure that nothing important will be left out. So if I keep coding, I won't get sidetracked into worrying about the hard parts until I absolutely have to.

Lines 530-560 are used to figure out how many "hits" a player had after the dice were rolled. Notice that this is done using two FOR-NEXT loops, one inside the other. The inside loop in lines 530-560 checks to see if a player bet any of the dice numbers that came up. The outer loop from statements 250-760 controls which player we are looking at. For now, I won't code the full CHECK FOR BANKRUPTCY module. I will instead code a short module (statements 740-750) to check for bankruptcy and STOP the program if there's a loser. Notice how the use of arrays has made this code simple to write. Try to imagine what it would look like if I had to name each variable separately!

The module called END_GAME is also not very important to the main logic of the program, so I'll ignore it for now. This means that the only modules I haven't looked at are GET BETS and DICE ROLL. I coded them in lines 1200-1900 and 2000-2990 respectively. The reason why, I am leaving a lot of room in the DICE ROLL routine is that I still don't know exactly how I am going to do all of it. Oh, I know how to roll the dice, but I haven't gotten around to figuring out what the graphic display of the dice and what the design of the screen will look like . . . and until I do, I can't really figure out all of the *hows* of this module. For now, I will code the DISPLAY routine to just show what the dice were.

In order to simulate rolling the dice, I will have to create three random numbers between 1 and 6. This is done using the RND function in statement 2110. Remember that RND is really random only when you start your program with a RANDOMIZE statement. We will eventually put this in statement 140.

But until I have fully debugged my program, I will leave the RANDOMIZE statement out. Without it, the dice rolls will not be truly random. They will always follow the same pattern from the start of the program. This allows me to replay a game exactly the same way each time, so that if I found a bug and had to correct my code, I can test the corrected code under the same conditions that caused the bug in the first place. With the RANDOMIZE statement in my program, I may never hit the same conditions that caused the bug and won't be sure that I made the right correction.

After coding in these statements, the result is found in Listing 1. Let's briefly review just what this program can and cannot do. First, it *does* play the game according to the rules of CHUCK-A-LUCK. It will handle the bets of up to 4 players. It will keep track of cash held by each player and declare a loser. Once I have this program debugged, I then have to plan what pieces I want to add next. The program is missing 3 major features. First, it stops as soon as one player goes bankrupt, and it cannot be restarted without rerunning the program. Second, it cannot display the rules. Third, it is boring because it doesn't have any of the graphics and sound features that the TI-99/4 can add to a program to make it interesting.

Once I have written enough code to run at least a "stripped-down" version of the program, I would turn my efforts to debugging the code I have written. Only after I was reasonably sure that this version of the program was working properly, would I begin to add more code. I would add one module at a time and retest. But that's the subject for my next article. 99'er

```
90 REM  LIST #1 CHUCK-A-LUCK
100 DIM DICE_VALUE(3),PLAYER_NAME$(4),PLAYER_CASH
    (4),PLAYER_BET(4),PLAYER_DICE(4)
110 REM  PGM START
160 GOSUB 20000
170 REM  BETTING LOOP
200 REM  GET BET
210 GOSUB 1200
220 REM  THROW DICE
230 GOSUB 2000
240 REM  UPDATE CASH BALANCE
250 FOR I=1 TO PLAYERS
260 IF PLAYER_CASH(I)=0 THEN 760
280 PRINT "";PLAYER_NAME$(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 "."
320 WIN=0
330 FOR J=1 TO 3
340 IF PLAYER_DICE(I)<>DICE_VALUE(J)THEN 360
350 WIN=WIN+1
360 NEXT J
370 IF WIN=0 THEN 690
380 WIN=WIN*PLAYER_BET(I)
390 PRINT "YOU ";"WIN";WIN;"DOLLAR";
400 IF WIN<2 THEN 420
410 PRINT "S";
420 PRINT "."
430 PLAYER_CASH(I)=PLAYER_CASH(I)+WIN
440 PRINT "YOU NOW HAVE";PLAYER_CASH(I);"DOLLAR";
450 IF PLAYER_CASH(I)<2 THEN 470
460 PRINT "S";
470 PRINT "."
480 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 STOP
760 NEXT I
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$(I);","
1360 PRINT "YOU HAVE";PLAYER_CASH(I);"DOLLAR";
1370 IF PLAYER_CASH(I)<2 THEN 1390
1380 PRINT "S";
1390 PRINT ".":"WHAT'S YOUR BET? "
1400 INPUT PLAYER_BET(I)
1410 IF PLAYER_BET(I)<1 THEN 1450
1420 IF PLAYER_BET(I)>PLAYER_CASH(I)THEN 1450
1430 IF PLAYER_BET(I)>50 THEN 1450
1440 IF INT(PLAYER_BET(I))<>PLAYER_BET(I)THEN 1470
1450 PRINT "THAT'S NOT POSSIBLE."
1460 GOTO 1230
1470 PRINT "WHAT NUMBER WILL YOU BET ON?"
1480 INPUT PLAYER_DICE(I)
1490 IF INT(PLAYER_DICE(I))<>PLAYER_DICE(I)
     THEN 1520
1500 IF PLAYER_DICE(I)<1 THEN 1520
1510 IF PLAYER_DICE(I)<7 THEN 1540
1520 PRINT "TRY AGAIN."
1530 GOTO 1470
1540 NEXT I
1550 RETURN
2000 REM
2100 FOR I=1 TO 3
2110 DICE_VALUE(I)=INT(RND*6)+1
2112 NEXT I
2120 PRINT "THE DICE ARE:"
2130 PRINT DICE_VALUE(1);"   ";DICE_VALUE(2);
     "   ";DICE_VALUE(3)
2140 RETURN
20000 PL$="PLEASE ANSWER THE QUESTION"
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 REM  INSTR
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
```

Spriter . . . from p. 39

those identified with subprograms are below the dashed line. The order of program execution in this figure is from left to right, and the order of subprogram calls is from top to bottom. The line numbers to which these various functions refer to in the program are listed at the bottom of each box.

The main task of drawing a series of sprite figurines is under the direction of Function 3 ("Draw & Save Sprite Codes"). The task of initializing the work area and handling individual figurines and their models is directed by subprogram EXPANDER which constructs this model when given the pattern identifier for the figurine. After this, DRAWER directs changes in the model display according to the user's keyboard inputs. Then it calls upon subprogram ADDPIX to make the corresponding changes in the pattern identifier for the figurine that is being drawn. When the figurine is complete, DRAWER will call subprogram SCREEPT to output the model on the thermal printer (if this option is chosen).

A Demonstration Program

After you generate cassette or disk data files of figurine pattern-identifiers with SPRITER, you are ready to incorporate these into an animation sequence within a program. The short demonstration program (Listing 2) is an example of a very simple way to do this. This program is in effect a continuous loop projector that sequences through a series of sprite figurines to produce animation of the sprite that is moved across the screen. The program reads the pattern identifiers from cassette tape or disk, then goes into the animation loop. You can stop the looping by pressing SHIFT C (on the 99/4) or FCTN 4 (on the 99/4A).

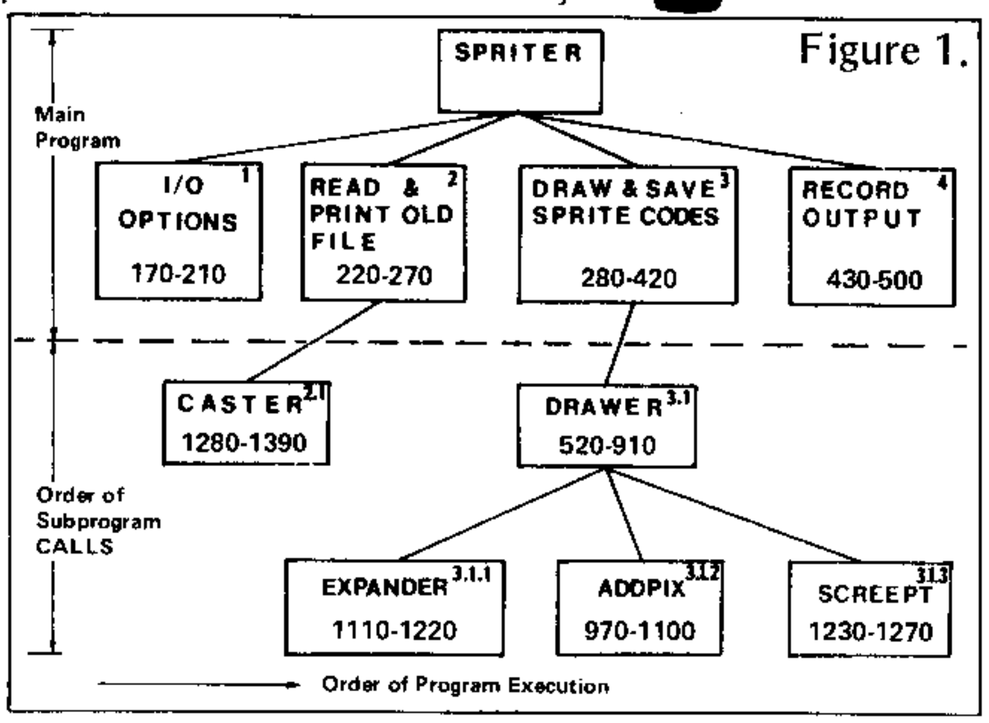
Keep in mind that this program is just a very simple demonstration of the sprite animation technique. You can use it to study the figurine files created by SPRITER, and perhaps as a starting point for writing more elaborate sprite animation programs that are more apt to your specific applications [We've also included an additional program (Listing 3) that incorporates Data Statements for those wishing to get a feel for the animation process *before* working with SPRITER—Ed.] 99'er

Listing 2.

NOTE: ➡ For Longer Loops use a Larger DIM

```
100 REM **********************
110 REM *   SPRITE DEMO      *
120 REM **********************
130 REM
140 REM BY FERNANDO CARACENA
150 REM 99'ER VERSION 1.5.1X8
160 REM
170 REM DEMONSTRATION OF SPRITE
    ANIMATION USING CASSETTE
    OR DISK DATA FILE.
180 CALL CLEAR
190 DIM CHA$(17),ID$(17)
200 CALL CASTER(NS,ID$(),CHA$())
210 FOR I=0 TO NS :: CALL
    CHAR(136-4*I,CHA$(I))
220 NEXT I
230 CALL CLEAR
240 CALL SPRITE(#1,136,2,30,30,0,
    -10):: CALL MAGNIFY(4)
250 FOR I=0 TO NS :: CALL PATTERN
    (#1,136-4*I):: CALL DELAY :
    : NEXT I :: GOTO 250
260 END
270 SUB CASTER(N,I$(),C$())
280 REM
290 REM **********************
300 REM SET FILE$="CS1" FOR
310 REM TAPE FILES; USE YOUR
320 REM FILE NAME:
330 REM      "DSK1.FILENAME"
340 REM FOR DISK FILES.
350 REM **********************
360 REM
370 OPEN #2:FILE$,INTERNAL,
    INPUT ,FIXED 128
380 INPUT #2:NAM$,N
390 FOR I=0 TO N
400 INPUT #2:I$(I),C$(I):
    : NEXT I :: CLOSE #2
410 SUBEND
420 SUB DELAY
430 FOR I=0 TO 15 :: NEXT I
440 SUBEND
```

Figure 1.

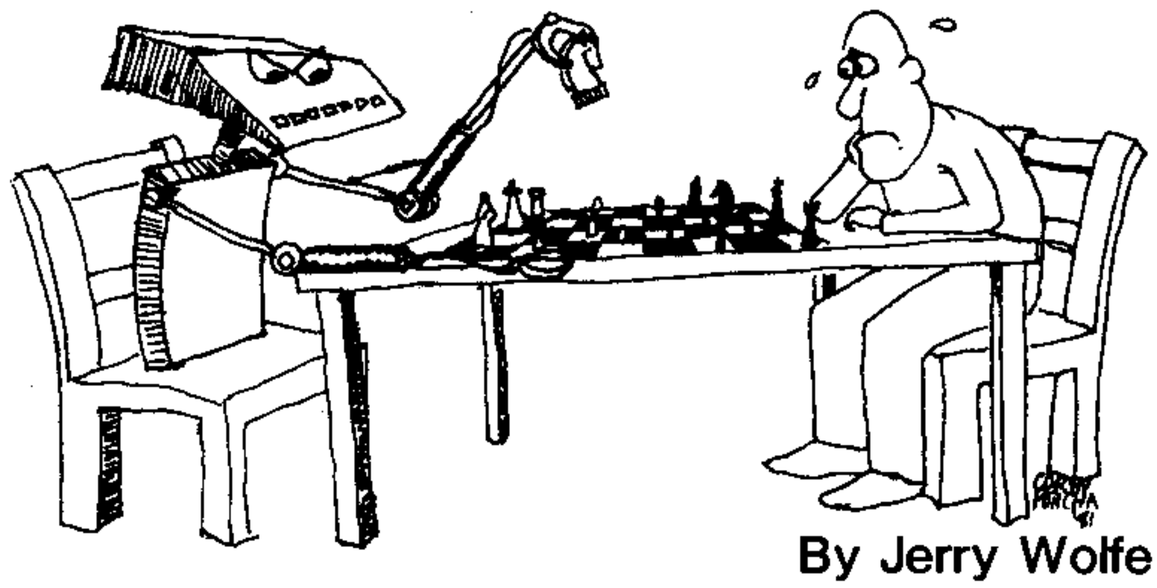


Listing 1.

```
100 REM **********************
110 REM *      SPRITER         *
120 REM **********************
130 REM
140 REM BY FERNANDO CARACENA
150 REM 99'ER VERSION 1.5.1X8
160 CALL CHARSET :: FOR I=96 TO 143 :: CALL CHAR(I,""):: NEXT I
170 INPUT " DO YOU HAVE A THERMAL PRINTER(Y/N)?":TP$
180 DIM CHA$(50),ID$(50)
190 INPUT "DO YOU WANT TO INPUT A FILE OF CHARACTERS FROM TAPE
    OR DISK (Y/N)?":AN$ :: IF AN$<>"Y" THEN 240
200 DISPLAY AT(24,1):"FILE NAME" :: ACCEPT AT(24,11)SIZE(10)
    VALIDATE(UALPHA,DIGIT):NAM$ :: IF POS(NAM$,"",1)<>0 THEN 200
210 PRINT "ENTER '1' FOR TAPE                    '2' FOR
    DISK" :: INPUT "(1/2)?":AN$
220 IF AN$="1" THEN @FILE$="CS1" ELSE IF AN$="2" THEN
    @FILE$="DSK1."&NAM$ ELSE GOTO 210
230 CALL CASTER(@FILE$,NS,ID$(),CHA$()):: GOTO 250
240 IF AN$<>"N" THEN 190 ELSE NS=0 :: GOTO 290
250 IF TP$="N" THEN 280
260 OPEN #1:"TP.U.E.S",OUTPUT :: FOR J=0 TO NS
270 PRINT #1:J,ID$(J):: NEXT J :: CLOSE #1
280 NS=NS+1 :: C$=CHA$(0)
290 FOR I=NS TO 1000
300 CALL DRAWER(TP$,C$,NS,AN$,CHA$(),ID$())
310 DISPLAY AT(2,1):ID$(NS):: DISPLAY AT(22,1):C$
320 DISPLAY AT(3,1):"PRESS ANY KEY TO CONTINUE."
330 CALL KEY(0,K,S):: IF S=0 THEN 330
340 CALL CLEAR :: INPUT "ENTER COLOR CODE FOR SPRITE. ":COL
350 CALL CHAR(96,C$):: CALL SPRITE(#1,96,COL,30,30,0,-30):: CALL MAGNIFY(4)
360 DISPLAY AT(10,3):"PRESS ANY KEY TO CONTINUE."
370 CALL KEY(0,K,S):: IF S=0 THEN 370 ELSE CALL DELSPRITE(ALL)
380 INPUT "DO YOU WANT TO SAVE THE CHARACTER CODE OF THIS SPRITE(Y/N)?":AN$
390 IF AN$="Y" THEN CHA$(NS)=C$
400 INPUT "DO YOU WANT TO CONTINUE(Y/N)?":AN$ :: IF AN$="N" THEN 430 ELSE
    IF AN$<>"Y" THEN 400
410 NS=NS+1
420 NEXT I :: END
430 INPUT "DO YOU WISH TO SAVE RESULTS ON TAPE OR DISK(Y/N)?":AN$
440 IF AN$="N" THEN 510 ELSE IF AN$<>"Y" THEN 430
450 DISPLAY AT(24,1):"ENTER FILE NAME" :: ACCEPT AT(24,11)SIZE(10)
    VALIDATE(UALPHA,DIGIT):NAM$ :: IF POS(NAM$," ",1)<>0 THEN 450
460 PRINT "ENTER '1' FOR TAPE                    '2' FOR DISK" :: INPUT "(1/2)?":AN$
470 IF AN$="1" THEN @FILE$="CS1" ELSE IF AN$="2" THEN @FILE$="DSK1."&NAM$
    ELSE GOTO 460
480 OPEN #1:@FILE$,INTERNAL,OUTPUT,FIXED 128
490 PRINT #1:NAM$,NS
```

Continued on p. 94

COMPUTER CHESS CORNER



By Jerry Wolfe

My column this time is concerned with "versions and diversions" since we are going to look at some variations of standard chess (versions), as well as a few interesting problems associated with chess but not directly related to playing the game (diversions). You'll be able to try all of this on your TI-99/4A computer with the *Video Chess* Command Module.

Diversions

By now, of course, you are already well acquainted with chess problems—having had two thrust upon you with each issue of *99'er Magazine*. Those problems have been taken from positions in actual games. But chess literature also abounds in problems that have little or no relevance to practical play, but are nevertheless extremely intriguing. Here are a few:

Problem 1: This is called the "Knight's Tour." Place a knight on an empty board (on A1 for example) and move the knight 63 consecutive times in such a way as to land on each square exactly once, and returning to the beginning square on the 64th move.

Problem 2: Remove the squares H1 and A8 from the chessboard. Is the "Knight's Tour" still possible now? You are required to prove that your answer is correct!

Problem 3: This problem involves a knowledge of chess plus the ability to deduce logically. While playing a game of chess, black became irked at his losing position and petulantly removed his king from the board. At that moment, white was in the middle of making his move; and for an instant after removal of the black king, the board was completely empty. White completed his move and black cooled down and replaced his king. But then he made the worst possible move on the board and white announced mate in two moves. Your task is to reconstruct the position just before white moved and give the exact sequence of moves leading to the checkmate of the black king. (Yes, the problem has a solution!)

Problem 4: Place eight queens on an otherwise empty board in such a way that no two queens are attacking each other.

Problem 5: Find the shortest number of moves necessary to produce a *stalemate* starting from the standard beginning position.

The above problems represent only a small sample, but perhaps give some idea of the variety of possibilities. Oh yes, I will provide solutions next time except for Problem 5, for which the minimal number is not known. It is a much smaller number than one would think on first seeing the problem. Try it and see what you can come up with . . .

About the Author

Jerry Wolfe is a professor of mathematics at the University of Oregon in Eugene, Oregon. He has been playing chess since the age of eleven and began playing in chess tournaments at the age of fourteen. He is the 1979 Oregon Open champion and has won numerous other local tournaments in the Pacific Northwest during his chess "career." Currently he holds the official rating of candidate master.

Versions

Besides the diversions provided by such puzzles, chessplayers have also been attracted by variations on the basic game of chess. "Speed Chess" or five-minute chess is a version requiring a chess clock. Initially each player is given five minutes of time. Play then proceeds until one side is checkmated, a draw is declared, or until one side runs out of time. (For those of you who are not acquainted with the use of a chess clock, I should explain that the player whose turn it is to move has his clock running until he makes his move. He then pushes a button stopping his own clock and starting that of his opponent.) Thus each game lasts no more than ten minutes. This version is widely popular at chess clubs and among tournament players.

Another currently popular version, especially with younger players is called "Siamese Chess." This involves four players divided into teams of two players each and requires two chess sets and (usually) two chess clocks. The partners sit on the same side of the table and play opposite colors. Thus the pieces that one partner captures will be the same color as those his partner will be playing on the adjacent board. As one partner captures a piece from his opponent, he passes it to the other partner. The reason for this is that in addition to the usual moves of chess, one is allowed to place new pieces on the board anywhere that is not occupied—with the one exception that pawns may not be placed on the back ranks (squares A1 - H1 or A8 - H8) where they could normally be promoted instantly to a more powerful piece.

The placement of new pieces on the board causes the chess battle to take place at an accelerated pace, and causes unusual and often hilarious positions to occur. To make matters worse, the clocks on each board are set for five minutes as in speed chess! The game ends when either a checkmate occurs in one of the games, both games are drawn, or one side runs out of time.

Although chess is far from being "played out," so much study has been devoted to the opening portion of the game that it is possible to go through the first twenty moves in some openings simply repeating moves that are already known to be good. These are called "book moves" because they can be found in chessbooks dealing with openings. This means that a player may obtain a substantial advantage in the opening stages of a game simply by memorizing several sequences of moves found in openings books. At the grandmaster level, this tendency is so refined that victory often hinges on knowing the latest wrinkle in the theory of some particular opening variation, and springing it on a less prepared opponent—one who must then expend extra time on his clock searching for the best reply to this surprise. To combat this over-refinement of opening theory, a simple variation of chess has been proposed. It is called "Prechess" and is played exactly like ordinary chess except for the first

"The placement of new pieces on the board causes the chess battle to take place at an accelerated pace, and causes unusual and often hilarious positions to occur."

eight moves of the game. These proceed as follows: Both sides line up their pawns in the usual way, but leave the row behind the pawns empty. Then the first eight moves consist of each side alternately (beginning with white as usual) placing down one piece at a time on the back row anywhere that is unoccupied. This is done until all eight pieces on each side are placed. Then the games continues in the usual way. Since all opening theory is based on the *standard* starting position (which this is usually not), the printed variations found in the opening books are useless.

This version appeals to many serious players and has the advantage that it can be played with a standard set, involves no bizarre rule changes, and basic chess principles apply as strongly as ever. By using the problem mode of the *Video Chess* program to set up the initial position, you can play "Prechess" on your computers. Try it sometime. Very unusual and interesting games often result.

Since I have already given five problems in this column, I will not add any more. We will return to our regular problem format in the next issue.

Spotlight . . . from p. 35

curing that causes the monsters to be generated . . . Somewhere along the line, management got upset by the idea of anything nuclear happening—especially a nuclear accident—so that scenario was completely thrown out, and eventually settled on the present *Tombstone City: 21st Century* [in which players find themselves in the 21st Century in an Old West ghost town threatened by an invading hoard of green alien monsters—villainous creatures called morgs who live on tumbleweeds and people].

GMK: *Did you get help from others during the development process, or did you work entirely on your own?*

JCP: Before I could start doing the game, I needed some help from the systems programmers to show me how to access the sound and get things up on the screen. As for the programming, that was all done by myself. There were several people who offered criticsm.

GMK: *Was this criticism given during the actual "playtesting" period?*

JCP: Yes. One of the main criticisms was that the game was hard to play. At first there wasn't a white background for a cactus indicating where the monster was coming out. In my mind, you really didn't need that extra help because the monster still comes out in a totally predictable place. It is really very simple—the generation is done from the top of the screen to the bottom, and from left to right. But maybe having the white square helps a little bit so you really don't have to actually think about where the monster is coming from.

GMK: *Were there any other critisms during the period of play-testing?*

JCP: I guess that was the main one. Later, toward the end of the process when more people had played it, people discovered that they could sit on the first screen and just shoot tumbleweeds all the time to run up a high score. This was a criticism that was not resolved in the released version, although that [technique of scoring] was not the way the game was intended to be played.

GMK: *When someone criticized part of the game how did you handle it?*

JCP: There were really different levels of criticism. For example, there was a suggestion to maybe change the spaceship to a six-shooter. That type of criticism I threw out because I could clearly picture a six-shooter, and I knew I didn't want it. Other levels of criticism, like the white background indicating where the monster comes out, I had my doubts about; but I went ahead and implemented it, and felt that it was probably a good feature after I played it awhile.

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

JCP: When the game is played as it was intended to be played, I think it can be very much a game of strategy. It does take some dexterity to manipulate the keyboard, however. Some people really have a problem using the keyboard, so from that sense, it is a game of skill. Due to the speed of reaction of the keyboard, I prefer to play the game this way rather than with joysticks. In my opinion, once you get used to the keyboard, it is very fast—and to me, it's very easy to play.

GMK: *I found the joystick reaction quite fast when compared with a game like The Attack. How do you account for such a difference in speed?*

JCP: The big difference between the two is because *The Attack* is written in GPL, a pseudo-assembler language. *Tombstone* was written in 9900 code, and 9900 code is, of course, quite a bit faster than GPL. In the GPL version of *The Attack*, I'm pretty sure they were utilizing the full speed of GPL, whereas in *Tombstone* we actually had to put in delay loops to keep the ship from moving too fast.

GMK: *What features or elements of strategy did you put into the game that a typical player might miss seeing?*

JCP: After you've played the game a while, I don't think there are really any

Continued on p. 48

Spotlight . . . from p. 47

hidden features, but there are some basic strategies that a first-time user might not realize. One of those features is what I call "dragging." That is where you have a lot of generating pairs of cacti all lined up in a row, and when the monster has generated from the first pair, you shoot it—destroying the first generating pair, and then you drag the monster over to the next generating pair and kill it next to those; it kills that generating pair, and then you drag the monster on and on until maybe you've got six generating pairs wiped out from a single monster. You do all this by positioning your spaceship so that the monster has to go by a generating pair of cactus in order to come get you. It's a chain reaction type set-up.

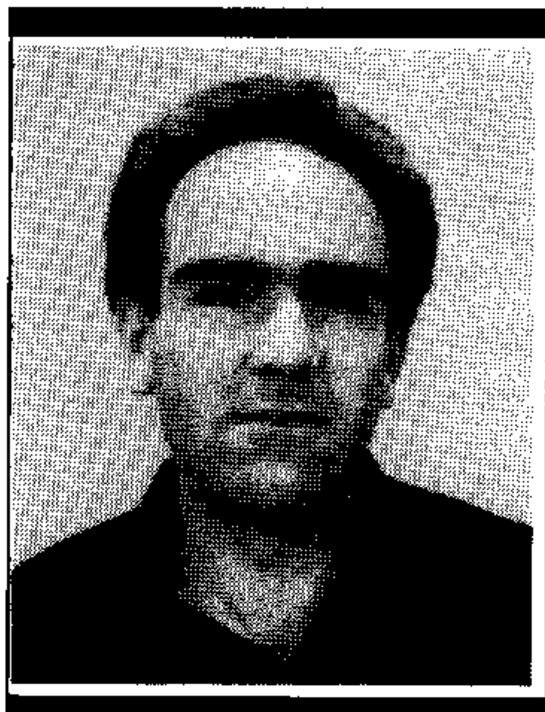
GMK: *What do you feel is the most challenging aspect of the game?*

JCP: Certainly, the hardest level to play on is Level 3, so that would present the most challenge. It is also very challenging to get yourself trapped in the safe area and allow nine monsters to be on the outside. From this position it is very hard to recover.

GMK: *Is it really possible to play well on Level 3?*

JCP: Yes. I quit playing on Levels 1 and 2, since I found that they could be mastered fairly easily. Level 3 I still haven't been able to master, so it is still very challenging.

GMK: *When one becomes a better player of Tombstone and is able to score significantly higher and advance farther, it appears that the game takes on a different character and becomes more addictive. Is this a fair assessment, and if so, how did you achieve it?*



"I now realize that graphics are probably the most important thing in a game. I think you can probably be all right by throwing graphics on the screen and designing the game around your graphics rather than designing the game first, and then doing the graphics. Sometimes it works better one way, sometimes the other. But a lot of times your graphics influence how the game is going to be played."

JCP: I think that's a fair assessment. I really didn't have in mind what would happen on the higher levels—when you have more cactus out there to start with. That's when the game, to me, becomes more interesting . . . when you've got maybe twenty generating pairs out on the screen to start with. Since I didn't intend it to, or even realize it would be this way, it was actually an accident.

GMK: *You must have become quite happy when you discovered this . . .*

JCP: Yes, happy that it did become more interesting at that level, but maybe a little disappointed that it wasn't that interesting when you *first* started playing it.

GMK: *How did you feel letting the game go after the final polishing was done?*

JCP: Overall, I felt pretty good about the game. I was a little upset about a few things. One thing that I guess didn't make a whole lot of difference was the title of the game. They changed it from its original title to *Tombstone City: 21st Century*—the *21st Century* to sort of justify the Western-based conflict. The other thing that I had to change was the actual tombstones to saguaro cactus. What are saguaro now, were originally old Western-type crosses—with the two sticks forming a cross. That was objected to because of religious reasons. It made a little bit more sense originally when you actually had the tombstones in there. So when the tombstones were taken out and replaced with the saguaro cactus, it didn't make as much sense to me . . . but I guess it's not a big difference.

GMK: *When you finally let the game go, were there any things that you thought about that you might have liked to have added?*

JCP: I thought about several things I could have added, but I was still pleased with the play of the game. You can go on and on with games, but there has to be a cut-off somewhere. Since I was pretty well satisfied with the point I had reached, I was ready to let it go.

GMK: *What did you learn from the experience of designing and programming this game?*

JCP: Well, of course, the main thing was learning the

Continued on p. 52

San Francisco Tourist . . . from p. 37

car is a red square (okay), a white square (crash), or a green square (fatal crash). After the new car position is drawn, the old position must be replaced by the appropriate colored square.

"Muir Woods" uses GCHAR to determine positions of trees for marking. Also, the person leaves a trail. So if the square was a blank, the trail is printed; but if it was a tree or a marked tree, that character stays there.

"Muir Woods" also demonstrates the use of a timer or counter in the CALL KEY loop. You may change the value 100 for SH in line 1910 if you want more or less time.

I wanted to use ENTER as the key to press for "firing," so the split keyboard method of detecting the "fire" key was not possible. If you use the split keyboard you can alternate calling the halves of the keyboard and detect the "fire" key sooner, but since the codes are different for the 99/4 and the 99/4A consoles, the game instructions would have to be different. ENTER is not detected on the 99/4A; the period returns key code 13. In these games the quickest way to detect ENTER is to let go of the arrow keys before pressing ENTER.

99'er

EXPLANATION OF THE PROGRAM
San Francisco Tourist

Line Nos.	
150-170	Defines functions to be used as random coordinates.
180-240	Clears screen; defines graphics characters for bridge.
250-380	Prints bridge and title; if the program is just starting, plays "I Left My Heart in San Francisco."
390	Prints chioces of games.
400-420	Defines graphics characters for games.
430-460	Receives player's input and branches appropriately.
470-500	Subroutine to press any key to start.
510-530	Subroutine to delay.
540-610	Prints instructions for "Crookedest Street"; defines graphics characters; waits for player to press a key.
620-750	Clears screen; defines graphics colors; prints game screen. DATA contains coordinates for printing road.
760-780	Initializes coordinates of road and car.
790-860	Prints 75 lines of crooked street randomly; last 15 lines are straight.
870-980	Makes sound, draws new position of car depending on key pressed; replaces old position with proper graphics character.
990-1070	Tests for crash; makes a sound and increments number of crashes.
1080-1170	Ending remarks; plays victory melody for zero crashes.
1180-1210	Procedure if car goes into green.
1220-1250	Delays then waits for player to press a key.
1260-1290	Clears screen; returns colors to black; branches to menu screen.
1300-1400	Prints instructions for "Muir Woods" and defines graphics characters and colors.
1410-1470	Clears screen, randomly draws 70 trees on screen.
1480-1520	Initializes time, marked trees, coordinates, graphics.
1530-1540	Places person at entrance and sounds initial beep.
1550-1860	Moves person depending on key pressed. Person will not "wrap" but stays at edge.
1870-1920	Increments time and prints time; if time=100, ends game.
1930-2010	Procedure for marking rree.
2020-2100	Ending statements; returns colors to black; branches to menu screen.
2110-2130	Ends program.

```
100 REM  ***************
110 REM  * SF TOURIST *
120 REM  ***************
130 REM  BY REGENA
140 REM  99'ER VERSION 1.5.1
150 DEF R19=INT(19*RND)+1
160 DEF R28=INT(26*RND)+2
170 DEF R=(-1)^(INT(4*RND))*(INT(4*RND))
180 CALL CLEAR
190 CALL CHAR(64,"18183C3C5A5A9999")
200 CALL CHAR(35,"010102040830FFFF")
210 CALL CHAR(47,"1818181818FFFF")
220 CALL CHAR(36,"80601806FFFF")
230 CALL CHAR(37,"01061860FFFF")
240 CALL CHAR(38,"80804020100CFFFF")
250 CALL SCREEN(4)
260 PRINT TAB(12);"@  @";TAB(11);"#/$%/&"
270 PRINT :::"** SAN FRANCISCO TOURIST **":::::
280 IF I<>0 THEN 390
290 P=300
300 CALL SOUND(P,330,0)
310 CALL SOUND(P,349,0)
320 CALL SOUND(P,440,0)
330 CALL SOUND(4*P,392,0,131,10,165,8)
340 CALL SOUND(P,440,0)
350 CALL SOUND(P,494,0,165,10,196,8)
360 CALL SOUND(P,523,0)
370 CALL SOUND(P,440,0,147,10)
380 CALL SOUND(4*P,294,0,220,8,175,10)
390 PRINT "WHICH DO YOU WISH TO VISIT?":
    :"  1  CROOKEDEST STREET"::"  2  MUIR
    WOODS":::"  3  END PROGRAM"::::
400 CALL CHAR(33,"FFFFFFFFFFFFFFFF")
410 CALL CHAR(41,"FFFFFFFFFFFFFFFF")
420 CALL CHAR(40,"0")
430 CALL KEY(0,K,S)
440 IF (K<49)+(K>51)=-1 THEN 430
450 CALL CLEAR
460 ON K-48 GOTO 540,1300,2110
470 PRINT "PRESS ANY KEY TO START.";
480 CALL KEY(0,K,S)
490 IF S<1 THEN 480
500 RETURN
510 FOR I=1 TO 500
520 NEXT I
530 RETURN
540 PRINT " ** CROOKEDEST STREET **"
550 CALL CHAR(96,"387C7C38387C7C38")
560 CALL CHAR(97,"C3633618183663C3")
570 PRINT ::"LOMBARD STREET IS THE":
    :"CROOKEDEST STREET IN THE":
    :"WORLD.  IT IS ONE BLOCK"
580 PRINT :"LONG ON A STEEP HILL.":
    :"YOUR CHALLENGE IS TO DRIVE":
    :"DOWN THE RED BRICK ROAD"
590 PRINT :"WITHOUT BUMPING THE CONCRETE":
    :"SIDES.  USE THE ARROW KEYS"::"TO STEER.":::
600 C=0
610 GOSUB 470
620 DATA 6,6,7,8,9,11,13,15,18,20,20,17,15,13,
    11,9,6,6,8,11,13,16,19,20
630 CALL CLEAR
640 CALL SCREEN(3)
650 CALL COLOR(2,7,16)
660 CALL COLOR(9,2,11)
670 CALL COLOR(1,12,3)
680 RESTORE 620
690 FOR I=1 TO 24
700 READ J
710 CALL HCHAR(I,J,40,8)
720 CALL HCHAR(I,J+1,41,6)
730 NEXT I
740 CALL HCHAR(7,17,96)
750 RANDOMIZE
760 J=18
770 X=17
780 G=41
790 FOR I=1 TO 75
800 IF I>59 THEN 860
810 J=J+R
820 IF J<21 THEN 840
830 J=21
840 IF J>1 THEN 860
850 J=1
860 PRINT TAB(J);"()))))))("
870 CALL SOUND(-100,110,1,-1,1)
880 CALL HCHAR(6,X,G)
890 CALL KEY(0,K,S)
900 IF (K<>83)+(K<>68)=-2 THEN 990
910 IF K=83 THEN 960
920 X=X+2
930 IF X<31 THEN 990
940 X=31
950 GOTO 990
960 X=X-2
970 IF X>2 THEN 990
980 X=2
990 CALL GCHAR(7,X,G)
1000 IF G=41 THEN 1060
1010 IF G=96 THEN 1060
1020 IF G=32 THEN 1180
1030 CALL SOUND(-50,-5,0)
1040 CALL HCHAR(7,X,97)
1050 C=C+1
1060 CALL HCHAR(7,X,96)
1070 NEXT I
1080 CALL HCHAR(22,1,32,64)
1090 PRINT "YOU MADE IT:"::"NUMBER OF CRASHES:";C
1100 IF C>0 THEN 1220
1110 DATA 330,392,523,659,523,659,659
1120 RESTORE 1110
1130 FOR I=1 TO 7
1140 READ S
1150 CALL SOUND(150,S,0)
1160 NEXT I
1170 GOTO 1220
1180 CALL SOUND(200,-5,0,400,0)
1190 CALL HCHAR(7,X,97,2)
1200 CALL HCHAR(22,1,32,64)
1210 PRINT "SORRY; THE CAR IS DAMAGED":
     :"BEYOND REPAIR"
1220 GOSUB 510
1230 PRINT ::"PRESS ANY KEY"
1240 CALL KEY(0,K,S)
1250 IF S<1 THEN 1240
1260 CALL CLEAR
1270 CALL COLOR(2,2,1)
1280 CALL COLOR(1,2,1)
1290 GOTO 250
1300 PRINT TAB(6);"** MUIR WOODS **"
```

Continued on p. 84

Space Station 1

Reviewed by Sam Pincus
Contributing Editor

Author:	Dominic Melfi
Program type:	Arcade "shoot 'em up"
Language:	TMS9900 Assembler Language
Distributor:	Data Force Inc. 10 S. 312 Hampshire Lane Hinsdale, IL 60521
Price:	$34.95, disk

The latest TI arcade games are written in 9900 Assembly Language, and demonstrate the tremendous speed that the TI-99/4A is really capable of. But Assembly Language Programming on the home computer is not the sole domain of TI—at least, not any longer, with two assemblers presently available and a third one on its way [See the article on the *Mini Memory Module* in this issue—Ed.] I was therefore extremely interested in seeing the first Assembly Language game program written by someone outside Texas Instruments.

The game is called *Space Station 1* and was written by Dominic J. Melfi. It is presently available on disk from Data Force Inc. for $34.95, and because it's in Assembly Language, it requires Extended BASIC and the 32K Memory Expansion peripheral. [Data Force has informed us that a cassette-based version may also be released—Ed.]

Loading the game is quite simple. Just place the disk in Drive 1 and turn on your TI-99/4A. The BASIC "boot" program is automatically called in; it in turn calls a special assembly load program. This special load program then loads the actual program.

Why all this rigamarole just to load the program? Because, as the author says, "It wasn't fair to make anyone wait the 3 or 4 minutes that TI's load program would require for a program this size." How fast is Mr. Melfi's program loader? To find out, I ran a time comparison of the two loaders. Using the Extended BASIC CALL LOAD routine written in GPL (Graphics Programming Language), it took 3 minutes and 25 seconds to load the game after the Extended BASIC program started. Using the special load routine, it took just 20 seconds from power-up to the start of the game! This is indicative of the pride and care taken by the author in providing the best possible product for the user.

The game disk comes together with 5 pages of documentation. The first 3 pages present the space game's scenario-providing the "history" for the years 2000-2020, and then explaining what the purpose of the game is and why it reacts as it does. The last two pages explain how to play the game. The basic idea of the game is quite simple: Your job is to protect a space station which is shown revolving in the middle of your screen.

Missile-firing alien ships attack in groups of three. You have four torpedoes with which to shoot down the aliens, by using a joystick or the arrow keys on the keyboard to control a target cross-hair. Pressing the "fire" button or the ENTER key launches a torpedo. As soon as your torpedoes are used up, they are replenished. After each wave is destroyed, another wave of aliens attack. They come from various directions—to the left, right, above and below the space station. In addition, a mother ship (usually invisible) runs across the top of the screen, and at certain times, it is subject to attack.

To tell you the truth, I skipped very quickly over the documentation, and instead went right to the game. This was a big mistake because I immediately got lost trying to follow the game. I suggest that you do read the documentation before you play. And read it *carefully*! It's quite possible to miss some important details that will affect your understanding of the game. This is really the

Continued on p. 79

Adventure Registry

Pyramid of Doom

An Adventure Successfully Completed

By Donald L. Wells
902 Doral Lane
Houston, TX 77073

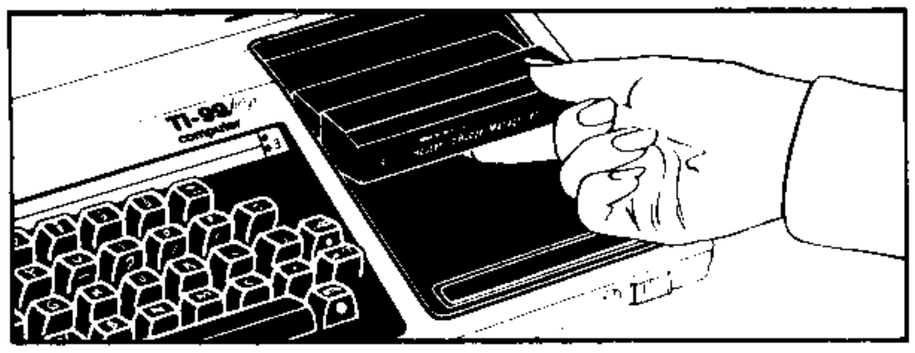
I'm in a desert. Visible items are a pool of liquid and a wooden pole sticking out from the sand. I'm carrying an empty canteen and an unlit flashlight. My obvious exit is Northeast. What shall I do?

Thus begins *Pyramid of Doom*, an adventure game from Texas Instruments and Adventure International—a game I was all set *not* to enjoy because it was all text and no graphics . . . I had seen adventure-type games on other systems with graphics that change as the story changes, so I thought, "Now how much fun can an adventure be without any graphics?"

Well, when I plugged in the master Adventure Command Module and loaded in the *Pyramid* data base, the completely unexpected discovery I made was something I really should have anticipated from my earlier experience with radio serials: *The*

Continued on p. 56

SAVE UP TO 20% ON SOFTWARE AND PERIPHERALS FOR YOUR TEXAS INSTRUMENTS HOME COMPUTER.



Package Title	Command Module	Diskette	Cassette
Home Management Personal Finance			
Home Financial Decisions	$ 29.95	—	—
Household Budget Management	39.95	—	—
Securities Analysis	54.95	—	—
Personal Record Keeping	49.95	—	—
Tax/Investment Record Keeping	69.95	—	—
Personal Real Estate	69.95	—	—
Personal Report Generator	49.95	—	—
Mailing List*	—	49.95	—
Personal Financial Aids	—	19.95	—
Checkbook Manager	—	19.95	—
Bus. Aids Lib. - Inventory Mgmt.	—	69.95	—
Bus. Aids Lib. - Invoice Mgmt.	—	69.95	—
Bus. Aids Lib. - Cash Mgmt.	—	39.95	—
Bus. Aids Lib. - Finance Mgmt.	—	39.95	—
Bus. Aids Lib. Lease/Purchase Decisions	—	69.95	59.95
Personal Financial Aids	—	—	14.95
Accounts Payable*	—	99.95	—
Accounts Receivable*	—	99.95	—
General Ledger*	—	99.95	—
Household Inventory System*	—	34.95	—
Inventory Management*	—	99.95	—
Mailing List*	—	49.95	—
Payroll*	—	99.95	—
Word Processing*	—	99.95	—
Education/Personal Enrichment			
Early Learning Fun	29.95	—	—
Beginning Grammar	29.95	—	—
Number Magic	19.95	—	—
Video Graphs	19.95	—	—
Video Chess	69.95	—	—
Physical Fitness	29.95	—	—
Early Reading †	54.95	—	—
Music Maker	39.95	—	—
Weight Control and Nutrition	59.95	—	—
Addition and Subtraction I †	39.95	—	—
Addition and Subtraction II †	39.95	—	—
Multiplication I †	39.95	—	—
TI LOGO (Requires Mem. Exp. Unit)	129.95	—	—
Reading Fun †	54.95	—	—
Teach Yourself BASIC	—	34.95	29.95
Music Skills Trainer	—	29.95	24.95
Computer Music Box	—	19.95	14.95
Market Simulation	—	19.95	14.95
Teach Yourself Extended BASIC	—	24.95	19.95
Music Maker Demonstration	—	14.95	—
Basketball Statistics	—	24.95	—
Bridge Bidding I	—	29.95	24.95
Speak & Spell Program	—	29.95	—
Speak & Math Program	—	29.95	—
Bridge Bidding II	—	29.95	24.95
Bridge Bidding III	—	29.95	24.95
Spell Writer	—	29.95	24.95
Scholastic Spelling Level III, IV, V, and VI	54.95 ea	—	—
Beginner's Basic Tutor	—	29.95	24.95
Entertainment			
Football	29.95	—	—
Video Games I	29.95	—	—
Hunt The Wumpus	24.95	—	—
Indoor Soccer	29.95	—	—
Mind Challengers	24.95	—	—
A-Maze-Ing	24.95	—	—
The Attack ††	39.95	—	—
Blasto ††	24.95	—	—
Blackjack and Poker	24.95	—	—
Hustle ††	24.95	—	—
Zero Zap ††	19.95	—	—
Hangman ††	19.95	—	—
Connect Four ††	19.95	—	—
Yahtzee ††	24.95	—	—

Package Title	Command Module	Diskette	Cassette
Adventure (Pirate Adventure Diskette or Cassette Game included)	49.95	—	—
Adventure Series			
*Adventureland	—	29.95	29.95
*Mission Impossible	—	29.95	29.95
*Voodoo Castle	—	29.95	29.95
*The Count	—	29.95	29.95
*Strange Odyssey	—	29.95	29.95
*Mystery Fun House	—	29.95	29.95
*Pyramid of Doom	—	29.95	29.95
*Ghost Town	—	29.95	29.95
*Savage Island I & II	—	39.95	39.95
*Golden Voyage	—	29.95	29.95
Tombstone City: 21st Century	39.95	—	—
TI Invaders	39.95	—	—
Car Wars	39.95	—	—
Munch Man	39.95	—	—
TI-Trek (w/speech)	—	14.95	—
Mystery Melody	—	14.95	9.95
Oldies But Goodies - Games I	—	19.95	14.95
Oldies But Goodies - Games II	—	24.95	19.95
Saturday Night Bingo	—	29.95	24.95
Draw Poker	—	24.95	19.95
All Star Baseball*	—	—	19.95
All Star Bowling*	—	—	19.95
Casino Pack*	—	19.95	—
Challenge II*	—	—	16.95
Galactic War*	—	21.95	19.95
SAM Defense*	—	—	19.95
Wall Street*	—	—	19.95
London Blitz*	—	—	19.95
Starship Concord*	—	—	19.95
Others			
Speech Editor	44.95	—	—
Statistics	44.95	—	—
Extended BASIC	99.95	—	—
Terminal Emulator II	49.95	—	—
Editor Assembler	99.95	—	—
Mini Memory	99.95	—	—
Programming Aids I	—	14.95	9.95
Programming Aids II	—	24.95	—
Math Routine Library	—	29.95	24.95
Electrical Engineering Library	—	29.95	24.95
Programming Aids III	—	19.95	—
Graphing Package	—	19.95	14.95
Structural Engineering Library	—	29.95	24.95
UCSD Pascal Compiler	—	124.95	—
UCSD P-System Assembler/Linker	—	99.95	—
UCSD P-System Editor/Filer/Utilities	—	74.95	—
TI Pilot	—	79.95	—
Course Designer Authoring System	—	199.95	—

Software Libraries	Price
The Home Financial Manager	139.95
The Family Entertainer	89.95
The Elementary Educator	99.95
The Music Educator	64.90
The Super Programmer	119.00
The Scott, Foresman Speaking Math Teacher	119.85
The Scott, Foresman Speaking Reading Teacher	109.90
The Speaking Scholastic Spelling Teacher	219.80
The TI Arcade Game Series	119.85
The Milton Bradley Game Series	114.75
The Computer Introductory Package	119.85

† Developed by Scott Foresman *†† Developed by Milton Bradley* ** Developed by Ehninger Associates*

Peripherals/Accessories	Price
TI-99/4A Console	$450.00
Disk Memory Drive	499.95
Disk Drive Controller	299.95
Solid State Printer	399.95
Memory Expansion (32K RAM)	399.95
P-Code Peripheral	399.95
RS-232 Accessories Interface	224.95
Telephone Coupler (Modem)	224.95
Solid State Speech Synthesizer	149.95
RF ModulatorTV Adapter	49.95
Periphral Expansion Box	$249.9*
RS-323 Card	$174.95*
Disk Controller Card	$249.95*
Expansion Box Disk Drive	$399.95*
Memory Expansion Card	$299.95*
P-Code Card	$249.95*
Wired Remote Controllers	34.95
Dual Cassette Cable	14.95
Monitor Cable	19.95
Audio Adapter (Headphone Jack)	19.95
Thermal Paper (2-Pack)	9.95
Video Controller (includes choice of cables)	699.95
Panasonic, Sony, or Pioneer Video Controller Cables	ea. 99.95
Printware	
Creative Programming Computer Competency Series—Volume I, II, III, or Allstar Projects	ea. 9.95

HERE'S HOW YOU CAN SAVE!!

Order Amount	% Discount
$30 - $99	10%
$100 - $249	12%
$250 - $749	15%
$750 - Up	20%

ORDER HOTLINE-TOLL FREE

1-800-858-4580

In Texas Call 1-806-745-8835

Just give us your name, shipping address and Visa or MasterCard number and we'll charge the full amount to your account. Be sure to include your purchase price, $2.00 shipping and handling for software (or $5.00 for peripherals) and 3% for Visa or MasterCard orders. Texas residents also add 5% sales tax. For mail-in orders, send to P.O. Box 64240, Lubbock, Texas 79464.

Unisource Electronics, Inc.

Spotlight . . . from p. 48

9900 code. I was really impressed with the capabilities of the home computer—in the speed we could actually get from the 9900. I now realize that graphics are probably the most important thing in a game. I think you can probably be all right by throwing graphics on the screen and designing the game around your graphics rather than designing the game first, and then doing the graphics. Sometimes it works better one way, sometimes the other. But a lot of times your graphics influence how the game is going to be played.

GMK: *Does this reasoning also hold true for sound effects?*

JCP: To me, sound isn't that important. The sound, I think, can be added as an afterthought. If you have a game that plays well, I think that should be the thing that holds the interest of the user. The sound should be just a fringe benefit.

GMK: *What are your personal high scores on each of the three levels?*

JCP: On Level 1, I never really played for a score—it's really a slow level. On Level 2, my high score was 528,000 occuring on Day 42. On Level 3, my high score was 136,500. That occured on Day 13.

GMK: *Do you think it's possible for other players—without the intimate knowledge of the game that you obviously have—to score this high?*

JCP: Yes. Once you reach a certain point on Level 2, your score can almost go on indefinitely. On Level 3, I was never able to get past Day 13. I think that a person would have to spend a lot of time to get a high score on Day 13, although it probably isn't impossible.

GMK: *What final tips can you offer to players of the game?*

JCP: The whole key for beginning players is to kill the monsters *immediately* when they've generated. This [advice] also goes for any player, regardless of skill. You must know beforehand *when* the monster is coming out, and exactly *where* it is coming out from so that you can immediately shoot it to destroy the generating pair. For the more advanced player, keeping the safe area clear is probably the most important thing; you can't afford to kill the monsters next to the safe area. If this happens, try to set it up where you can use your "dragging" tactic to move the cactus away from the safe area.

GMK: *One final question. In Level 3, how can you stay alive the longest?*

JCP: In Level 3, you just cannot afford to make mistakes—you cannot kill the monster and form another generating pair. If you start forming other generating pairs by your mistakes, it becomes almost irrecoverable after you get so many on the screen . . .

Memory . . . from p. 23

DSRLNK Routines

DSRLNK links an assembly language program to a Device Service Routine (DSR) or subprogram in ROM. As with GPLLNK and XMLLNK, TI cautions you to make sure you know what you are doing before using DSRLNK.

[A DSR is a machine language program that TI has burned into ROMs found in each of its peripherals. Since each peripheral contains its own custom "operating system," the TI-99/4 did not have to be designed to anticipate future peripheral requirements—Ed.]

TI BASIC Interface Utilities

TI BASIC interface utilities allow an assembly language program to read or assign values to variables passed in a parameter list from a CALL LINK statement in a TI BASIC program. These utility routines include argument-passing utilities and an error-reporting utility.

The following are the TI BASIC interface utilities.

- Assign a numeric value to a numeric variable.
- Assign a string to a string variable.
- Retrieve the value of a numeric parameter.
- Retrieve the value of a string parameter.

– Report an error. (The assembly language program can report any existing TI BASIC error or warning message upon returning to TI BASIC.

EASY BUG Debugger

Also inside the Mini Memory Module's ROM is EASY BUG. EASY BUG is a versatile program development tool with which you can (1) debug your assembly language programs, (2) access the input/output ports of the computer, (3) load programs, and (4) store programs. And, as the name implies, it really is easy to use. With EASY BUG, you can inspect and (optionally) modify the contents of CPU and VDP memory, display the contents of ROM, run assembly language programs from EASY BUG, directly access the peripheral devices which are connected to the computer via the 9900 microprocessor's serial I/O port (the CRU), and save and load programs on cassette.

LINE-BY-LINE SYMBOLIC ASSEMBLER

A line-by-line symbolic assembler on a cassette tape is supplied with the Mini Memory Module. It assembles assembly language statements and stores the object code directly into the 99/4's CPU RAM. The assembler is a derivitive of the line-by-line assembler used by the TM990/189 board (the University Module). You can make both forward and backward references to one- or two-character labels with the assembler. Each source statement you enter is immediately assembled into object code and stored into memory. Some source code is retained in a nine-page text buffer. You can scroll the screen to review previously entered lines of source code by pressing the up- and down-arrow keys. The source program cannot be saved, however.

The line-by-line assembler occupies about 2K bytes. When it is loaded into the Mini Memory Module's 4K byte RAM, you still have about 2K bytes of memory for your assembly language program.

Assembler Directives

The assembler recognizes seven directives:

– The AORG (Absolute Origin) directive establishes the location counter value to set the starting address of the assembled code.

– The BSS (Block Starting with Symbol) directive reserves a block of memory without initializing the space.

– The DATA (Data Initialization) directive intializes a word or words of memory to a specific value.

– The END (End Program) directive terminates the assembler and causes a display of the number of unresolved references, if any.

– The EQU (Equate) directive defines a value for a symbolic constant.

– The SYM (Symbol Table Display) causes a display of all symbols and their associated values in the program.

– The TEXT (String Definition) directive causes a string of characters to be translated into their ASCII code and stored as a part of a program.

[An assembler "directive" is a programming-aid command which directs the assembler to perform certain operations at assembly time. An assembler may execute many "instructions" (telling the microprocessor to perform single functions such as Add or Move) to satisfy one directive–Ed.]

DEMONSTRATION PROGRAM

Along with the line-by-line assembler on the cassette is an assembly language demonstration program called LINES which draws a colorful line design on the screen. The LINES program can be run only on the TI-99/4A Home Computer, however, because it requires the enhanced graphics processor contained in the TI-99/4A. [A forthcoming issue will carry an explanation of how the "bit map" mode (necessary for the LINES program) of the TMS9918A Video Display Processor works–Ed.]

OPERATION

TI has a knack for creating complex and versatile programs that are still simple to operate; they've definitely done it again with the Mini Memory Module. When you plug in the module, turn on the computer, and pass the opening credits on the Master Title Screen, you are presented with a simple, three-choice selection screen. You can choose TI BASIC, EASY BUG, or MINI MEMORY.

If you select MINI MEMORY, you are presented with a second three-choice selection screen. You can choose to load an object program into memory and run it, run a previously loaded program already in memory, or re-initialize the module to prepare it for loading new programs or storing data. Pick a number, pluck a key, and you're off and running. It's as easy as eating oatmeal cookies!

CONCLUSION

This has got to be one of the best deals around. There's 4K bytes of RAM with battery backup so that all the good stuff stored in the RAM is not lost when you turn off the console or even when you remove the module. There's 10K bytes of ROM and GROM which give you seven additional TI BASIC subprograms (including PEEK and POKE), access to system routines from assembly language, and routines to allow you to interface assembly language programs to TI BASIC. You've got a user-friendly program debugger, a symbolic line-by-line assembler, and a captivating graphics demonstration program. All of this, plus 84 pages of documentation for $99.95 (suggested retail price). With all this to offer, it's really not too hard to see why there's definitely more to the Mini Memory Command Module than meets the T-eye . . . 99'er

*Ad placed this way on advertiser request.

Business . . . from p. 32

word REC and the record number of the record to be written:

```
PRINT #2,REC N: PN,D$,Q,PR
```

You can use the EOF function for a random-access file, but this is not the best way. Better is to use the zero'th record to hold special information about the file, expecially the length of the file. As soon as you open the file, read that record:

```
INPUT #2, REC 0: FL
```

Then, before accessing any record, compare its record number with FL:

```
230 IF N>FL THEN 260
240 INPUT #2,REC N: PN,D$,
    Q,PR
250 GOTO 280
260 PRINT "INVALID RECORD
    NUMBER. REENTER"
    . . .
```

When we wish, we are allowed to read or write records sequentially in a random-access file. And of course we should CLOSE a file at the end of the program.

Which Record Contains What?

Okay, so you can easily get the 119th record in your random-access file. But how do you know that the information you want is in the 119th record? This is the hard part. If you are willing to assign product numbers 1 to 200 to the 200 items in your inventory file, you have no problem. At least, not until you discontinue some products and add others. In many cases, you can't assign the key to your file (product number, social security number, account number, or whatever) like this at all. So we need some scheme that associates a record number with each of your keys.

There are a lot of ways to do this. I will show one here, and show a very different one, perhaps in the next issue. I invite you clever readers to write me, c/o *99'er Magazine*, with schemes you are proud of using. I would be happy to show your schemes too.

My first scheme is an index, which I keep in a file of its own. Actually, it could be kept in the first several records of your random-access file if you wish. Let's suppose an inventory system, with up to 200 products. The product numbers are already assigned, as integers, like 17, 29, 83, 104, 105, etc. We can keep our index in a pair of arrays in main storage while we run our system: these arrays don't take a lot of room.

```
60 DIM IPN(200),ILOC(200)
70 OPEN #1: "DSK1.INVINDEX"
   ,SEQUENTIAL,INTERNAL
   . . .
180 FOR I=1 to 200
190 INPUT #1: IPN(I),ILOC(I)
200 NEXT I
```

The IPN array holds the product numbers, and the ILOC array the record numbers in the random-access file for the corresponding products. When we want to access a product, we search the IPN array, find the record number, then use it to access the product record directly.

Using this scheme, the sequence of records in the random-access file matters very little. The sequence in the index file (and therefore in the arrays) matters more. The easiest thing, but least efficient, is to search the IPN array sequentially, with the product numbers either in ascending sequence or in no particular sequence. One bettter idea is to put the most frequently used records at the front of the index file, thus cutting down on the average number of index entries your program must search. Studies have shown that in situations like this, 80% of the desired accesses are to 20% of the items. A still more efficient (but longer to program) method is a binary search, requiring that the index be in ascending sequence by product

number. But let's come back to that idea another time.

Putting It All Together

Let's see how some of this works. We will see, at least in outline, how to (1) update a particular record, using the index, and (2) how to add a new record to the file (and of course to the index). Then in the next issue, we will see a more comprehensive program, using the index idea, as well as one using a different scheme.

First, let's be a little more precise about how we keep information, again using an inventory system as the context.

1. The RELATIVE file is named INVENTORY; its first record (numbered 0) contains the allocated length of the file; the number of records actually used must not exceed that number. If the allocated length is 201 records, for example, we might at some time be using 160, and these would be numbered 1 to 160.

2. The index file is named INVINDEX; it contains an index entry for each of the allocated records in INVENTORY. The index entries are in sequence by product number. The unused records are identified in the index by a product number like 32767, which is larger than any actual product number. In addition, at the very beginning of the INVINDEX file are

   (a) the number of allocated records

   (b) the number of currently active records

As part of our program initialization, we must open the files and read the index into our arrays:

```
60 DIM IPN(200),ILOC(200)
70 OPEN #1: "DSK1.INVINDEX"
   ,SEQUENTIAL,INTERNAL
80 OPEN #2: "DSK1.INVENTOR
   Y",RELATIVE,INTERNAL
   ,UPDATE,FIXED 92
90 INPUT #1: NALLOC,NACTV
100 FOR I=1 TO NALLOC
110 INPUT #1: IPN(I),ILOC(I)
120 NEXT I
```

Now, suppose the program has accepted a product number APN, and needs to retrieve the INVENTORY record for that product; we will show for simplicity a sequential, rather than a binary search through the index file:

```
310 FOR J=1 TO NACTV
320 IF APN=IPN(J) THEN 370
330 IF APN>IPN(J) THEN 350
340 NEXT J
350 PRINT "PRODUCT NOT ON
    FILE"
360 GOTO . . .
370 INPUT #2,REC ILOC(J): PN,
    D$,Q,PR, . . .
```

If the program goes on to update some fields of the record, the record can be rewritten with its updated contents very simply:

```
470 PRINT #2,REC ILOC(J): PN,
    D$,Q,PR, . . .
```

Inserting a new record for a new product number is a little tricky. Where to put it in the inventory file is no problem; it can go right after the last active record. The index will make it accessible at the right time, with no problem. But we have more to do with the index. We must insert a new index entry in its proper place in sequence. Let's look at that process. Suppose that the product number to be inserted is PN, and that we have ascertained that such a number is not in the file.

```
600 IF NACTV<NALLOC THEN
    630
610 PRINT "NO MORE SPACE
    IN THE INVENTORY FILE"
620 GOTO . . .
630 NACTV=NACTV+1
640 PRINT #2,REC NACTV: PN,
    D$,Q,PR
```

Continued on p. 56

Business . . . from p. 55

```
650 REM ADJUST INDEX
660 FOR J=1 TO NACTV
670 IF PN>IPN(J) THEN 690
680 NEXT J
690 FOR K=NACTV TO J+1
    STEP -1
700 IPN(K)=IPN(K-1)
710 ILOC(K)=ILOC(K-1)
720 NEXT K
730 IPN(J)=PN
740 ILOC(J)=NACTV
750 REM REWRITE THE UP
    DATED INDEX FILE
760 RESTORE #1
770 PRINT #1: NALLOC,NACTV
780 FOR I=1 TO NALLOC
790 PRINT #1: IPN(I),ILOC(I)
800 NEXT I
```

None of these operations takes very long. We always have the index file, the index arrays, and the random-access file itself in sync.

Do you have a better scheme? You may very well have, especially for your particular application. There is a lot of room for different ways of using and managing random-access files. After all, what we have is really the capability of managing large arrays—kept on disk instead of main storage. I hope you can see the importance of, and get some idea of how to use, random-access files from this introduction. We will continue with a fuller example in the next issue. And I would be very happy indeed if some of you would share with me *your* schemes for managing random-access files.

Pyramid . . . from p. 50

words provide the story line, but what's in your head provides the graphics! Now, by comparison, the graphics on that other system seem silly and childlike in their simplicity. And although I'm sure that *your* pyramid probably does not look a thing like *my* pyramid, I am equally sure that *yours* is the perfect one for *you*.

So here we are in the middle of the desert with an empty canteen and an unlit flashlight. If you go East, you find yourself at the pyramid. Fine, but now what? The "now what" is just the beginning of a truly fascinating journey—that is *if* you ever figure out how to get into the pyramid. And watch out for that small desert nomad! I wouldn't turn my back on him too often if I were you . . .

Once inside the pyramid, the mystery intensifies as you track down the treasures and accumulate points. There is a sarcophagus, a giant oyster, and a mad mummy to contend with if you are to find all the treasures. The *Pyramid of Doom* will provide a strenuous workout for your powers of logic and deductive reasoning—pitting you up against a very intriguing and demanding maze of clues and trails. When you finally drop that last treasure in the right spot and say SCORE—and are thus rewarded with the yellow screen—the euphoria you experience over this accomplishment is unbelievable. Get lucky, and you might finish *Pyramid* in a few hours; but a few *days*—or even *weeks*—is the more likely case, however.

One of the nicest features of the Texas Instruments implementation of the Adventure Series (consisting of eleven Adventures) is the ability to save the game at the point where you are, and to pick up again right at the point where you left off. This is also very handy for going into places where you *know* you shouldn't go . . . Just save the game up to that point, then go ahead and jump off the cliff or into the pit! If you meet an untimely end, you can pick up where you left off, remembering to avoid that hazard the next time around.

Pyramid of Doom is available on cassette tape (PHT 6052) or disk (PHD5052) for the suggested retail price of $29.95 and requires the *Adventure* Command Module at a suggested retail of $49.95 (which includes a free *Pirates Adventure* on cassette tape (PHM3041T) or disk (PHM 3041D).

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

LOGO TIMES

T.M.

THE MAGAZINE OF THE LOGO LANGUAGE

Avoiding Turtle Traps

Problem Solving With LOGO

TI LOGO With A Little T.L.C.

Building With LOGO:

Tinkertoy of the Mind

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, they 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

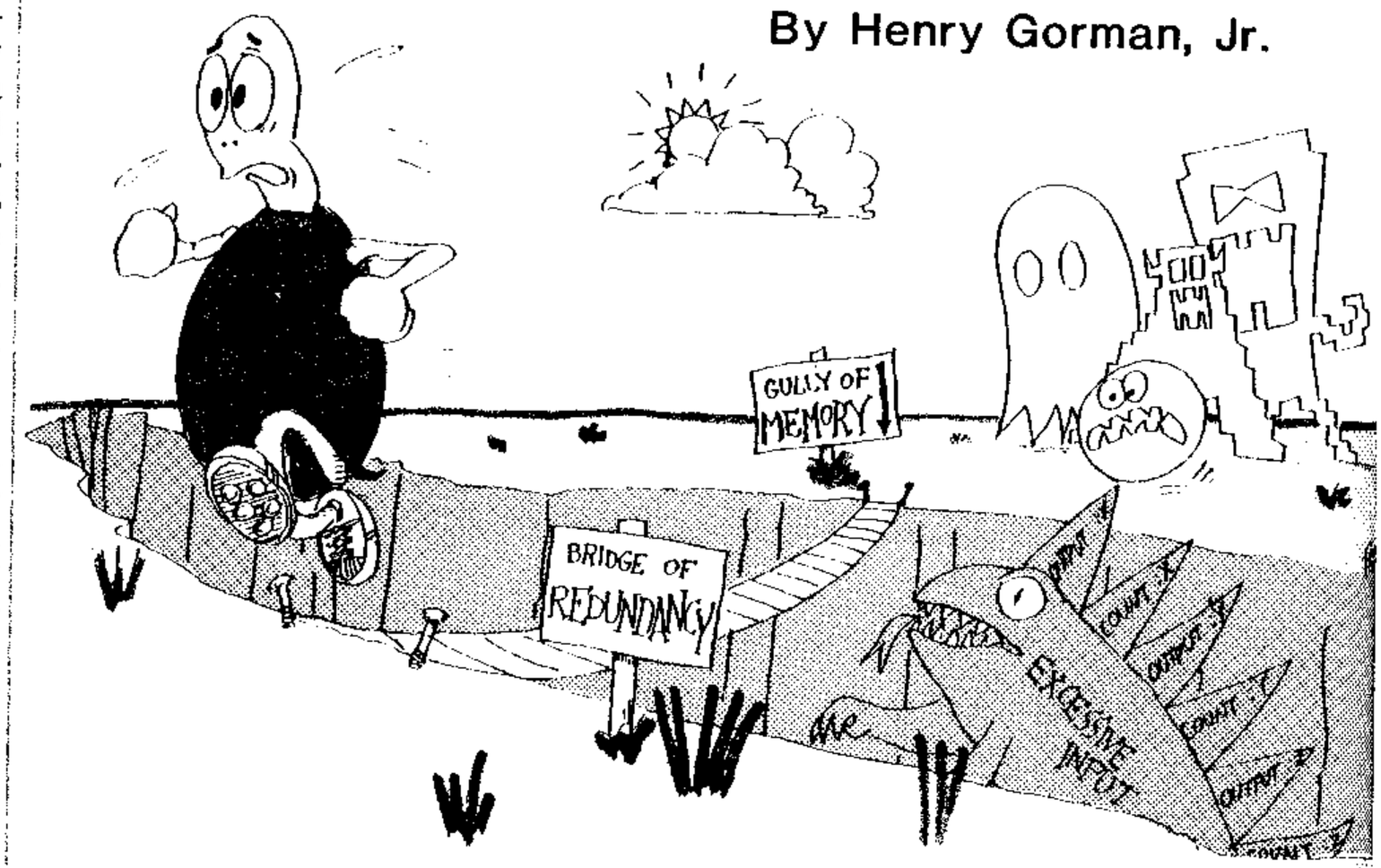
Roger B. Kirchner
Department of Mathematics
Carleton College
Northfield, MN 55057

James H. Muller
Young Peoples' LOGO Association
1208 Hillsdale Drive
Richardson, TX 75081

AVOIDING TURTLE TRAPS:

Writing Better LOGO

By Henry Gorman, Jr.



Seymour Papert and his colleagues purposefully decided to structure LOGO to facilitate the writing of good computer programs. The concept of good programming is not superficially apparent. Of course, a program should accomplish its intended goal, but all programmers recognize that any goal can be achieved by many different types of programs. Beyond simply "working," there are a number of criteria by which programs can be judged. Programs which have multiple applications are generally better than single-purpose programs. Programs which are easier to debug and which can be understood by people other than the authors (or which can be understood by the authors at a future time) are more desirable. Pragmatically, programs which run faster or with fewer bits of memory are better than slower or more memory-intensive programs. Finally, some programs are esthetically more appealing than others.

It is possible to find examples of program applications in which one or more or the criteria are in conflict. However, it is more often the case that the criteria are in accord. All of the criteria except for esthetics are straight-forward and relatively objective. Still, writing esthetic programs is so satisfying that esthetics should be considered first here.

Programs with many different branches from *GO* commands (yes, you can write GO in Logo, but why write poor programs when good ones are easier to write) are particularly "inelegant." Also inelegant are programs with hundreds of lines of code, especially when that code contains several repetitions of a series of commands. Programs with many inputs are generally less esthetic than those with fewer inputs. Compare the esthetics of two programs which count the number of words in a list:

```
TO COUNT :N :LIST
IF BUTFIRST :LIST = [ ]
OUTPUT :N
COUNT :N + 1 BUTFIRST
:LIST
END
```

This program requires typing as input along with COUNT and the list in question a starting value of :N — 0.

```
TO COUNT :LIST
IF :LIST = [ ] OUTPUT 0
OUTPUT (COUNT BUTFIRST
:LIST) + 1
END
```

This program requires no superfluous input.

In the last issue of *99'er Magazine*, Roger Kirchner presented a fairly complex program, DYNATURTLE, which created a turtle that obeys Newtonian laws of motion. Despite the complexity of the program, DYNATURTLE is relatively elegant: DYNATURTLE only has the three lines which are INITIALIZE, SETDYNATURTLE, and CONTROL. Each of those lines is, in turn, a brief program which serves a unique function. Contrast DYNATURTLE's elegance

with a "spaghetti-pole" BASIC program which would achieve the same effects. Such a program would be long and littered with extensive GO-TO's.

A subtler example of elegant and inelegant programs can be made from my GRAMMAR program in the last issue of *99'er*. The program was modified from an earlier POET program and was written:

```
TO GRAMMAR :ART :NOUNS
:ADJ :VERBS
TYPE SELECT (NUMB (COUNT
:ART)) :ART
SPACE
TYPE SELECT (NUMB (COUNT
:NOUNS)) :NOUNS
SPACE
TYPE SELECT (NUMB (COUNT
:VERB)) :VERBS
SPACE
TYPE SELECT (NUMB (COUNT
:ADJ)) :ADJ
SPACE
TYPE SELECT (NUMB (COUNT
:NOUNS)) :NOUNS
PRINT"
PRINT".
WAIT 30
GRAMMAR :ART :NOUNS
:ADJ :VERBS
END
```

Notice how much of each line is repetitive. A better Logo program would have been to take advantage of that redundancy and use a broader application program:

```
TO WORDS :X
TYPE SELECT (NUMB (COUNT
:X)) :X
SPACE
END
```

Then GRAMMAR could be written:

```
TO GRAMMAR :ART :NOUNS
:ADJ :VERBS
WORDS :ART
WORDS :NOUNS
WORDS :VERBS
WORDS :ADJ
WORDS :NOUNS
PRINT".
WAIT 30
GRAMMAR :ART :NOUNS
:ADJ :VERBS
END
```

The second GRAMMAR program is more elegant and is shorter. It achieved greater simplicity by taking out of GRAMMAR all of the repeated functions and placing them in WORDS. All of the functions achieved by the program WORDS are directed at placing a single word from a designated set of words. The specification of the set and type of words is left for the program surrounding WORD, GRAMMAR. A common format for many well-written Logo programs is:

```
TO DOSOMETHINGSPECIFICALLY
    :SPECIALINPUT
GENERALPURPOSEPROGRAM
    :GENERALINPUT
END

TO GENERALPURPOSEPROGRAM
    :GENERALINPUT
LOGO commands :GENERALINPUT
END
```

On occasion it is necessary to string together several general-purpose programs inside a specific-purpose program. When that is the case it is often required by the general program that there be some set-up steps and some "fix-up" steps before and after the general program. Such programs have a form:

```
TO GENERALPURPOSE
SETUP
GENERALFUNCTIONS
FIXUP
END
```

Mathematicians may indeed recognize a similarity between the concept of elegance and esthetics in programming and the expression of algebraic functions. There are many ways to express algebraic functions, but it is often more useful and always more elegant to express such functions in a form which collects common factors and simplifies terms even where such simplification may require a "set-up" or a quick "fix-up" manipulation along with the factoring.

The second major aspect to writing better Logo programs is that of writing programs which do not run out of memory and which run as quickly as possible. It is important to understand the major feat accomplished by Texas Instruments and by the MIT LOGO Lab in putting LOGO on the 99/4. LOGO is a very high level computer language which requires large amounts of memory. The architecture of microcomputers limits the speed with which large amounts of memory can be addressed. The TI LOGO which emerged from the joint efforts of TI and MIT represents an effort to compress the code to the minimum memory requirement without compromising its applications. There are two "tricks" which they built into TI LOGO to make LOGO feasible on a micro. If you use these tricks you can gain even greater satisfaction from your

Continued on p. 64

Student Reactions to a Four Week LOGO Class

By Gene Branum

The expectations of the students varied—we wanted to know more about computers, we wanted a different Jan-term experience, or maybe just a free Jan-term. Whatever the motivation, all came away affected in some way by our experience. All experienced both the frustration of failure and the flush of triumph as the computer finally "did what it was supposed to."

The format for our experience was a four-week mini-term (Jan-term) at Austin College. Our class met five days a week for two hours, and we were required at least one hour of work on our own as well. This requirement was easily met and as one student put it, "it was not unusual to spend four hours at a time" on the computer. Needless to say, the experience was very intense, and there was a great deal of self-teaching. This was felt to be one of the greatest strengths of the course.

Professor Hank Gorman did a fine job of teaching the basics early in the course. As he told us his expectations, we scoffed. After two weeks, he told us we would be drawing cartoons and making up games. Even though his leadership was great, the majority were insecure about "the machine." Our confidence, however, grew with experience and familiarization.

The two greatest aspects of the course for all of us were (1) the team experience and (2) experience in general problem solving skills. The true strength of LOGO is that students, working together, can teach each other massive amounts of material. The realization that *everyone* had problems put us all on the same level. Sharing ideas and solutions became important for everyone because no one could work totally independently. Many social experiences allow students to interact, but LOGO is one of the few that forces students to *think together.*

Without exception, all of the students involved in the course commented that, after LOGO, they knew better how to approach a complex problem. Dr. Gorman spent several class periods on problem solving skills such as decomposition, recursion, naming, multiple discriptions, and the "little men." These skills not only aided our search for solutions to LOGO problems, but any problem that requires a thinking solution. The overriding principle of LOGO is that the simple builds to the complex, which is its major strength as a system for any age-group.

While it was widely agreed upon that none of us "mastered" LOGO, each of us developed confidence in our abilities to control the computer and make it do what we requested. The LOGO experience allowed everyone to use logical approaches to problem solving and gain valuable hands on experience in a discipline that continues to increase in importance.

[Examples of the students work are at the conclusion of "Avoiding Turtle Traps" on pages 64 and 65.]

PROBLEM SOLVING WITH

One of the pleasures of having a language like LOGO to work with is that it gives us something to "think with," and it encourages us to think in what Papert has called "mind sized bites." The solution of a problem can be identified with the definition of a procedure. If the problem is simple, we can specify the procedure directly. Otherwise, we try to specify it in terms of a small number of simpler procedures.

Often, this method leads to a complete solution of a problem. But sometimes, a problem is so complex that the method leads to an indefinite number of problems. A solution seems hopeless.

But, suppose that new problems have the same form as previously encountered problems, and are simpler. The problem will be solved at least "theoretically," if the rules lead to a solution in a finite number of steps. Such a solution is said to be "recursive."

One of the beauties of a language such as LOGO is that recursive procedure definitions are allowed. And writing a LOGO procedure not only gives a "theoretical" solution, but one which can be carried out by executing the procedure. Of course, for the latter, one needs access to a TI-99/4(A) with TI LOGO (or some other implementation of LOGO).

In thinking through the solution of a problem, one often works "both ends." The big picture leads to smaller pictures. But also details occur which can be incorporated into procedures, which then make the solution of larger problems easier.

Translating into Pig Latin

As a concrete example of these ideas, consider the momentous task of translating an English word into Pig Latin. According to my children, the rule is to add "HAY" at the end of a word beginning with a vowel, otherwise to take the "consonant sound" from the front, add "AY" to it, and put it at the end. Thus "AND" translates to "ANDHAY", and "BREAK" translates to "EAKBRAY".

These rules lead immediately to a LOGO procedure for accomplishing the task:

```
TO TRANWORD W
TEST MEMBER FIRST :W [A E
I O U ]
IFT OUTPUT TRANVWORD :W
IFF OUTPUT TRANCWORD :W
END
```

This procedure reduces our problem to the solution of three simpler problems, which we might need to reduce further. The procedures we need are:

```
MEMBER object list
                ; returns TRUE if object is in
                list and returns FALSE other-
                wise.
TRANVWORD word
                ; translates if word begins
                with vowel
TRANCWORD word
                ; translates if word begins
                with consonant
```

We can hope that MEMBER is a utility built into LOGO. It isn't, but this is no problem. Nearly anything that isn't a primitive can be built in.

At any stage in the solution process we can decide to work on big problems or focus on little ones. The solution of a problem isn't a linear process, even if solutions are usually presented as if the process were orderly and straight-forward. The LOGO procedures document and organize progress.

Let's focus on the problem of deciding membership. If an object is in a list, it is either the first item of the list, or else it is the first object of the rest of the list, or it is not in the list. The definition is naturally recursive:

```
TO MEMBER X SET
IF :SET = [ ] THEN OUTPUT
"FALSE
TEST :X = FIRST :SET
IFT OUTPUT "TRUE
IFF OUTPUT MEMBER :X BF
:SET
END
```

With this definition, MEMBER FIR
:W [A E I O U] will return TRUE
:W begins with a vowel, and FALSE i
doesn't.

The definition of TRANVWORD
so simple we can do it anytime. Let's
it now:

```
TO TRANVWORD W
OUTPUT WORD :W "HAY
END
```

The primitive WORD (undocument
takes two words as input and outp
the word formed by joining them.

The definition of TRANCWO
takes more thinking. We want it to
recursive. We want to move letters fr
the beginning to the end until the f
letter is a vowel, and then add "A
We are led to:

```
TO TRANCWORD :W
TEST MEMBER FIRST :W [A E
I O U ]
IFT OUTPUT WORD :W "AY
IFF OUTPUT TRANCWORD
(WORD BUTFIRST :W FIRST :W)
END
```

If we try (think through, or exec
in LOGO) TRANCWORD "BRE
we find it will return EAKBRAY, as
sired. And TRANCWORD "YOU
turns OUYAY. But TRANCWORD "
runs out of space because the recurs
cannot end. Evidentally Y must
added to the list of vowels. But t
TRANCWORD "YOU would re
YOUHAY and not OUYAY.

Can you fix this bug? We want Y
count as a vowel only if it isn't the f
letter. One solution is to use two inp
to TRANCWORD, one of which is a f
This solution, as well as the gene
zation to translating a sentence, ca
seen by reading the PIGLATIN pr
dure and the procedures it calls.

PROCEDURES

```
TO MEMBER X SET
IF :SET = [ ] THEN OUTPUT "FALSE

TEST :X = FIRST :SET
IFT OUTPUT "TRUE
IFF OUTPUT MEMBER :X BF :SET
END


TO TRANVWORD W
OUTPUT WORD :W "HAY
END


TO PRINTPIG LINE
TEST :LINE = [ ]
IFT PRINTCHAR 13
IFF TYPE TRANWORD F :LINE PRINTC
HAR 32 PRINTPIG BF :LINE
END
```

```
TO TRANCWORD K W
IF :K = 1 THEN MAKE "VOWELS [A E
 I O U ]
IF :K = 0 THEN MAKE "VOWELS [A E
 I O U Y ]
TEST MEMBER FIRST :W :VOWELS
IFT OUTPUT WORD :W "AY
IFF OUTPUT TRANCWORD 0 ( WORD BU
TFIRST :W FIRST :W )
END


TO TRANWORD W
TEST MEMBER FIRST :W [A E I O U ]
IFT OUTPUT TRANVWORD :W
IFF OUTPUT TRANCWORD 1 :W
END
```

```
TO PIGLATIN
CS
PRINTPIG [I WILL HELP ]
PRINTPIG [YOU LEARN PIGLATIN ]
A:
PRINT [ ]
PRINTPIG [TYPE A SENTENCE ]
MAKE "LINE READLINE
IF :LINE = [ ] THEN PRINTPIG [TH
AT WAS FUN ] STOP
PRINTPIG :LINE
GO "A
END


TO HELP
CS
PRINT [FOR PIGLATIN PRACTICE, ]
PRINT [TYPE "PIGLATIN" ]
END
```

LOGO

By Roger B. Kirchner

The Tower of Hanoi

Now we turn to a less frivolous example. The Tower of Hanoi is a puzzle familiar to many. It consists of three pegstands. One contains a "tower" of circular rings. The object is to move the tower from one peg to another, moving one ring at a time, and never moving a larger ring on top of a smaller one. There is rumored to be Buddhist priest working on a puzzle with 64 rings, and the world will end when he finishes. How much should we worry, if he makes one move per second?

We can use LOGO to worry about this problem. We need a procedure, say NUMMOVES which takes for input the number of rings and outputs the number of moves. Suppose we think of the task this way: Move the top n−1 rings to an auxillary peg, then move the largest ring, then move the smaller n −1 rings onto the largest.

This way of viewing the problem leads to the following recursive definition for NUMMOVES:

```
TO NUMMOVES N
TEST :N = 1
IFT OUTPUT 1
IFF OUTPUT 1 + 2*
(NUMMOVES :N − 1)
END
```

"Trying" this procedure, we find that NUMMOVES 2 = 3, and also that NUMMOVES 3 = 7. The reader might try to find a formula for NUMMOVES n, and also the value of NUMMOVES 64.

Of more interest is a procedure for actually solving the puzzle, and more than that, for implementing the solution graphically. By the above reasoning, what we need is a procedure SOLVE with four inputs:

```
SOLVE n peg1 peg2 peg3
        ; move the top n rings from
        peg1 to peg2 using peg3.
```

Using the rules we obtain:

```
TO SOLVE N P1 P2 P3
TEST :N = 1
IFT GETRING :P1 SETRING :P2
IFF SOLVE :N − 1 :P1 :P3 :P2
SOLVE 1 :P1 :P2 :P3
SOLVE :N − 1 :P3 :P2 :P1
END
```

If our goal is to implement the puzzle graphically, we need to implement GETRING and SETRING. The turtle can be used, but it will be more spectacular, more like the colorful toy with plastic rings, if we use tiles. Readers may want to anticipate the solution we will present next issue.

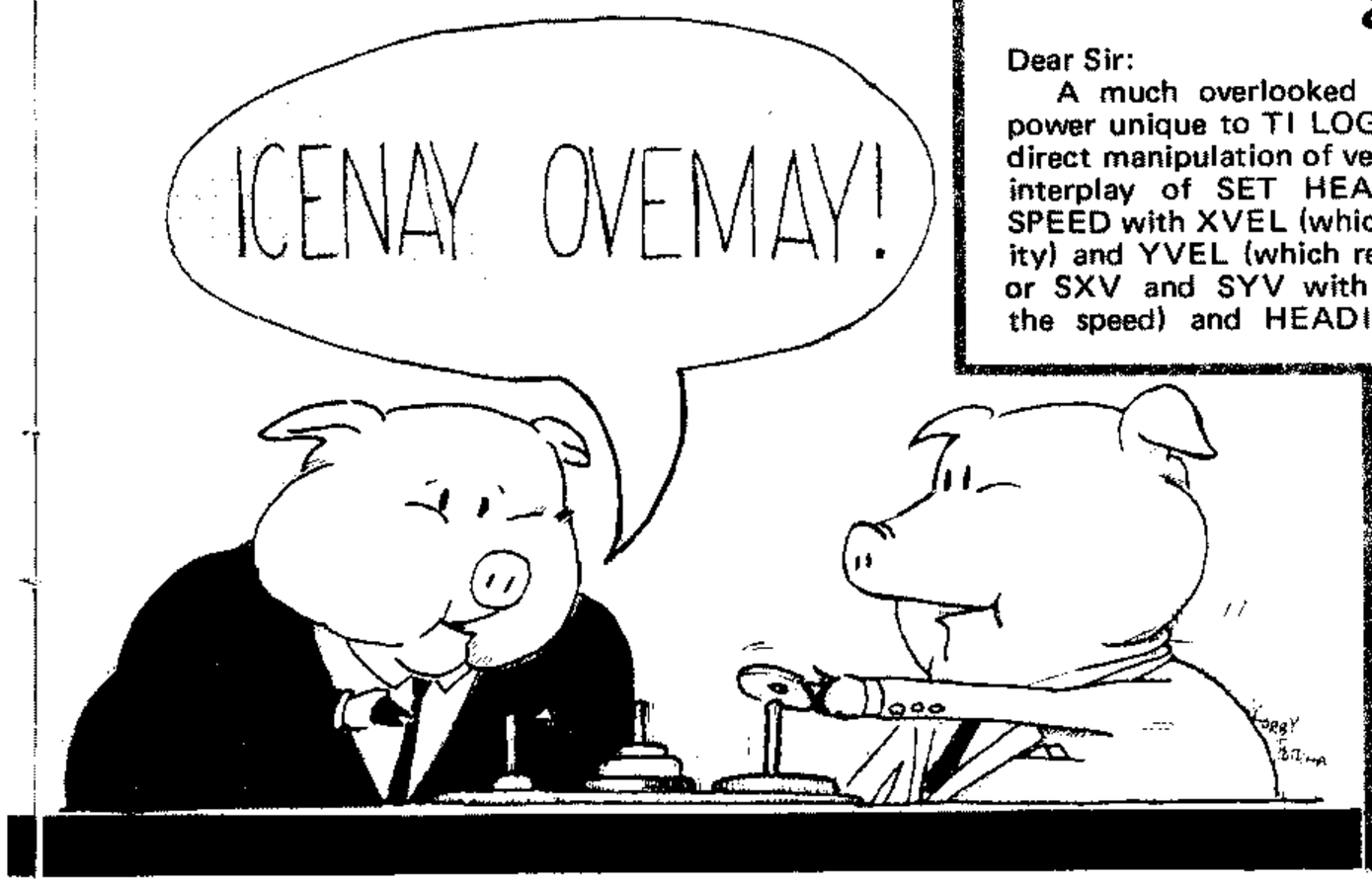
In the meantime, let's implement GETRING and SETRING simply so we can test our solution:

```
TO GETRING P
TYPE [ PICK UP ] PRINTCHAR
32
TYPE :P PRINTCHAR 32
PRINTCHAR 32
END

TO SETRING P
TYPE [SET ON ] PRINTCHAR 32
PRINT :P
END
```

Now, if we enter SOLVE 2 "A "B "C, the output will be:

```
PICK UP A  SET ON C
PICK UP A  SET ON B
PICK UP C  SET ON B
```

The number of moves for three rings is 3, as expected. What will be the seven moves for SOLVE 3 "A "B "C ? Try it! [Watch for Tower of Hanoi II in our next issue—Ed.]



Letters on LOGO

Dear Sir:

A much overlooked and underplayed power unique to TI LOGO is its ability in direct manipulation of vectors—that is, the interplay of SET HEADING and SET SPEED with XVEL (which returns X velocity) and YVEL (which returns Y velocity) or SXV and SYV with SPEED (returns the speed) and HEADING (returns the heading) not only in computation, but also, of course, in that the user *sees* these vector manipulations happen.

I understand from university physics professors that the inability of students to fully understand vectors is one of the primary hindrances on the college level. TI LOGO gives it to a lower primary child.

R. M. Bjes
Pittsburgh, PA

99'er Program Bug

The FLYAWAY procedures as listed in last issue, did not show the colons in front of variables in the procedure names. This was O.K. in the early version of LOGO, but in later releases the colons have to be included. For example, TO SETPLANE P must instead be typed in TO SETPLANE :P and TO CHECK P must likewise be typed in TO CHECK :P etc.

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, NOT THROUGH DISTRIBUTORS, JOBBERS OR OTHER MIDDLEMEN, ASSURING YOU THE BEST SERVICE, EARLIEST SHIPMENT AND LOWEST POSSIBLE PRICES. **(213) 781-6029 (213) 873-4750 SOURCE TI-0166**

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

APPLICATION PROGRAMS

Home Management/Personal Finance — See price shipping offer on soft ware

		Command Modules	
PHM	3006	Home Financial Decisions	22.95
PHM	3007	Household Budget Management (Data storage system is required)	30.95
PHM	3012	Securities Analysis	41.95
PHM	3013	Personal Record Keeping (Data storage system is recommended)	38.95
PHM	3016	Tax Investment Record Keeping (Disk system is required)	53.95
PHM	3022	Personal Real Estate (Data storage system is recommended)	53.95
PHM	3044	Personal Report Generator (Data storage system is recommended)	38.95
		Diskette	
PHD	5001	Mailing List	53.95
PHD	5003	Personal Financial Aids	15.95
PHD	5021	Checkbook Manager	15.95
PHD	5022	Business Aids Library — Finance Management (Extended BASIC Command Module is required)	30.95
PHD	5024	Business Aids Library — Inventory Management (Personal Record Keeping or Statistics Command Module is required)	53.95
PHD	5027	Business Aids Library — Invoice Management (Personal Record Keeping or Statistics Command Module is required)	53.95
PHD	5029	Business Aids Library — Cash Management (Extended BASIC Command Module is required)	30.95
PHD	5038	Business Aids Library — Lease Purchase Decisions	53.95
		Cassette	
PHT	6003	Personal Financial Aids	11.95
PHT	6038	Business Aids Library — Lease Purchase Decisions	46.95

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	3059	Scholastic Spelling — Level III** (Solid State Speech™ Synthesizer is required)	41.95
PHM	3060	Scholastic Spelling — Level IV** (Solid State Speech™ Synthesizer is required)	41.95
PHM	3061	Scholastic Spelling — Level V** (Solid State Speech™ Synthesizer is required)	41.95
PHM	3062	Scholastic Spelling — Level VI** (Solid State Speech™ Synthesizer is required)	41.95
PHM	3064	Typing Tutor**	30.95
		Diskette	
PHD	5007	Teach Yourself BASIC	25.95
PHD	5009	Music Skills Trainer	25.95

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	19.95
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

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	3035	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

SPECIAL

TEX-COMP PRICE

$279.95

Console only

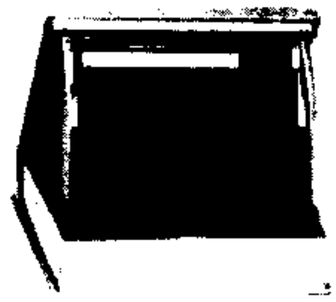
Reg. $49.95 modulator (PHA2100) only $14.95 when Purchased with console.

OTHER APPLICATION PROGRAMS

			Your Cost
		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

*Available in First Quarter 1982
**Available in Second Quarter 1982
†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

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.

Increase your TI Invaders score. More reponsive action.

Available exclusively from TEX-COMP. 34.95 each (specify 99/4 or Atari plug connection).

TERMS: All prices F.O.B. Los Angeles. For fastest service use cashiers check or money order. 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) We will ship UPS C.O.D. with 25% deposit C.O.D. to be paid by cash or certified check. 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

SEND FOR FREE ORDER KIT

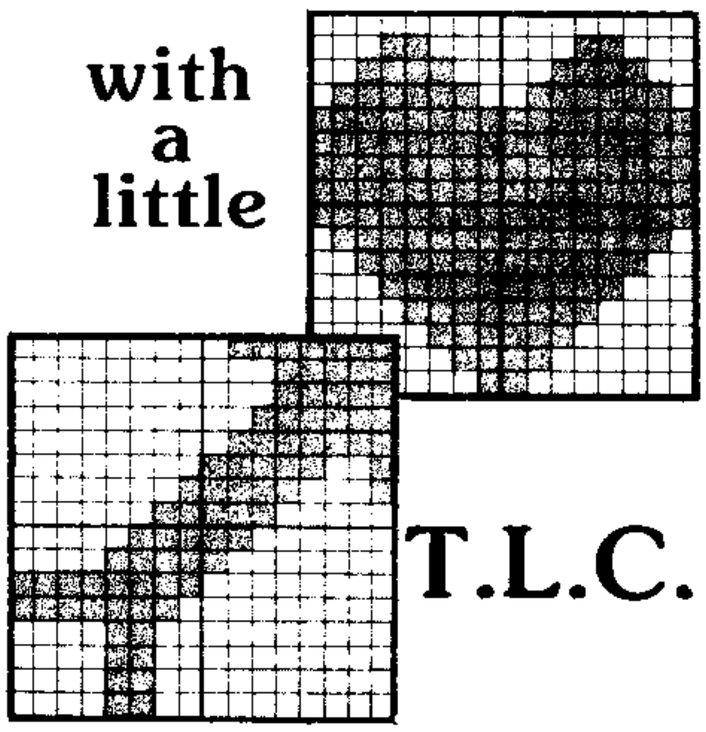
TEX-NET AND THE SOURCE

AMERICA'S INFORMATION UTILITY

Telephone-linked information and reference service

Registration fee...$79.95 Reg. $100.00

TI LOGO with a little T.L.C.



By James H. Muller

Last fall, we had the opportunity to test the LOGO philosophy in a most unique manner. We started a group in Irving, Texas, from absolutely "ground zero" and gave them the opportunity to develop their own computer-based learning environment, and to explore all of the possiblilities available to them. Only one thing, however—I don't think that this is what they were really looking for in the beginning . . .

Late last summer, Donna Bearden, Director of Communications for the Texas Area 5 Health Systems Agency, learned of our activities in Richardson, Texas through a mutual friend, Jack Kishpaugh. Jack is a paraplegic who, using a TI-99/4, developed an award-winning entry in the Johns Hopkins University contest. Donna did much of the documentation for him, and in the process was introduced to programming and home computing. When Jack described the work of the Young Peoples' LOGO Association (YPLA), Donna had to learn more.

Because of Donna's interest and enthusiasm, and because she has three young pre-school and primary grade children, we gave her two computer systems, a bunch of software, and LOGO. She picked them up one rainy Sunday afternoon and took them home. She had everything she needed. But this wasn't what she expected.

What does really happen when a novice gets home with a complete computer system? How do you face the task of assembling console, monitor, disk system, memory expansion, and tape recorder? Do you yell for help? Do you curse the mad character that dumped this "stuff" on you with no explanation or training? Or do you begin to explore the literature, breaking the task down into readily-acceptable tasks, and then assemble them one at a time?

Donna did what I expected her to do—she dug in and got it running. Her very few phone calls proved that any interested person can tackle the home computer, peripherals, software, and LOGO, almost without question. Soon Donna was running a neighborhood group that has since developed into the first YPLA Turtle Learning Center outside of Richardson. As you might sense from the title of this article, those initials were not chosen by accident.

Donna has become an avowed Logophile and an excellent teacher. She has proven to us over and over again that the LOGO philosophy is real—it is practical, and it can and does work if properly implemented.

There are a number of other computer clubs for young people that amount to little more than a few computers set up in a local library or possibly in a scout troop that allow the youngsters to do their own thing. More often than not, an adult supervises these sessions and answers questions. But, left to their own devices, young people tend to flow towards the path of least resistance. They may complete programming exercises, but allowed to do so, most will quickly drop the "work" and play arcade games.

"You can lead a horse to water, but you can't make him drink!" Or alternately, "You can teach a child but you can't 'learn' him." You certainly might have difficulty forcing a horse to drink. And young people will not learn if they so chose *not* to. But if you create the environment in which that horse becomes thirsty, it *will* drink. And if you create the learning environment that excites and challenges the young people, they'll learn in spite of themselves.

This is the world of LOGO, the YPLA, and the Turtle Learning Center program. We're nothing more (or less) than a "dynamic carrot" leading a cart built by and for the YPLA membership. Donna Bearden has brought that idea to life through a group of primary grade youngsters in Irving.

Certainly the youngsters, ages 4 to 11, could be left to fully explore the graphics capabilities of LOGO. But Donna has challenged them to explore—to use their imaginations and creativity in a structured discovery of the computer and LOGO. The results speak for themselves. The procedures submitted by her group are very well thought-out procedures that take good advantage of LOGO capabilities.

At Christmas time, Donna sent us the first disk-based Christmas card we had ever received. Then, in mid-February, we received our first disk-based Valentine's Day card. The shapes involved are a simple heart and an arrow on a diagonal, pointing up from right to left. There is little need to diagram them. To see LOGO in action, as we believe it should be used, we invite you to enter the V-DAY procedure: It shows what can happen when children with but a few weeks of LOGO are challenged to excel. Donna has invited any others using LOGO for preschool and primary grade youngsters to contact her at 1908 Sandy Lane, Irving, Texas 75060.

99'er

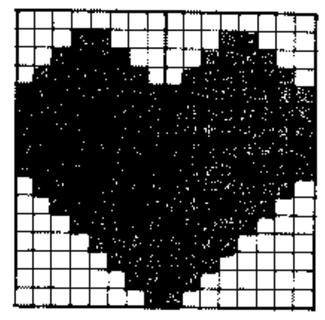
```
TO MESSAGE
CALL "*****************************
*** "L0
CALL "ROSES_ARE_RED "L1
CALL "COMPUTERS_ARE_FUN "L2
CALL "I'D_LIKE_TO_STAY "L3
CALL "BUT_I'VE_A_DATE_WITH_SPRIT
E_1 "L4
CALL "*****************************
*** "L5
PRINT :L0
PRINT :L1
PRINT :L2
PRINT :L3
PRINT :L4
PRINT :L5
END

TO START
TELL :ALL
CARRY 11
SC :RED SS 0
HOME

END
```

```
TO VALENTINE
TELL TILE 32
SETCOLOR [11 4 ]
RISE
PAUSE
ARROW
HIT
MESSAGE
BLINK
END

TO VANISH
TELL :ALL
SC 0 SS 0 SH 0
END
```

```
TO BLINK
TELL :ALL
SC :RED
TELL TILE 32
SC [4 15 ]
WAIT 40
TELL :ALL
SC :WHITE
TELL TILE 32
SC [15 4 ]
WAIT 40
BLINK
END


TO V_DAY
CS
RISE
PAUSE
ARROW
HIT
MESSAGE
BLINK
END
```

```
TO RISE
TELL TILE 32
SETCOLOR [11 4 ]
TELL SPRITE 6
CARRY 11
SC :RED SXY 0 ( - 80 )
SH 0 SS 10
END


TO SHOW
START
HOME
EACH [SH YOURNUMBER * 13 ]
SS 20
WAIT 90
START
END


TO ARROW
TELL 3
CARRY 13 SXY - 84 ( - 64 )
SC :WHITE SH 52 SS 10
END
```

```
TO HIT
TELL 3
TEST YCOR = 0
IFT SHOW
IFF HIT
END

TO PAUSE
TELL SPRITE 6
TEST YCOR = 0
IFT SS 0
IFF PAUSE
END

** DONE **
```



Turtle Traps . . . from p. 59

computer. The first feature is an automatic "garbage-collector." A garbage collector is a part of the operating system which takes used memory and makes that memory available for further uses. Of course, the garbage collector should not destroy and overhaul memory which has not completed all of its work. The way that the automatic garbage-collector in LOGO recognizes when a unit of memory has served its purpose is by checking the instructions written in that memory. Below are examples of programs which permit or exclude the collector:

```
TO POLYGON :SIDES :ANGLE
FORWARD :SIDES
LEFT :ANGLES
POLYGON :SIDES :ANGLES
END
```

This program will never run out of memory in TI LOGO (when all memory has been used up in TI LOGO, the message "out of space" appears) because the garbage collector notes that each time POLYGON is run (referred to as the "level" of POLYGON) there are no further commands or instructions after the line POLYGON :SIDES :ANGLES (called the recursive call line); so the piece of memory that is used to store POLYGON at that level is collected and reused.

```
TO SIDE :LENGTH
FORWARD :LENGTH
END
```

This program will never run out of memory in TI LOGO because the program terminates.

```
TO POLYGON :SIDE :ANGLE
FORWARD :SIDE
LEFT :ANGLE
IF HEADING = 0 STOP
POLYGON :SIDE :ANGLE
PENUP
END
```

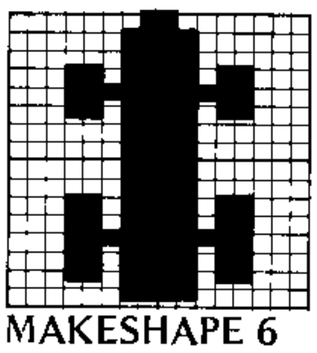
This program could use up all available memory before it reaches its stop conditions because the garage collector cannot refurbish the memory used to execute this POLYGON at any level since there is work left to be done (namely PENUP) once control is passed back to that level of POLYGON.

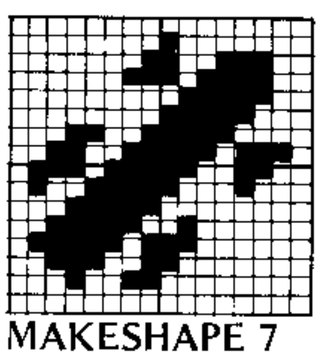
Unfortunately, the garbage collector is not empowered with the authority to decide if any instructions following the recursion call are worth keeping and so

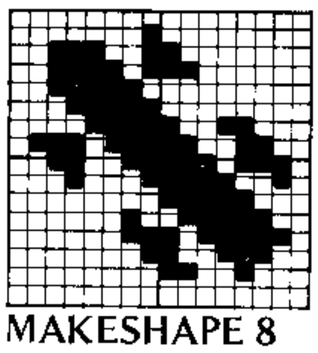
Spinout
By Gene Branum

This program was designed as a cartoon to depict two Indianapolis-style racing cars racing, crashing, burning, and being towed away. The central program, SPINOUT, contains 7 subprograms. These short programs make the central program neat and concise.

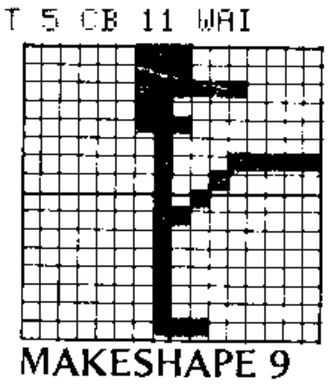
```
TO SPINOUT
WAVE
MOVE
WAIT 350
SWERVE
WAIT 20
SPIN
WAIT 50
BURN
TOW
END

TO WAVE
TELL 3 CARRY 9
SC 1
TELL 4 CARRY 10
SC 1
TELL 3 SX - 75 SY 0
TELL 4 SX - 60
SY 10
SETUP
TELL 4 REPEAT 4 [SY 5 WAIT 10 SY
 10 WAIT 10 SY 15 WAIT 10 SY 10 ]
BEEP
WAIT 30
NOBEEP
END

TO MOVE
TELL 0
SS 12
TELL 1 SS 19
TELL [3 4 ] SC 0
END

TO SWERVE
TELL 1
REPEAT 10 [SX - 5 WAIT 5 SX 0 WA
IT 5 SX - 2 WAIT 5 SX 10 WAIT 6 ]
SX 15
END
```

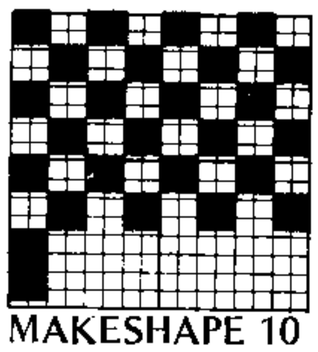

MAKESHAPE 6

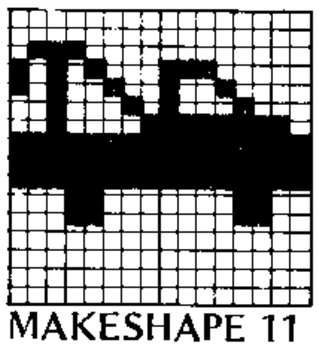

MAKESHAPE 7


MAKESHAPE 8

```
TO SPIN
CALL [0 1 ] "TEAM
TELL :TEAM SS 12
REPEAT 5 [CARRY 6 WAIT 5 CARRY 8
 WAIT 5 CARRY 6 WAIT 5 CARRY 7 W
AIT 5 CARRY 6 WAIT 5 CARRY 8 WAI
T 5 ]
END

TO BURN
REPEAT 10 [CB 6 WAIT 5 CB 11 WAI
T 5 CB 9 WAIT 5 ]
TELL :TEAM
SC 1 SS 0
CB 7
END

TO TOW
TELL 5 CARRY 11
SC 9
SX 100
SY - 70
WAIT 100
SH 270
SS 10
WAIT 115
SS 0 WAIT 205
CALL [0 1 5 ] "T
TELL :T
SH 90
SS 10
WAIT 175 SC 0 SS 0
END

TO SETUP
TELL 0
CARRY 6
SC 6 SS 0 SH 0
HOME
TELL 1 CARRY 6
SC 4
SS 0 SH 0
HOME
TELL 0 SX 15
END
```


MAKESHAPE 9

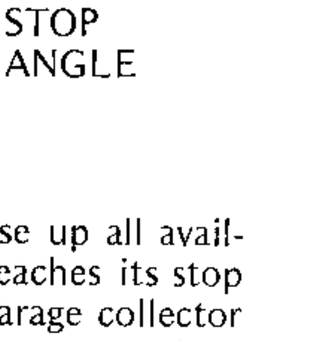

MAKESHAPE 10


MAKESHAPE 11

Space Pylon Racer
By Wesley Adams

Once set up, the player guides his saucer through pylons. Two shapes must be made first (check graph paper). The keys control the saucer. E moves it upward, X moves it downward, D moves it forward, S moves it backwards, F speeds it up, A slows it down. If the ship hits a pylon, the beep sounds.

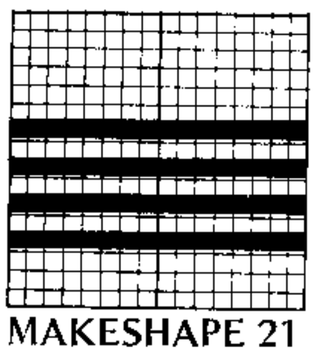
```
TO GAME
TELL :ALL SS 0 SC 0 CARRY 0
SET
TELL 0
SPR
END

TO SPR
CONTROL
CHECK
SPR
END
```


MAKESHAPE 20
Saucer

```
TO SET
TELL 0 CARRY 20 SC :RED SXY - 100 0
TELL [1 2 3 4 5 6 ]
CARRY 21 SC :BLACK
TELL 1 SXY - 50 80
TELL 2 SXY - 50 60
TELL 3 SXY 30 ( - 80 )
TELL 4 SXY 30 ( - 60 )
TELL 5 SXY 100 80
TELL 6 SXY 100 60
END
```

Use arrow keys to change direction.
Use F for fast speed.
Use A for slow speed.

```
TO CONTROL
IF RC? = "TRUE
TEST 3 = 4 ELSE STOP
CALL RC "Z
IF :Z = "E SH 0 STOP
IF :Z = "X SH 180 STOP
IF :Z = "D SH 90 STOP
IF :Z = "S SH 270 STOP
IF :Z = "F SS 10 STOP
IF :Z = "A SS 5 STOP
END
```


MAKESHAPE 21
Pylon

```
TO CHECK
TEST BOTH ( BOTH XCOR < - 35 XCO
R > - 52 ) ( YCOR > 74 )
IFT BEEP WAIT 15 NOBEEP
TEST BOTH ( BOTH XCOR < - 35 XCO
R > - 52 ) ( YCOR < 67 )
IFT BEEP WAIT 15 NOBEEP
TEST BOTH ( BOTH XCOR > 29 XCOR
< 49 ) ( YCOR > - 79 )
IFT BEEP WAIT 15 NOBEEP
TEST BOTH ( BOTH XCOR > 29 XCOR
< 49 ) ( YCOR > - 59 )
IFT BEEP WAIT 15 NOBEEP
TEST BOTH ( BOTH XCOR > 99 XCOR
< 120 ) ( YCOR > 81 )
IFT BEEP WAIT 15 NOBEEP
TEST BOTH ( BOTH XCOR > 99 XCOR
< 120 ) ( YCOR > 59 )
IFT BEEP WAIT 15 NOBEEP
END
```

the following POLYGON program could run out of memory:

```
TO POLYGON :SIDE :ANGLE
FORWARD :SIDE
LEFT :ANGLE
POLYGON :SIDE :ANGLE

END
```

The only difference between the first POLYGON program and that here is the empty line following the recursion call and before END. The garbage collector sees that there is a line of commands (it cannot tell that the line is useless) and so it is barred from refurbishing the memory! Empty lines use up memory and can block garbage collection (depending on their location), so empty lines should be eliminated from your programs.

Finally, the operating system can work faster when fewer sprites are being used—i.e. programs which use no sprites run faster than programs which use sprites. The more sprites in use (generally), the slower the system operates. The reason for the slight degradation in response time is obvious—the system has to check to see which, if any, sprites must be displayed or moved. The way that the system checks on its sprites is to look up the highest number of sprite called upon. For example, TELL 31 or TELL SPRITE 31 would cause the system to check on every sprite from 31 on down through to sprite 0. Such a check is necessary (from the user's perspective) only if all 32 sprites are being used. If only one sprite is needed, then the user should type TELL 0 or TELL SPRITE 0 and the system would skip the checkup on sprites 1 through 31, thus saving a small amount of time.

Elsewhere in this issue of *99'er* there are some programs written by students in a January Term course I taught on LOGO. Their programs show an emerging appreciation for elegance, speed, and simplicity in programming. I have not edited their work (except for correcting their typographical errors), so that there are still more elegant ways of achieving their programs' goals; at the same time, note that they all grasp the essentials of esthetic programming.

99'er

Munchie

By Carlos Valles

Munchie illustrates how sprites can be programmed to move by one's command to certain places where an object may be, then (by testing coordinates within the procedure) eat that object, and continue on until all objects have been eaten. You move Munchie by using arrow keys, and set speed by using keys 0, 5, and 1. You should stop Munchie when passing over the object to be eaten.

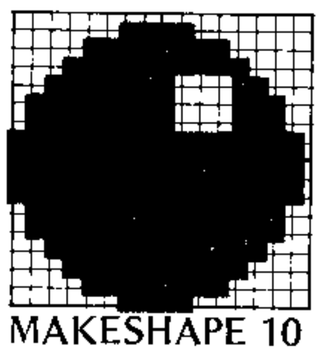
```
TO MUNCHIE
SETUP
MOVE
END
```

```
TO WIPE
CS
TELL :ALL
SC 0
SS 0
SXY 110 95
END
```

```
TO CHECK
TEST BOTH XCOR > -55 XCOR < -45
IFT TEST BOTH YCOR > -5 YCOR < 5
IFT TELL 0 SS 0 EAT1
IFF CARRY 10 BEEP WAIT 5 CARRY 11
NOBEEP
TEST BOTH XCOR > 45 XCOR < 55
IFT TEST BOTH YCOR < 35 YCOR > 25
IFT TELL 0 SS 0 EAT2
IFF CARRY 10 BEEP WAIT 5 CARRY 11
NOBEEP
TEST BOTH XCOR < -10 XCOR > -20
IFT TEST BOTH YCOR > -80 YCOR <
 -70
IFT TELL 0 SS 0 EAT3
IFF MOVE
END
```

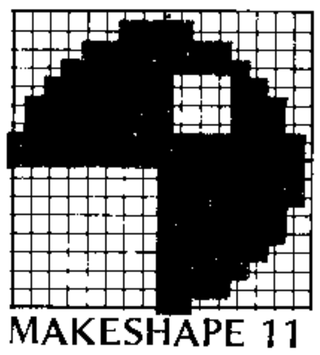
```
TO MOVE
TELL 0 SC :WHITE
CARRY 10 BEEP WAIT 5 CARRY 11 NOBEEP
TEST RC?
IFF CALL "A "M
IFT CALL RC "M
IF :M = "S SH 270
IF :M = "E SH 0
IF :M = "D SH 90
IF :M = "X SH 180
IF :M = "0 SS 0
IF :M = "5 SS 5
IF :M = "1 SS 10
IFF CARRY 10 BEEP CARRY 11 NOBEEP
CHECK
END
```

Press key 1 to 5 to make munchie move.
Press key E, X, S, or D, (arrow keys) to return it.



MAKESHAPE 10

```
TO EAT3
REPEAT 25 [BEEP WAIT 2 NOBEEP
WAIT 2 ]
TELL 3 SC 0
TELL 0 SS 5
MOVE
END

TO EAT2
REPEAT 25 [BEEP WAIT 2 NOBEEP
WAIT 2 ]
TELL 2 SC 0
TELL 0 SS 5
MOVE
END

TO EAT1
REPEAT 25 [BEEP WAIT 2 NOBEEP
WAIT 2 ]
TELL 1 SC 0
TELL 0 SS 5
MOVE
END
```



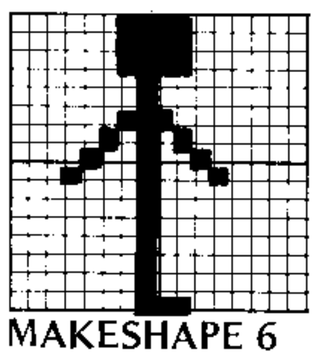
MAKESHAPE 11

```
TO SETUP
TELL 0 SXY -20 40
TELL 1 CARRY :PLANE
SC :BLUE SXY -50 0
TELL 2 CARRY :TRUCK
SC :GREEN SXY 50 30
TELL 3 CARRY :ROCKET
SC :RED
SXY -10 (-70)
END
```

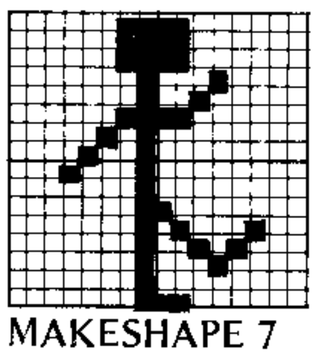
```
TO WIPE
CS
TELL :ALL
SC 0
SS 0
SXY 110 95
END

TO JUMP
TELL 2
CARRY 6
SY 14
WAIT 10
SY 8
WAIT 10
END

TO KICK
SETUP
WAIT 120
TELL 2
CARRY 7
TELL 1
SH 90
SS 10
RISE
TELL 5
CARRY 11
REPEAT 10 [JUMP ]
END
```

Fieldgoal Movie

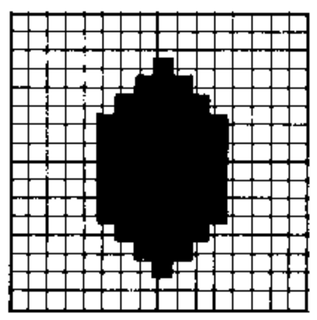
By Rex Baker



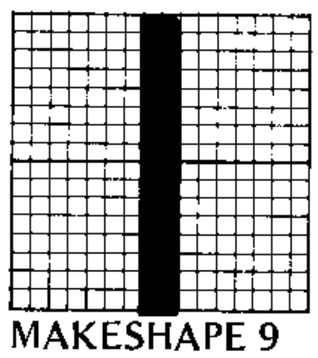
MAKESHAPE 6



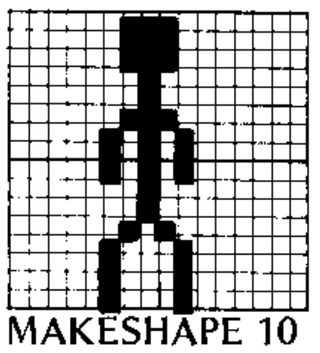
MAKESHAPE 7



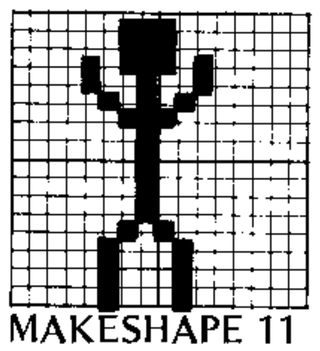
MAKESHAPE 8



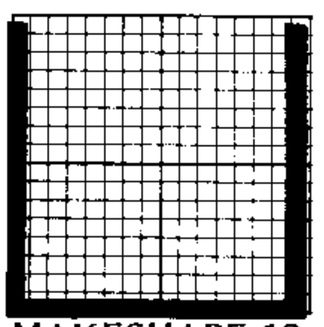
MAKESHAPE 9



MAKESHAPE 10



MAKESHAPE 11

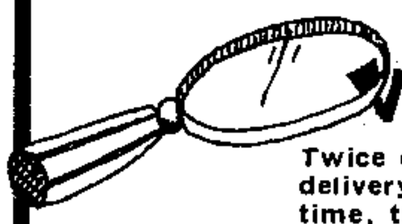


MAKESHAPE 12

```
TO SETUP
WIPE
TELL 1 CARRY 8
SC :YELLOW
HOME
TELL 2 CARRY 6
SC :RED
SXY -5 8
TELL 3 CARRY 9
SC :BLUE
SXY 70 8
TELL 4 CARRY 12
SC :BLUE
SXY 70 23
TELL 5 CARRY 10
SC :BLACK
SXY 80 (-8)
END
```

```
TO RISE
TELL 1
CARRY 8
SH 0
SS 10
SH 45
WAIT 60
SH 90
WAIT 100
SH 135
WAIT 40
SC 0
SS 0
END
```

UPDATE: REGENA



WHO IS REGENA???

Twice each month, like clockwork, a peculiar event occurs. An air freight delivery truck rolls up to our door and drops off a mysterious parcel. Each time, the outside wrapper carries markings from a different country. One thing, however, always remains the same—the two-word return address: "From Regena."

The contents of each package are similar—manuscripts and tapes of articles and programs for and about the TI-99/4 computer, plus a one-line note: "Hope you can use this—Regena." We've never been able to contact this mysterious programmer. Since we presume that he or she gets to see the magazine, it would seem logical that Regena must be a subscriber. But we can't be certain, as large numbers of issues also get sent in bulk to dealers and distributors all over the U. S. and abroad.

Also, there's the matter of the formal letter we received one day from a Zurich bank, with instructions to mail Regena's payment checks to a numbered Swiss account. All very secret and strange indeed . . .

Here at the magazine, we've tried to figure it out: Why would anyone who goes to all this trouble to keep their identity and whereabouts such a secret be writing for a computer magazine? Unless, of course . . . But no, that's the stuff that spy thrillers and James Bond movies are made of . . .

Regena's manuscripts (each typed on a different machine, on paper bearing different watermarks) and program tapes (long strips of reel-to-reel recording tape, rolled up tightly and inserted into 35 mm film cans) have been thoroughly analyzed for clues. It is true, we have found a few, but we suspect, however, that they were deliberate "red herrings." Perhaps you, our readers, have detected some clues in the many articles and programs of Regena that we've published. Or perhaps one of you has actually made contact with our mysterious programmer. In any case, please send us your ideas or information.

And Regena—if you're reading these words—won't you please "come in from the cold . . ."

Background: Before last issue went to press, Regena's usual package arrived—this time postmarked from an obscure group of islands in the South Atlantic, the Falklands. The usual software package contained an additional note, "See you in San Francisco at the West Coast Computer Faire."

Location: Civic Center and Brooks Hall entrance, San Francisco

Date/Time: Saturday, 20 March 1982/10:00 AM

Scene: Thousands of showgoers entering Faire and jamming outer lobby. *99'er Magazine* publisher attempting passage through crowd. Destination: Press Room.

10:07 Arrive Press Room. Spot message to meet at Texas Instruments booth; message signed, "By Regena."

10:13 Arrive TI booth. Huge group congregated within display area; publisher starts circulating.

10:14 Still no contact made when corner video monitor suddenly flashes red color in rapid succession; no one standing nearby.

10:15 Publisher approaches monitor and types in name in responce to on-screen text prompt. New message appears: "LOOK IN YOUR RIGHT JACKET POCKET."

10:16 Publisher finds 35mm film can containing familiar rolled-up strip of recording tape with typed note, "From Regena."

10:17 Publisher starts crowd-search in contact attempt.

10:22 Repeated calling of name "Regena?" through crowd has produced no response up to the time when terminated by magazine subscriber recognizing publisher and questioning him about program in previous issue.

10:26 Other subscribers recognize publisher and engage him in conversations.

10:43 Publisher arrives back at Press Room. New message in place of old: "See you again at 99'er TI-Fest. Circumstances should be O.K. by then for Regena to reveal identity."

Location: Magazine's editorial office—Eugene, Oregon

Date/Time: Monday, 22 March 1982/2:45 PM

Scene: Transfer and loading of new Regena tape results in "San Francisco Tourist" game program for current issue's Gameware Buffet.

Tex-Thello . . . from p. 36

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:

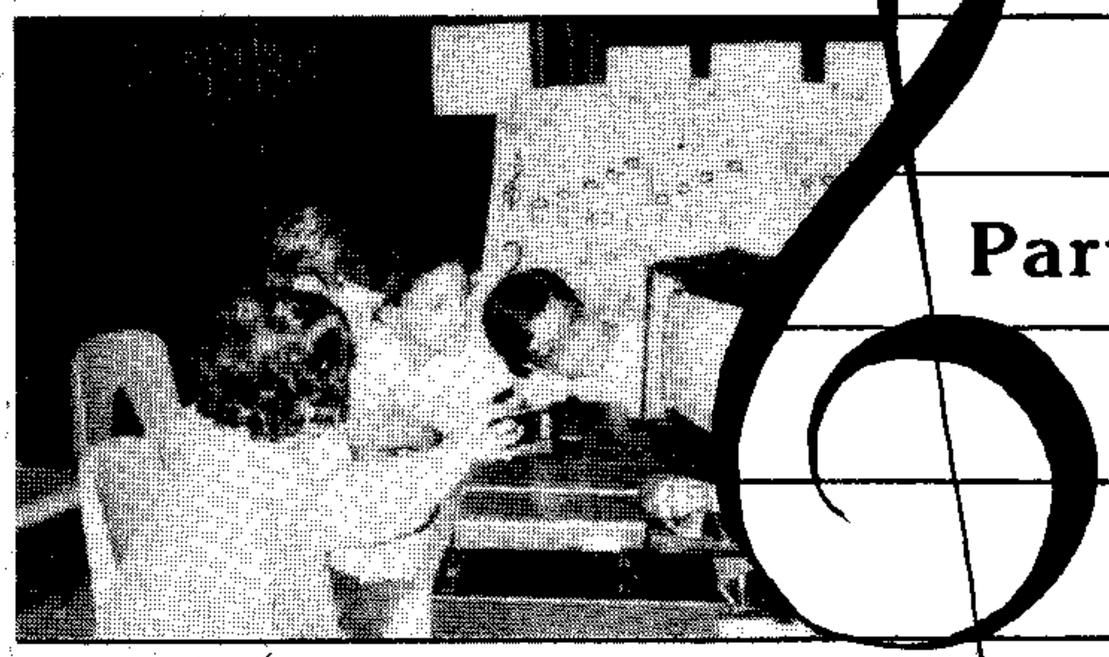
99'er Magazine / Editorial Dept.
2715 Terrace View Drive,
Eugene, Oregon 97405

EXPLANATION OF THE PROGRAM
Tex-Thello

Line Nos.	
160	Dimensions arrays for squares captured.
170-240	Stores the name "COMPUTER" for player.
250-400	Option screens; user presses a key for choices.
410-510	Players input names; stored in PLAY(1,10).
520-610	Initializes positions of board.
620-730	Prints labels for game.
740-920	Defines graphics characters and colors.
930-980	Draws starting Tex-Thello board.
990-1090	Draws starting four positions.
1100-1170	Initializes squares around four center squares; starts for first player on move number 5.
1180-1230	Prints player's name (or computer) and black squares indicating whose move.
1240-1330	Player presses column number then row number for move.
1340-1360	Computer prints move.
1370-1480	Checks for legal move.
1490-1550	Sets values of surrounding squares to zero.
1560-1620	Shows move on screen and switches appropriate captured squares; increments TURN (number of moves).
1630-1740	Checks to see if board still contains two colors, otherwise branches to end of game.
1750-1790	Changes player number for next turn and branches to beginning of main loop.
1800-2040	Tallies squares for each player and prints score.
2050-2100	Asks if player wants to play again; branches appropriately or ends program.
2110-2250	Subroutine to check if there is a legal move.
2260-2510	Subroutine to place colored square on board where player or computer indicates his move.
2520-2820	Subroutine to check how many squares may be captured.
2830-2940	Subroutine to color captured squares.
2950-4240	Subroutine to calculate computer's move. EXTRA is the number of squares that can be captured; HARD is the level of difficulty (1, 2, 3). For the different levels the board positions have different values.

```
100 REM  ***************
110 REM  * TEX-THELLO *
120 REM  ***************
130 REM  BY J. CRAWFORD COOK II
140 REM  REVISED BY C WHITELAW
150 REM  99'ER VERSION 1.5.1
160 DIM EX(18),EY(18)
170 DATA 67,79,77,80,85,84,69,82
180 RESTORE
190 COMPLAY=0
200 CALL CLEAR
210 FOR I=1 TO 8
220 READ PLAY(1,I)
230 PLAY(2,I)=PLAY(1,I)
240 NEXT I
250 PRINT TAB(9);"TEX-THELLO";;;;;
260 PRINT "CHOOSE:";;;"1  ONE PLAYER VS. COMPUTER";
    ;"2  TWO PLAYERS";;;;
270 CALL KEY(0,K,S)
280 IF (K<49)+(K>50)=-1 THEN 270
290 IF K=50 THEN 410
300 CALL CLEAR
310 PRINT "CHOOSE:";;;"1  EASY GAME";
    ;"2  INTERMEDIATE GAME";;"3  HARD GAME";;;;
320 CALL KEY(0,K,S)
330 IF (K<49)+(K>51)=-1 THEN 320
340 HARD=K-48
350 CALL CLEAR
360 PRINT "CHOOSE -- COMPUTER PLAYS";;;
    "1  FIRST--RED";;"2  SECOND--YELLOW";;;
370 CALL KEY(0,K,S)
380 IF (K<49)+(K>50)=-1 THEN 370
390 COMPLAY=K-48
400 IF COMPLAY=1 THEN 470
410 PRINT ;;"FIRST PLAYER NAME (RED)"
420 INPUT Z1$
430 FOR I=1 TO 10
440 CALL GCHAR(23,I+4,PLAY(1,I))
450 NEXT I
460 IF COMPLAY=2 THEN 520
470 PRINT ;;"SECOND PLAYER NAME (YELLOW)"
480 INPUT Z1$
490 FOR I=1 TO 10
500 CALL GCHAR(23,I+4,PLAY(2,I))
510 NEXT I
520 FOR I=1 TO 3
530 DIR(I)=I-2
540 NEXT I
550 FOR I=0 TO 9
560 FOR J=0 TO 9
570 A(I,J)=1
580 IF I*J*(J-9)*(I-9)<>0 THEN 600
590 A(I,J)=A(I,J)+1
600 NEXT J
610 NEXT I
620 CALL CLEAR
630 PRINT TAB(19);"X=   Y="
640 E$="FFFFFFFFFFFFFFFF"
650 CALL CHAR(120,E$)
660 CALL COLOR(12,2,16)
670 FOR I=1 TO 8
680 Y=3+2*I
690 X=Y+4
700 CD=48+I
710 CALL VCHAR(Y,8,CD)
720 CALL VCHAR(4,X,CD)
730 NEXT I
740 A$="FF80808080808080"
750 B$="FF0101010101010101"
760 C$="8080808080808080FF"
770 D$="0101010101010101FF"
780 CALL CHAR(96,A$)
790 CALL CHAR(97,B$)
800 CALL CHAR(98,C$)
810 CALL CHAR(99,D$)
820 CALL CHAR(104,A$)
830 CALL CHAR(105,B$)
840 CALL CHAR(106,C$)
850 CALL CHAR(107,D$)
860 CALL CHAR(112,A$)
870 CALL CHAR(113,B$)
880 CALL CHAR(114,C$)
890 CALL CHAR(115,D$)
900 CALL COLOR(9,2,16)
910 CALL COLOR(10,2,9)
920 CALL COLOR(11,2,11)
930 TYPE=1
940 FOR X=1 TO 8
950 FOR Y=1 TO 8
960 GOSUB 2270
970 NEXT Y
980 NEXT X
990 FOR X=4 TO 5
1000 FOR Y=4 TO 5
1010 IF X=Y THEN 1050
1020 TYPE=2
1030 A(X,Y)=3
1040 GOTO 1070
```

Continued on p. 90



Notes on a Computer Score:

Part 2 - The TI-99/4 Assists Gifted Children in the Learning Process

By Norma Clulow

Although the TI-99/4 proved itself to be a valuable enrichment tool in my traditional music classes (see *99'er Magazine;* Vol. 1, No. 4), I began to realize its full potential during a summer enrichment program for gifted children at New Horizons Academy.[1] It was exciting to be able to allow a curriculum to evolve as children enthusiastically identified their own interests and pursued ways of expressing them creatively through use of the computer.

The Educational Setting

New Horizons Academy is a private school that was founded by Nanci Lucas as an alternative to public school education on the belief that children are naturally excited about learning, and are capable of handling academic pursuits beyond their years. When an individualized curriculum is designed to allow for advancement through basic skills and extensive opportunity for enrichment and acceleration, children find learning exciting and meaningful.

In addition to the regular academic curriculum, the Academy periodically provides workshops that are open to any interested children. Since 1978, "Summer Spectacular" has been offered with courses in computers, creative dramatics, archeology, photography, etc. I became involved with New Horizons last summer when I taught two sessions in "Computer Music" with our TI-99/4.

[1]New Horizons Academy, 1716 Perrysburg -Holland Road, Holland, Ohio 43528

About the Author

Mrs. Clulow has had 12 years experience as an instrumental, vocal, and general music teacher in grades 1-12. She received a B.S. in Music Education from Ohio State University and an M.Ed. in Elementary Education from Bowling Green University.

The Computer Music classes were intended to familiarize students with basic music concepts and provide for individualized and accelerated learning in a manner consistent with the philosophy of the Academy. A typical group consisted of eight students ranging in age from 7-13. The group was scheduled to meet eight times in a two week period with each session lasting one hour. (After the first day, however, the students "demanded" that I arrive at 8:45 A.M. and not leave until 2:15 P.M. Several times it was even 3:45! Seeing your students eagerly asking to cut other classes, skipping lunch, and having their parents pick them up late—all just so they could work on their projects—was tremendously rewarding.

> **"Seeing your students eagerly asking to cut other classes . . . and having their parents pick them up late — all just so they could work on their projects — was tremendously rewarding."**

The Educational Model

I employed Renzulli's theoretical framework, the *Enrichment Triad Model (Renzulli, 1977).* The model contains three constructs which convey the types of learning activity believed to be best for gifted children.

In *General Exploratory Activities* (Type I) students are exposed to a broad range of possibilities. None of these are presented in detail. The purpose is merely to introduce the students to the range of possible alternatives open to them. *Group Training Activities* (Type II) follow, providing the students with fundamental information of potential use in subsequent development of their interest areas. These activities are content oriented.

During the two preceding phases, students begin to identify their interests and develop the skills to create a final product. In *Individual and Small Group Investigations of Real Problems* (Type III) each student determines a problem or project of particular interest that is based on the information obtained in the previous activities, and then pursues that choice in greater depth.

This final element of the Enrichment Triad is perhaps the most important. Ideally, the students will exemplify the "turned-on professional" and pursue their objectives with intense motivation and commitment.

In many respects, Renzulli's model parallels Seymour Papert's principles of continuity, power, and cultural resonance (Papert, 1980).

Implementation of the Model

In the first few days, I exposed the students to a variety of musical activities including a TI-99/4 concert of familiar children's songs such as "Happy Birthday," "Yankee Doodle," and "Pop Goes the Weasel"—all complete with graphics. We also played rhythm instruments, the autoharp, resonator bells and recorders, drew our impressions of music while listening to recordings, identified environmental sounds, and discussed the commonalities and differences of all sounds.

The Texas Instruments *Music Maker* Command Module contains two options by which children can write music. In the exploratory activities, they utilized the Sound Graphs option in which "the composer" need not have any prior knowledge of music notation and theory. In this mode the students experimented with duration of notes by controlling the length of the line in whatever voice they were composing (3 voices or 3-part harmony is possible). Frequency is determined by the height of the line on the screen, and there is a volume choice. From all of our exploratory activities, students came to the conclusion that all sounds have duration, frequency, and volume in common. These concepts were effectively and concretely exemplified by the *Music Maker's* Sound Graphs Mode.

In summary, I believe that the exploratory activitities altered the students' *experience* of music. They began to see music in a new way as part of the continuum of sound and noise; the "freshness" of this new perspective contributed to their desire to move toward the next phases of the model.

Group Training Activities were concerned with content-oriented learning. The objectives were to provide students with a basic knowledge of music theory and an understanding of how a computer program is written—information which they could use as tools in developing their interest areas. Several computer music games and drills were used by the entire group, but as a child's interest waned, he was allowed to break away from the group activity and pursue individual work in his primary interest area.

The computer games included (1) *Mystery Words* in which the players learned the names of treble and bass clef notes; (2) *Rhythm*, which provided ear training in the recognition of quarter notes, eighth notes, and quarter rest patterns; and (3) TI's *Music Skills Trainer*, which contains four games to improve the player's skills in recognition and recall of pitches, intervals, chords, and phrases played by the computer. [See *99'er Magazine;* Vol. 1, No. 4 and Vol. 1, No. 2] We took a look at the program listings for our favorite songs and games and "brainstormed" about what all those commands could possibly mean. Discovering how changing the duration, frequency, and/or volume in a CALL SOUND statement affects the tones produced by the computer was a popular Group Training Activity. The children soon started drawing conclusions and generalizations about how to program. At this point it was necessary to hand out information from the *User's Manual* for students to take home and study over the weekend—homework at *their* request!

"... the exploratory activities altered the students' experience of music. . .almost all students elected to write a computer program to play a musical composition."

Additional content-oriented learning took place when students experimented with the Traditional Mode of the computer's *Music Maker* Command Module. I used the Traditional Mode to help children discover information about key signatures, time signatures, tempo, and music notation—including how various notes such as whole, half, quarter, eighth, and sixteenth and their corresponding rests relate to each other and can be organized into a composition that is musically correct.

In moving into Type III of Renzulli's Model, almost all students elected to write a computer program to play a musical composition; some students selected compositions with which they were already familiar, and others wrote original compositions. Many investigated how to use graphics and color to enhance their creations, and designed a title screen to be displayed during the computer's performance of their work. Compositions included *The Entertainer, Mr. Tambourine Man, Amazing Grace, Beethoven's Ninth (Ode to Joy),* and *Jingle Bells.* Byron, Allan, and Steve exhibited a competitive spirit when comparing the number of lines and difficulty of their programs; Steve wrote his program to play *Beethoven's Ninth* in three-part harmony; Bryan wrote his original composition to flash a change of screen color to emphasize musical contrasts at appropriate points in the music; and Peggy reworked her original composition many times until she was satisfied with the rhythmic structure.

It is important to note that not every student was equally enthusiastic about programming. For example, Adrienne seemed to prefer taking Computer Music for enjoyment and the personal satisfaction of becoming familiar with it, but did not have a genuine interest in becoming a creative producer.

The satisfaction children feel from the opportunity to communicate the results of their work to an appropriate audience was obvious when three ladies from Springfield School District and a banker visited our classroom one day. Byron stopped his work to take over for Mrs. Lucas and me as we were explaining the Computer Music Class. He and two others enthusiastically gave a presentation of their music and proudly explained what had gone into its composition. In addition, the already high level of enthusiasm increased when the class found out they could present their finished products to their parents and others on Visitation Day and possibly have them published in *99'er Magazine.*

With this group I served mainly as a resource person and passed the responsibility for learning and investigating on to the students. Students were introduced to concepts of programming or music theory as they explored and found the need to use this knowledge to make their programs more complex. It was a good example of making the material relevant. The Computer Music course allowed for freedom of choice in that no course requirements were established ahead of time; instead, the class members were allowed to develop their own "courses of study" as their interests developed. Likewise, the time allocations were flexible because the entire staff allowed students to skip their classes and come to Computer Music all day if they wanted.

By only requiring students to play the games described in Type II activities until they no longer were interested, mastery of competencies became more streamlined and exciting—as Renzulli suggests. After playing *Mystery Words*

about ten minutes, seven year old Michael put it this way, "Do we have to play it anymore? We know this now!"

It was interesting to observe how the gifted children mastered the basics much more rapidly and efficiently than my regular general music students. The need for individualization and enrichment for the gifted is obvious when one realizes that playing the same games which intrigued my regular classes for several of their thirty minute music periods, tired the gifted in ten to fifteen minutes; they then requested permission to return to programming their *own* creations. This is an example of Renzulli's differentiation between the need for "real investigative activities" for gifted versus "training exercises"—the phenomenon which prompted the development of his Enrichment Triad Model.

It is important to point out that Renzulli's Model is not a fixed, rigid framework; the three activity types often overlap. For example, while the actual composing of music is Type III (Individual and Small Group Investigations), the trial-and-error initial discoveries as to the computer's musical capabilities might be considered Type I (Exploratory). Likewise, there is overlap in some of the students' other Type II activities. Since brainstorming provides children with the skills needed to explore alternative solutions to problems, our discussions about environmental sounds and computer language were practice exercises in developing the processes that enable a learner to deal more effectively with content, yet this took place during an exploratory activity.

Furthermore, process training (Type II) occured when students wrote their own programs as observed when Byron said, "The computer really programs you; you don't program it." He was referring to the fact that the best programmers learn to think like the computer in that they think out the process of writing their program instead of memorizing what to write. Essentially, they must think about how the computer thinks and determine what they must say and how to say it to get the computer to accomplish their goal. Obviously, the programming required to achieve the final product is a Type III activity, yet the development of thinking processes involved in Type II is also present.

The experience of teaching at New Horizons has given me new insights into how children think and how different their learning styles can be. It was exciting to be able to allow a curriculum to evolve as children enthusiastically identified their own interests and pursued ways of expressing these interests creatively.

Post Script

Although the Academy already had four CBM computers and a TRS-80, the magical attraction of students to our TI-99/4 did not go unrecognized. Soon the Academy purchased a TI-99/4A and subsequently a second one together with a variety of the high quality educational software offered by Texas Instruments.

My husband and I have conducted several other enrichment sessions at the Academy and have become increasingly excited about the profound potential of computer facilitated learning.

99'er

References

Papert, Seymour. *Mindstorms: Children, Computers, and Powerful Ideas.* New York: Basic Books, Inc., 1980.

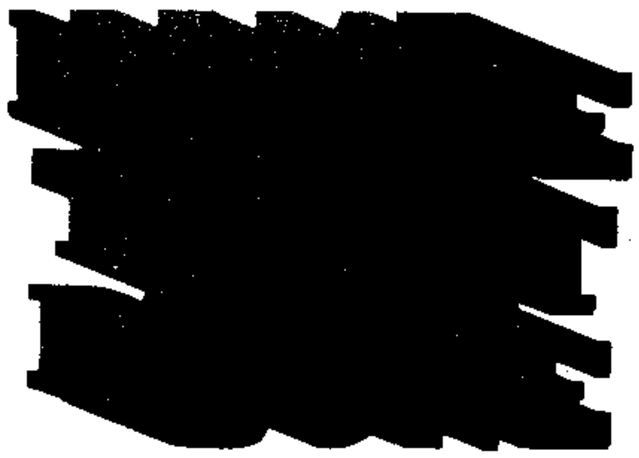
Renzulli, Joseph S. *The Enrichment Triad Model: a guide for developing defensible programs for the gifted and talented.* Connecticut: Creative Learning Press, Inc., 1977.

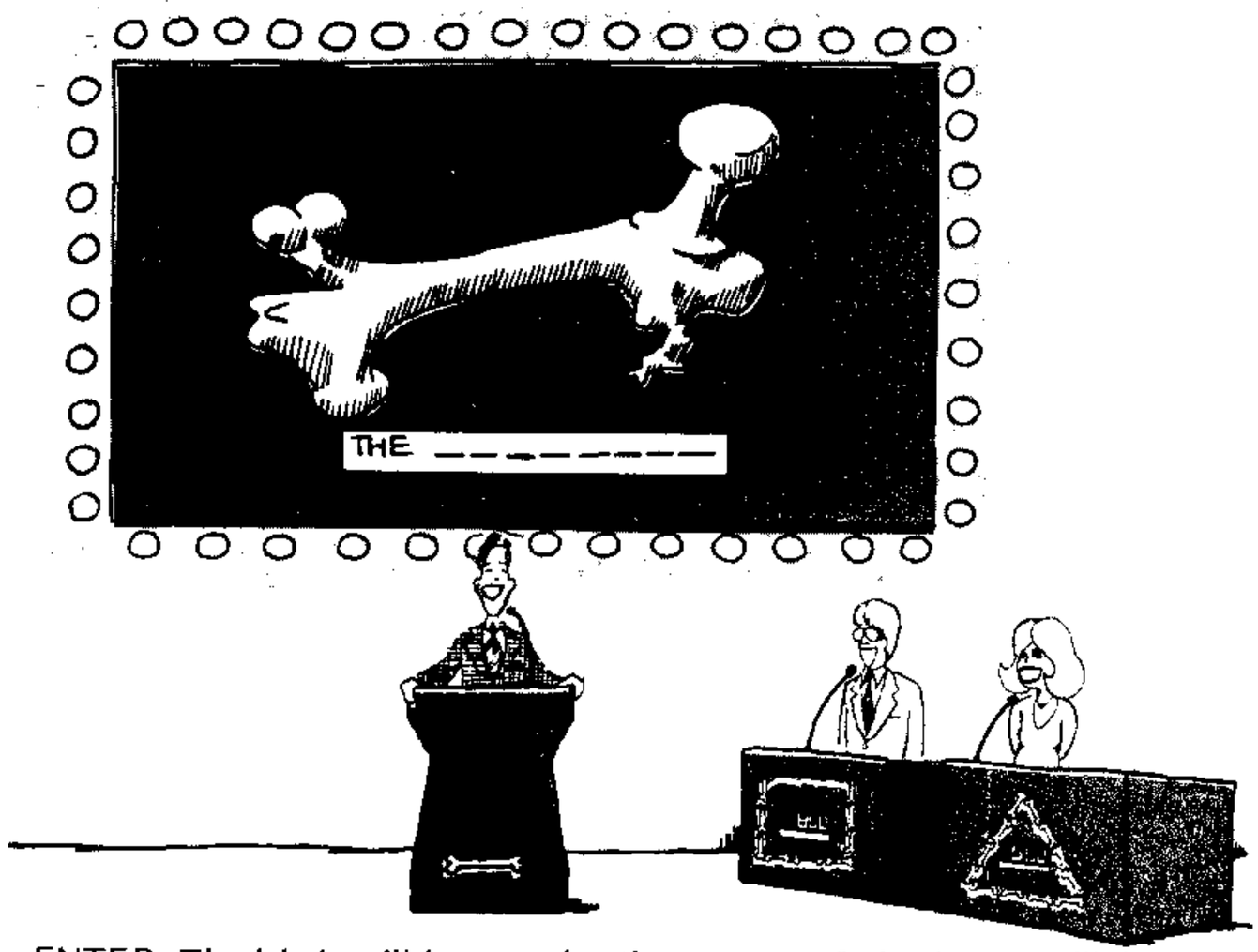
```
30 REM BEETHOVEN'S 9TH
40 REM
50 REM STEVE TUCKER AGE 11
60 REM ROSSFORD, OHIO
70 REM
80 REM 99'ER VERSION 1.5.1XB
90 REM
100 CALL CLEAR
110 CALL SCREEN(3)
120 CALL COLOR(5,2,11,6,2,11,7,2,11,
    8,2,11,1,2,11,4,2,11,2,2,11)
130 DISPLAY AT(11,9):"BEETHOVEN'S"
140 DISPLAY AT(13,8):"9TH SYMPHONY"
150 H=1000
160 Q=500
170 E=235
180 CALL SOUND(Q,659,0,523,2,262,2)
190 CALL SOUND(Q,659,0,523,2,330,2)
200 CALL SOUND(Q,698,0,587,2,440,2)
210 CALL SOUND(Q,784,0,659,2,392,2)
220 CALL SOUND(Q,784,0,659,2,262,2)
230 CALL SOUND(Q,698,0,587,2,220,2)
240 CALL SOUND(Q,659,0,523,2,196,2)
250 CALL SOUND(Q,587,0,494,2,247,2)
260 CALL SOUND(Q,523,0,440,2,330,2)
270 CALL SOUND(Q,523,0,440,2,262,2)
280 CALL SOUND(Q,587,0,494,2,175,2)
290 CALL SOUND(Q,659,0,523,2,220,2)
300 CALL SOUND(750,659,0,523,2,
    220,2)
310 CALL SOUND(E,587,0,494,2,196,2)
320 CALL SOUND(H,587,0,494,2,196,2)
330 CALL SOUND(Q,659,0,523,2,262,2)
340 CALL SOUND(Q,659,0,523,2,330,2)
350 CALL SOUND(Q,698,0,587,2,440,2)
360 CALL SOUND(Q,784,0,659,2,392,2)
370 CALL SOUND(Q,784,0,659,0,262,2)
380 CALL SOUND(Q,698,0,587,2,220,2)
390 CALL SOUND(Q,659,0,523,2,196,2)
400 CALL SOUND(Q,587,0,494,2,247,2)
410 CALL SOUND(Q,523,0,440,2,330,2)
420 CALL SOUND(Q,523,0,440,2,262,2)
430 CALL SOUND(Q,587,0,484,2,175,2)
440 CALL SOUND(Q,659,0,523,2,220,2)
450 CALL SOUND(750,587,0,484,2,
    196,2)
460 CALL SOUND(E,523,0,392,2,131,2)
470 CALL SOUND(H,523,0,392,2,131,2)
480 CALL SOUND(Q,587,0,494,2)
490 CALL SOUND(Q,587,0,494,2)
500 CALL SOUND(Q,659,0,523,2)
510 CALL SOUND(Q,523,0,440,2)
520 CALL SOUND(Q,587,0,494,2)
530 CALL SOUND(E,659,0,523,2)
540 CALL SOUND(E,692,0,587,2)
550 CALL SOUND(Q,659,0,523,2)
560 CALL SOUND(Q,523,0,440,2)
570 CALL SOUND(Q,587,0,494,2)
580 CALL SOUND(E,659,0,523,2)
590 CALL SOUND(E,698,0,587,2)
600 CALL SOUND(Q,659,0,523,2)
610 CALL SOUND(Q,587,0,494,2)
620 CALL SOUND(Q,523,0,440,2)
630 CALL SOUND(Q,587,0,494,2)
640 CALL SOUND(H,392,0)
650 CALL SOUND(Q,659,0,523,2,262,2)
660 CALL SOUND(Q,659,0,523,2,330,2)
670 CALL SOUND(Q,698,0,587,2,440,2)
680 CALL SOUND(Q,784,0,659,2,392,2)
690 CALL SOUND(Q,784,0,659,2,262,2)
700 CALL SOUND(Q,698,0,587,2,220,2)
710 CALL SOUND(Q,659,0,523,2,196,2)
720 CALL SOUND(Q,587,0,494,2,247,2)
730 CALL SOUND(Q,523,0,440,2,330,2)
740 CALL SOUND(Q,523,0,440,2,262,2)
750 CALL SOUND(Q,587,0,494,2,175,2)
760 CALL SOUND(Q,659,0,523,2,220,2)
770 CALL SOUND(750,587,0,523,2,
    196,2)
780 CALL SOUND(E,523,0,392,2,131,2)
790 CALL SOUND(H,523,0,392,2,131,2)
800 FOR I=1 TO 500 :: NEXT I
810 DISPLAY AT(20,5):"PLAY IT
    AGAIN (Y/N)?"
820 CALL KEY(0,KEY,STATUS)
830 IF KEY=78 THEN 870
840 IF KEY<>89 THEN 820
850 CALL HCHAR(20,1,32,32)
860 GO TO 180
870 END
```

99'er

```
30 REM AMAZING GRACE
40 REM
50 REM JEANEAN MITCHELL
60 REM AGE 11 MAUMEE, OH
70 REM
80 REM 99'ER VERSION 1.5.1XB
90 REM
100 CALL CLEAR
110 CALL SCREEN(13)
120 CALL COLOR(5,16,4,6,16,
    4,7,16,4,8,16,4,1,16,4,
    2,16,4,4,16,4)
130 DISPLAY AT(10,8):
    "AMAZING GRACE"
140 DISPLAY AT(19,7):
    "JEANEAN MITCHELL"
150 L=2000
160 M=250
170 H=1000
180 Y=500
190 V=10
200 D=294
210 G=392
220 B=494
230 A=440
240 E=330
250 K=587
260 CALL SOUND(Y,D,V)
270 CALL SOUND(H,G,V)
280 CALL SOUND(M,B,V)
290 CALL SOUND(M,G,V)
300 CALL SOUND(H,B,V)
310 CALL SOUND(Y,A,V)
320 CALL SOUND(H,G,V)
330 CALL SOUND(Y,E,V)
340 CALL SOUND(H,D,V)
350 CALL SOUND(Y,D,V)
360 CALL SOUND(H,G,V)
370 CALL SOUND(M,B,V)
380 CALL SOUND(M,G,V)
390 CALL SOUND(H,B,V)
400 CALL SOUND(Y,A,V)
410 CALL SOUND(L,K,V)
420 CALL SOUND(Y,E,30)
430 CALL SOUND(Y,B,V)
440 CALL SOUND(H,K,V)
450 CALL SOUND(M,K,V)
460 CALL SOUND(M,B,V)
470 CALL SOUND(H,G,V)
480 CALL SOUND(Y,D,V)
490 CALL SOUND(H,E,V)
500 CALL SOUND(M,G,V)
510 CALL SOUND(M,E,V)
520 CALL SOUND(H,D,V)
530 CALL SOUND(Y,D,V)
540 CALL SOUND(H,G,V)
550 CALL SOUND(M,B,V)
560 CALL SOUND(M,G,V)
570 CALL SOUND(H,B,V)
580 CALL SOUND(Y,A,V)
590 CALL SOUND(L,G,V)
600 FOR I=1 TO 500 :
    : NEXT I
610 DISPLAY AT(23,5)
    BEEP:"ENCORE?(Y/N)"
620 CALL KEY(0,KEY,
    STATUS)
630 IF KEY=78 THEN 660
640 IF KEY<>89
    THEN 620
650 GOTO 260
660 END
```

99'er





By Regena

Time to review the Ezekiel's "Dry Bones" song . . . Leg bone connected to the hip bone . . . Or was it the ankle bone? Or *what* bone is *where*?? This program is designed to teach the names of the major bones of the human body and where they are located, and then turn what could be a dry, repetitious drill into an enjoyable game of "Name That Bone."

The menu screen of the program gives the choice of major parts of the body, head, arms, torso, and legs, or end the program. Each section will label the main bones of that part of the body:

1. HEAD — frontal, parietal, zygomatic, temporal, maxilla, mandible
2. ARMS — humerus, ulna, radius, carpus, metacarpus, phlanges
3. TORSO — spine, ribs, clavicle, scapula, sternum, ilium, ischium, sacrum, coccyx
4. LEGS — femur, tibia, fibula, patella, tarsus, metatarsus, phlanges

You may study the labeled diagram of the bones as long as you wish, then press ENTER. The labels will be erased and it will be your turn to "Name That Bone." The bones are listed in a random order at the left of the screen for your choice of answers. A bone will be chosen randomly and will blink red and white until you press a number corresponding to the name of the bone. If you are correct, an arpeggio is played; if you are incorrect, a noise is sounded. You must press the correct answer to continue, and it won't take long for you to learn the names of your bones.

After each bone is chosen once, you will be asked TRY AGAIN? (Y/N). If the response is N, the program returns to the menu screen. If the response is Y, the names of the bones will be rearranged and the bones will be chosen in a different order.

Programming Techniques

There are four main parts of the body form which to choose, and each part uses the same program logic, so subroutines are used. The subroutines are located at the beginning of the program. For some microcomputers, execution is faster for subroutines called closer to the beginning; however, the speed in TI BASIC does not seem to depend upon the location of the subroutine.

EXPLANATION OF THE PROGRAM
Name That Bone

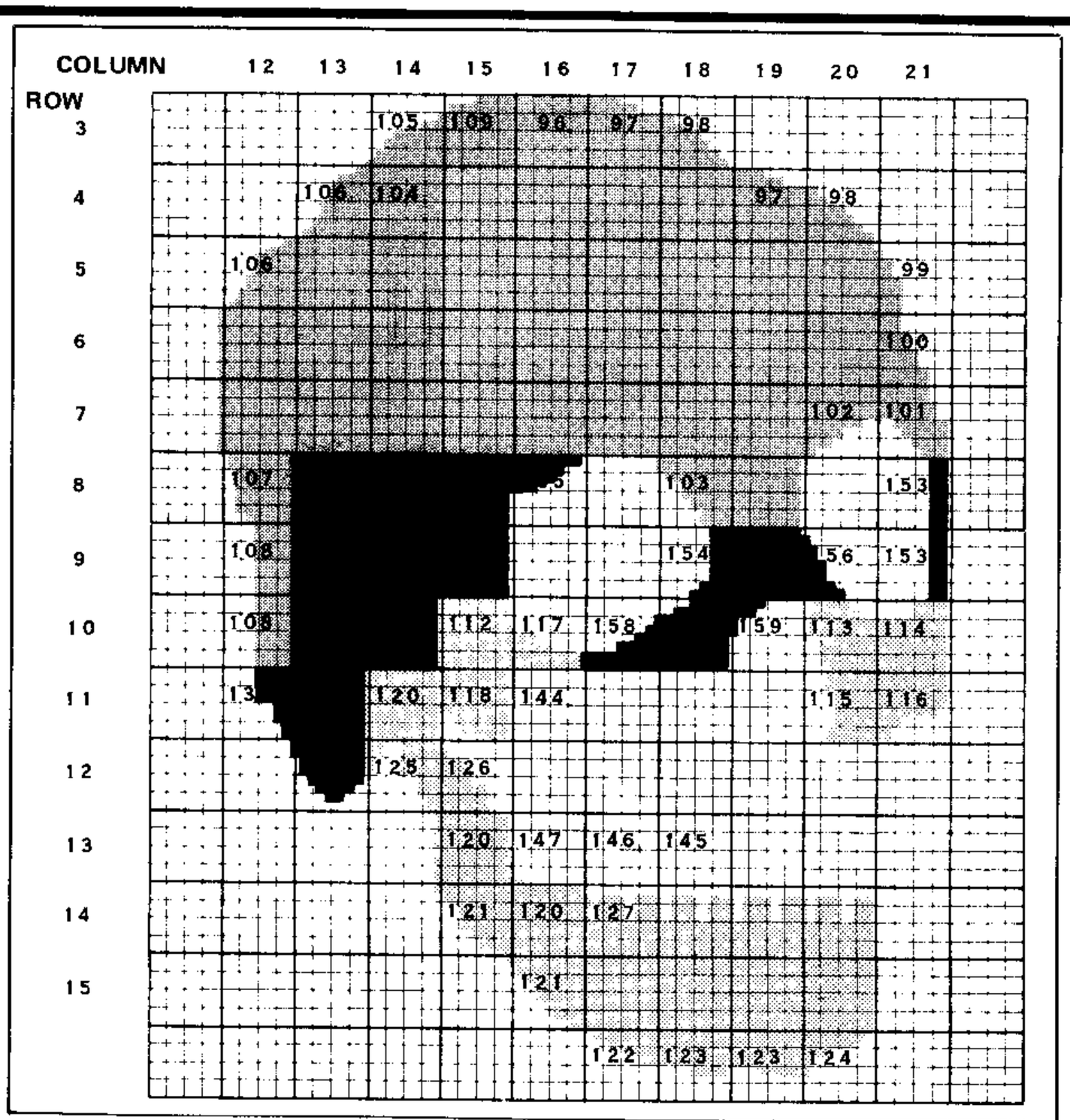
Line Nos.	
150	Branches to title screen.
160-210	Subroutine reads C and C$ from DATA to define graphics characters.
220-310	Subroutine prints PRESS ENTER and waits for the user to respond.
320-360	Subroutine reads DATA to draw graphics.
370-980	Subroutine for main program logic.
370-390	For the number of bones R, reads the name of the bone and the corresponding character set number.
400-520	Randomly prints the names of the bones for the multiple-choice answers and arranges the corresponding character set number and answer number.
530-580	Prints NAME THAT BONE at the top of the screen.
590-660	Randomly chooses a bone and blinks it red and white while waiting for the user to press the answer.
670-780	If the answer is correct, plays an arpeggio and goes to the next bone; if the answer is incorrect, sounds a noise and awaits another key press.
790-980	Prints TRY AGAIN? (Y/N) and branches appropriately after Y or N is pressed.
990-1100	Defines graphics characters for head.
1110-1120	Labels head bones.
1130-1260	Draws skull and waits for user to press ENTER.
1270-1300	Clears labels.
1310-1350	Main procedure for head.
1360-1440	Defines characters for arm.
1450-1500	Labels arm bones.
1510-1580	Draws arm bones and waits for user to press enter.
1590-1620	Clears labels.
1630-1670	Main procedure for arm.
1680-1820	Defines characters and colors for torso.
1830-1850	Labels torso bones.
1860-2020	Draws torso bones and waits for user to press ENTER.
2030-2080	Clears labels.
2090-2160	Main procedure for torso.
2170-2240	Defines characters for leg.
2250-2260	Labels leg bones.
2270-2370	Draws leg bones and waits for user to press ENTER.
2380-2410	Clears labels.
2420-2460	Main procedure for leg.
2480-2600	Prints title screen and draws stick figure.
2610-2700	First time through the program defines the first character in each character set as a solid block then asks if instructions are desired.
2710-2790	Prints instructions and waits for user to press ENTER.
2800-2890	Prints choices of head, arms, torso, legs, or end program.
2900-2990	Waits for user's choice and branches appropriately.

For each part of the body, different characters are defined. The appropriate DATA statement is RESTOREd, then the subroutine to define characters (Lines 160-210) is called. The labels for the bones are printed then the bones are drawn, again RESTOREing the corresponding DATA statement and calling a subroutine (320-360).

The main procedure is in Lines 370-980. The program will read from DATA the names of the bones and the character set number, then randomly prints the bones and chooses the bones for the quiz.

The graphics characters were designed so that a specific bone could be blinked by using CALL COLOR statements. The characters of one bone must be in one character set, and another bone in another character set. When the main part of the body is first drawn, all the characters are yellow, but as the bone is chosen, the characters in that set will blink. An example is shown with the skull bones.

NOTE: The wrist and hand bones may be known as CARPUS and METACARPUS or CARPALS and METACARPALS. The carpals are the elements of the carpus (wrist bone). You may wish to relabel these parts to be consistent with how you want them learned.




```
100 REM  ********************
110 REM  * NAME THAT BONE *
120 REM  ********************
130 REM  BY REGENA
140 REM   99'ER VERSION 1.5.1
150 GOTO 2480
160 FOR I=1 TO N
170 READ C,C$
180 CALL CHAR(C,C$)
190 NEXT I
200 CALL CLEAR
210 RETURN
220 DATA 80,82,69,83,83,32,60,69,78,84,69,82,62,32
230 RESTORE 220
240 FOR Y=19 TO 32
250 READ G
260 CALL HCHAR(24,Y,G)
270 NEXT Y
280 CALL KEY(0,K,S)
290 IF K<>13 THEN 280
300 CALL HCHAR(24,19,32,13)
310 RETURN
320 FOR I=1 TO N
330 READ X,Y,G,R
340 CALL HCHAR(X,Y,G,R)
350 NEXT I
360 RETURN
370 FOR I=1 TO R
380 READ BONE$(I),B(I)
390 NEXT I
400 RANDOMIZE
410 FOR I=1 TO R
420 RR=INT(RND*R+1)
430 IF BONE$(RR)="" THEN 420
440 B$(RR)=BONE$(RR)
450 BB(RR)=B(RR)
460 ANS(RR)=I
470 CALL HCHAR(15+I,2,48+I)
480 FOR J=1 TO LEN(B$(RR))
490 CALL HCHAR(15+I,J+3,ASC(SEG$(B$(RR),J,1)))
500 NEXT J
510 BONE$(RR)=""
520 NEXT I
530 DATA 78,65,77,69,32,84,72,65,84,32,66,79,78,69,32
540 RESTORE 530
550 FOR Y=9 TO 23
560 READ G
570 CALL HCHAR(1,Y,G)
580 NEXT Y
590 FOR I=1 TO R
600 RR=INT(RND*R+1)
610 IF B$(RR)="" THEN 600
620 CALL HCHAR(14,2,63,3)
630 CALL KEY(0,K,S)
640 CALL COLOR(BB(RR),16,1)
650 CALL COLOR(BB(RR),7,1)
660 IF S<1 THEN 630
670 IF K-48=ANS(RR)THEN 700
680 CALL SOUND(500,-5,1)
690 GOTO 630
700 CALL HCHAR(14,2,32,3)
710 CALL SOUND(150,262,1)
720 CALL SOUND(150,330,1)
730 CALL SOUND(150,392,1)
740 CALL SOUND(150,330,1)
750 CALL SOUND(200,262,1)
760 CALL COLOR(BB(RR),12,1)
770 B$(RR)=""
780 NEXT I
790 DATA 84,82,89,32,65,71,65,73,78,63,32
800 RESTORE 790
810 FOR Y=22 TO 32
820 READ G
830 CALL HCHAR(23,Y,G)
840 NEXT Y
850 CALL HCHAR(24,26,40)
860 CALL HCHAR(24,27,89)
870 CALL HCHAR(24,28,47)
880 CALL HCHAR(24,29,78)
890 CALL HCHAR(24,30,41)
900 CALL KEY(0,K,S)
910 IF K=78 THEN 2480
920 IF K<>89 THEN 900
930 FOR Y=16 TO 24
940 CALL HCHAR(Y,2,32,12)
950 NEXT Y
960 CALL HCHAR(23,22,32,10)
970 CALL HCHAR(24,26,32,5)
980 RETURN
990 RESTORE 1000
1000 DATA 97,FOFCFFFFFFFFFFFF,98,0000C0E0F8FCFEFF,99,00B0C0C0E0E0E0E,100,E0E0F0F0F8F8FCFC
1010 DATA 101,FCFCFEFE3F1F0F07,102,FFFFFFFFFEF8F0C,103,FFFF7F1F1F0F0707,109,0F7FFFFFFFFFFFF
1020 DATA 105,000000031F3F7FFF,106,01070F0F1F1F3F7F,107,7F7F3F3F3F3F3F1F,108,0F0F0F0F0F0F0F0F
1030 DATA 137,0F0F0F0F03030101,138,FFFF7F7F3F1E0C,155,FFFEFCF,153,0303030303030303,159,F0C0B08
1040 DATA 154,0303030303030F0F,156,00B0C0C0E0E0F0F8,157,0F3FFFFFFFFFFFFE,158,000000001030FFFF
1050 DATA 115,7C7E7F7F3F3F1F1F,114,0F0FFFFFFFFFFFFF,115,1F1F0F0F0F0F1F1F,116,FFFFFFFFFEE0C0C
1060 DATA 117,000000B0E0FCFFFF,118,FFFFFFFFFF3F1F07,121,FF7F3F1F1F0F0701,122,7F1F0701
1070 DATA 123,FFFFFFFF0F,124,FEFCF0C08,125,CFCF878703030101,126,00B0E0F0F0F8FBFB
1080 DATA 127,00BBBBBBFFFFFFFF,145,FFFFFFFF77777777,146,FFFF7B7B7B7B,147,FFFFFFF FFFDB1BC
1090 N=37
1100 GOSUB 160
1110 PRINT " PARIETAL"::TAB(20);"FRONTAL";::" TEMPORAL"
1120 PRINT TAB(20);"ZYGOMATIC"::TAB(19);"MAXILLA":::"       MANDIBLE"::::::::
1130 RESTORE 1140
1140 DATA 3,14,105,1,3,15,109,1,3,16,96,1,3,17,97,1,3,18,98,1,4,13,106,1,4,14,104,1
1150 DATA 4,15,96,4,4,19,97,1,4,20,98,1,5,12,106,1,5,13,104,2,5,15,96,6,5,21,99,1
1160 DATA 6,12,104,3,6,15,96,6,6,21,100,1,7,12,104,3,7,15,96,5,7,20,102,1,7,21,101,1
1170 DATA 8,12,107,1,8,13,136,3,8,16,155,1,8,18,103,1,8,19,96,1,8,21,153,1,9,12,108,1
1180 DATA 9,13,136,3,9,18,154,1,9,19,152,1,9,20,156,1,9,21,153,1,10,12,108,1,10,13,136,2
1190 DATA 10,15,112,1,10,16,117,1,10,17,158,1,10,18,157,1,10,19,159,1,10,20,113,1,10,21,114,1
1200 DATA 11,12,137,1,11,13,136,1,11,14,120,1,11,15,118,1,11,16,144,3,11,20,115,1,11,21,116,1
1210 DATA 12,13,138,1,12,14,125,1,12,15,126,1,12,16,144,5,13,15,120,1,13,16,147,1,13,17,146,1
1220 DATA 13,18,145,3,14,15,121,1,14,16,120,1,14,17,127,4,15,16,121,1,15,17,120,4
1230 DATA 16,17,122,1,16,18,123,2,16,20,124,1,16,21,32,1
1240 N=66
1250 GOSUB 320
1260 GOSUB 230
1270 RESTORE 1280
1280 DATA 4,4,32,8,6,22,32,7,9,4,32,8,10,22,32,9,13,21,32,7,16,9,32,8
1290 N=6
1300 GOSUB 320
1310 R=6
1320 DATA FRONTAL,9,PARIETAL,10,ZYGOMATIC,11,MANDIBLE,12,TEMPORAL,14,MAXILLA,15
1330 RESTORE 1320
1340 GOSUB 370
1350 GOTO 1330
1360 RESTORE 1370
```

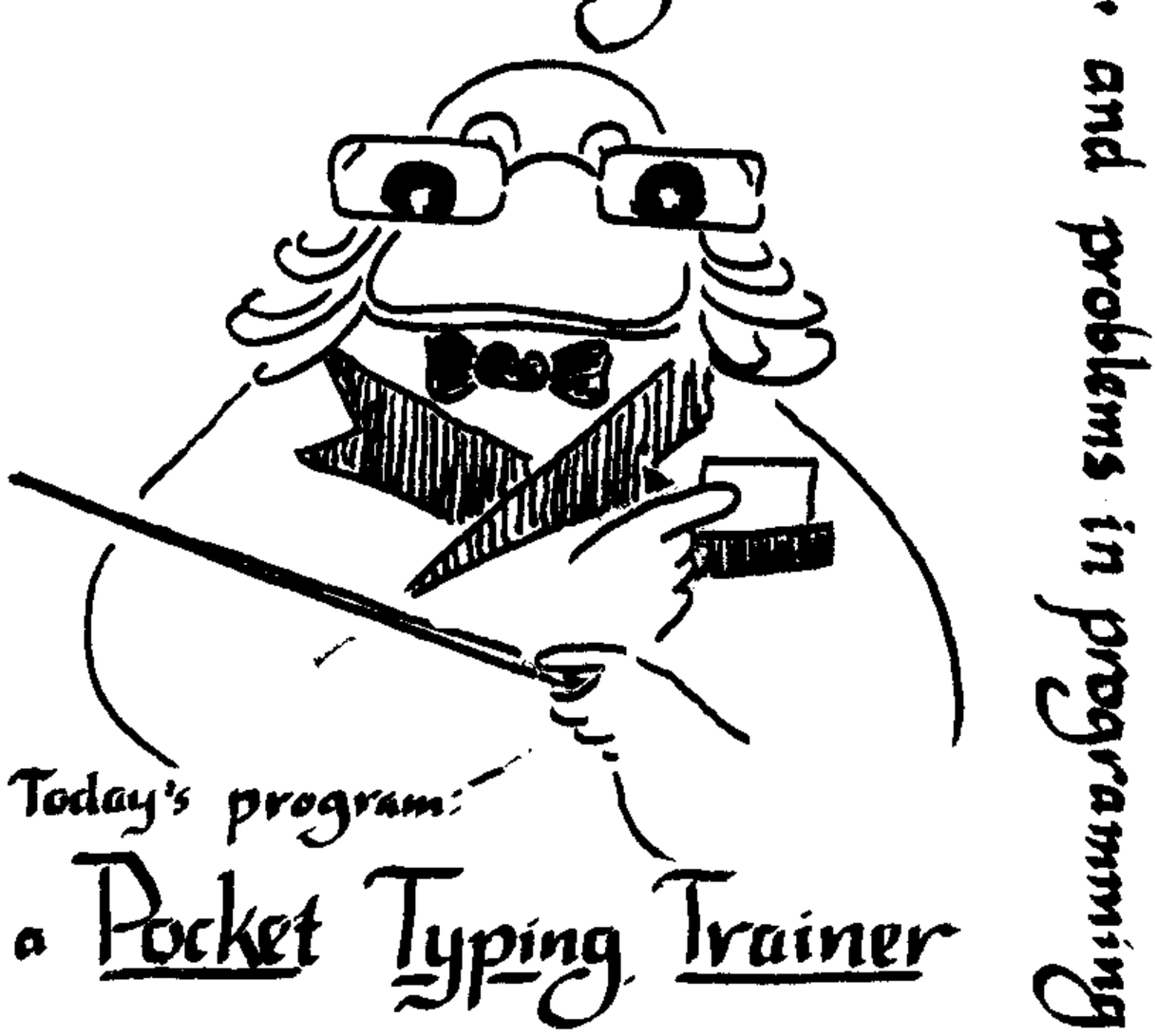
Continued on p. 78

Professor Holl's Pocket Programs . . .

By S.T. Holl

Qtrs 327A, Yerba Buena Island
San Francisco, CA 94130

```
100 REM   **POCKET TYPING TRAINER**
110 REM   99'ER VERSION 1.4.1
120 REM   BY S.T. HOLL
130 DISPLAY "TYPE IN THE LETTERS YOU
    WANT TO PRACTICE TODAY"
140 INPUT LETTER$
150 LENGTH=LEN(LETTER$)
160 OUT$=NUL$
170 FOR I=1 TO 5
180 OUT$=OUT$&SEG$(LETTER$,INT
    (LENGTH*RND^3+1),1)
190 NEXT I
200 DISPLAY "       ";OUT$
210 INPUT "       ":IN$
220 IF OUT$=IN$ THEN 320
230 FOR I=1 TO 5
240 L$=SEG$(OUT$,I,1)
250 IF L$=SEG$(IN$,I,1)THEN 280
260 N=POS(LETTER$,L$,1)
270 LETTER$=L$&SEG$(LETTER$,1,N-1)&SEG$
    (LETTER$,N+1,LENGTH-N)
280 NEXT I
290 CALL SOUND(100,131,3)
300 CALL SOUND(100,110,3)
310 GOTO 160
320 CALL SOUND(100,110,3)
330 CALL SOUND(100,262,2)
340 GOTO 160
```



Here is a pocket-sized program for the TI-99/4A—small enough to fit on a 3x5 card—that is not only quick to key in, but is also educational, illustrates a powerful technique with random numbers, and is fun for all ages. The *Pocket Typing Trainer* asks which characters the user would like to practice, and then plays back an endless series of random 5 character groups for him to copy. Two tones rising at the end of the typist's response says "correct"; two tones descending here mean "oops." Try it ! If you are a beginning typist, start with characters ASDF, the home keys of the left hand. Stop the program with a FCTN 4 (or SHIFT C on the 99/4) keystroke when you can type those four consistently without looking at them, and RUN the program again with ASDFJKL, and so forth . . . If you are already a typist and you want to practice some of the unusual features of the TI-99/4A keyboard, as well as some of the characters important in BASIC (but not usually part of the typist's repertoire), try the characters "$( )*+–.

You are unlikely to notice it, but the *Pocket Typing Trainer* tends to focus on the characters which the typist is getting wrong—a remarkably sophisticated feature to find in a pocket-sized program—and one which brings me to my next point.

Skewing the Distribution

Line 180 is where OUT$, the random character string, is manufactured a character at a time. It *might* have been written without the ^3, in which case equal segments of the interval from zero to one would be assigned to the characters given by the typist (Since the 99/4's built in random number generator, RND, generates "uniform random" numbers, every character would have the same chance of being chosen.) *With* ^3, the random numbers are cubed before a character is chosen. Since the numbers are less than 1, they get smaller as they are cubed; this results in many more RND's corresponding to characters at the left end of the letter$ string. For example, suppose that LETTER$, the string of characters which the typist wants to practice, has four characters. If RND turns out to be .50001 then the character a bit more than half way down LETTER$ (i.e., the third character) would be the one chosen. But if we cube RND, the result is .12500 which is well within the first quarter of the range from 0 to 1; and the first character is chosen. Perhaps diagrams 1 and 2 would help to illustrate this more clearly. The *Pocket Typing Trainer* takes advantage of this by moving missed letters to the beginning of LETTER$ (line 270).

The lesson here is that uniform random numbers like those provided by RND are a perfectly satisfactory foundation for any sort of randomness one could desire. This includes the statisticians' favorites: Gaussian, binomial, gamma, and so on. One simply needs to apply the proper transformations.

Homework

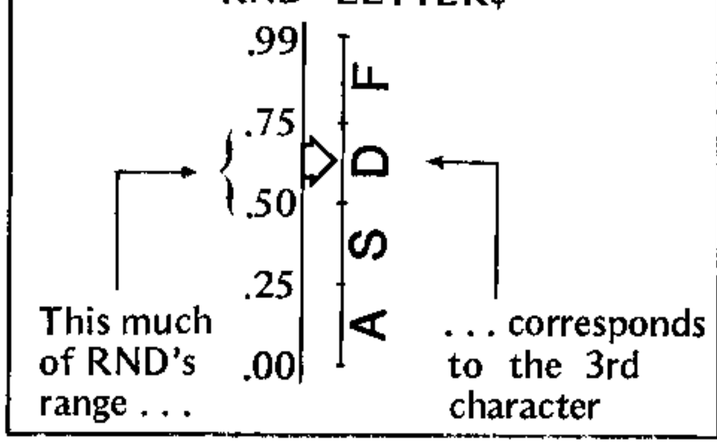
Tailoring and embellishing programs to suit users' personalities is at least half the fun of computing. The *Pocket Typing Trainer* can be extended in many directions. Here are some of the options:

Problem # 1 (simple): Modify the program to allow the typist to choose how many random characters he'd like on a line.

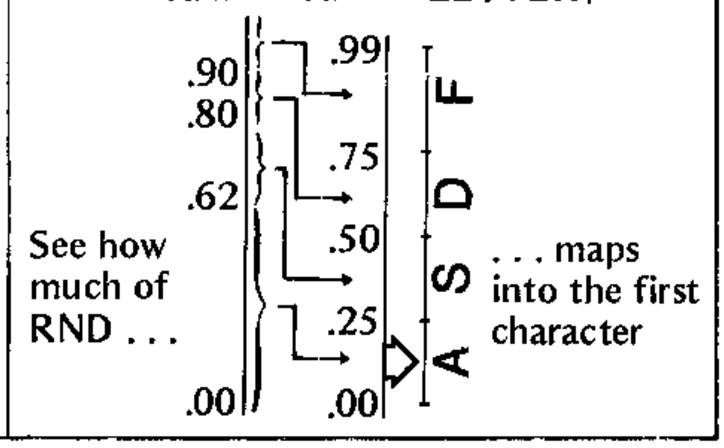
Problem # 2 (moderate): Change the program to heighten the emphasis on characters which the typist is getting wrong whenever his or her error rate is high.

Problem # 3 (sound and graphics practice): Keep score, and periodically (say every 25 lines) treat the typist to a colorful and melodic display, one whose elaborateness is greatest for a perfect score.

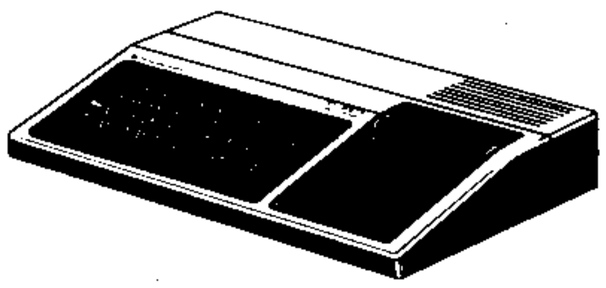
without ^3: Diagram 1.



with ^3: Diagram 2.



TEXAS INSTRUMENTS



TI-99/4A $319.00

includes console, cassette cable, and RF Modulator

Peripheral Box	199.97
Speech Synthes	109.97
Telephone Modem	164.97
RS232 Interface	164.97
Disk Drive Cont.	229.97
Disk Memory Drive	369.97
32K Expansion	287.97
Solid State Printer	299.97
10" Monitor	369.97
Joy Sticks (pair)	27.49

Extended Basic	75.00
Household Mgmt.	31.49
Personal Rec. Keep	38.97
Early Lrn. Fun	23.49
Add Sub I or II	31.49
Video Chess	54.97
Video Games I	23.49
Car Wars	31.49
TI Invaders	31.49
Tombstone City	31.49
Munch Man	31.49
Yahtzee	19.49
Blasto	19.49
Adventure (Cass)	38.97
Editor Assembler	77.97
Terminal Em. II	38.97

IN PENNSYLVANIA
1 - 412 - 321-5444

1-800-441-7410 TOLL FREE

NORTH HILLS COMPUTER

— a subsidiary of DigiCom Systems Corporation —

1016 Madison Avenue, Pittsburgh, PA 15212

MAIL TO:
NORTH HILLS COMPUTER
1016 Madison Avenue
Pittsburgh, PA 15212

NAME

ADDRESS

CITY STATE ZIP

☐ Check ☐ Mastercard ☐ Visa

☐ Money Order, Cashiers Check

Account # _____ Exp. _____

1. _____	$	
2. _____	$	
3. _____	$	
4. _____	$	
5. _____	$	
6. _____	$	
Subtotal	$	
3% Charge Card fee	$	
Shipping & Handling	$	3.00
PA residents add 6% tax	$	
TOTAL	$	

Mastercard or Visa by mail or phone.
Mail cash, check, money order.
Personal check (1 add'l. week to clear)
$3.00 Shipping & Handling

DISCOUNTS ON THIRD-PARTY SOFTWARE AND EPSON PRINTERS

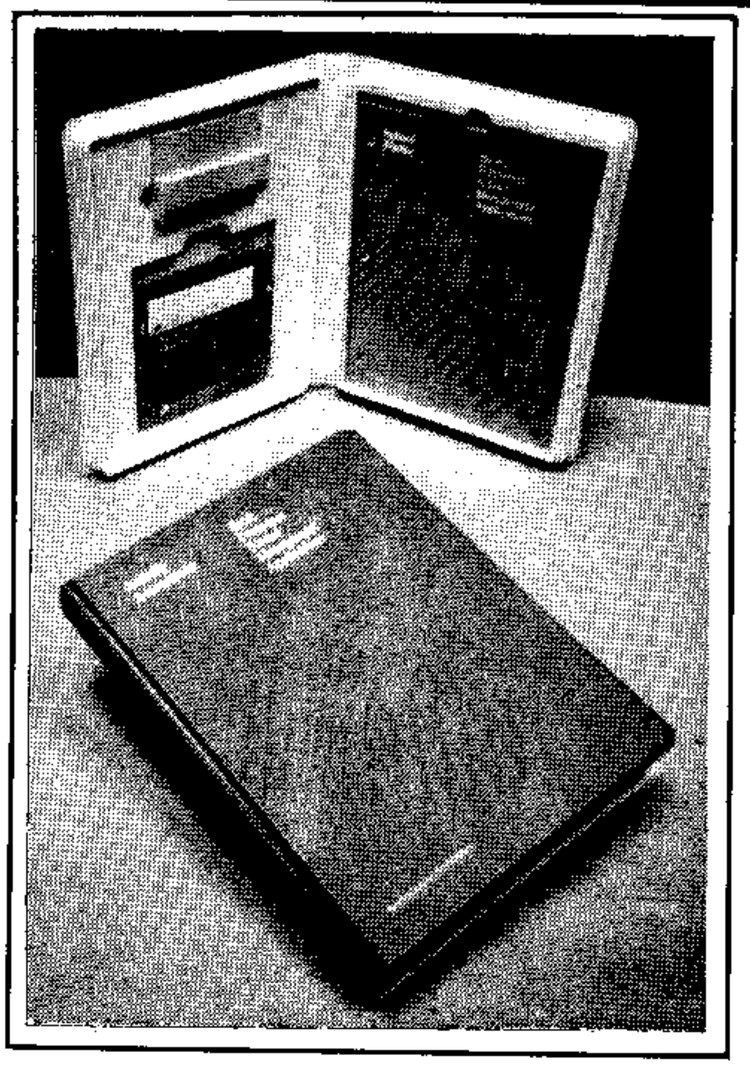
CALL FOR OTHER PRICES.
Equipment subject to price change and availability without notice.

Software Exchange

The Software Exchange is a program set up to offer low cost quality programs to TI-99/4 and 99/4A users, while at the same time encouraging authors to write more new programs. Anyone can participate—just send in your original program to tape or diskette, with a list of any peripherals needed. Each time this program is sold, the author will receive a $3.00 royalty. Programs will be sold for $4.00 to cover the cost of postage and the cassette or diskette. Catalogs of the programs submitted will be issued every two months and sent to all members. Members pay a $15.00 yearly fee to cover the cost of printing and postage of the catalogs.

You may also be interested in our regular line of software and peripherals. Catalogs are $1.00 (refundable on any order, free to members of the software exchange).

Data Systems
2214 W. Iowa St.
Chicago IL. 60622



THE SCOTT, FORESMAN School Management Applications Development Process

By Dr. Tom Hansen
2 Barrington East
Barrington, NH 03825

The Scott, Foresman School Management Applications are designed to help school district personnel with district, school, and classroom management tasks. The programs for the applications are stored on electronic chips placed inside a module that plugs into a Texas Instruments TI-99/4A microcomputer. The data that are entered for each application are automatically stored on diskettes. Data are displayed for entry and editing on a TV monitor screen, and reports are produced with a TI impact printer. For some applications, a card reader can be used for data input. The complete package of programs, hardware, and user manuals enables school districts to perform a wide range of computing functions at prices that are much lower than were possible before the microcomputer revolution came upon us.

The applications were developed as a joint project with Edusystems, Inc. (ESI) of Saint Paul, Minnesota. The development process included background research, developing descriptions of the applications, writing detailed specifications, writing several programmed versions of each application, and creating the final program chips and user manuals. The major steps in the development process are described below.

Background Research

The individuals who did the initial research to determine the potential applications had many years of experience in designing, developing, and using computer-based management applications in schools. The designers had experience with both *large*-computer school applications as well as *microcomputer* applications—experience that enabled them to develop a comprehensive list of feasible and useful microcomputer applications for school management. Selected school administrators, teachers, and clerical staff members of schools were then consulted for additional input regarding potential applications.

Application Descriptions

After the list of potential applications had been developed and reviewed, a one to two page description of each application was written. The descriptions listed the types and lengths of the proposed data elements that could be stored, the editing capabilities required, the types of reports that could be produced, and the total capacity of each application. The descriptions also illustrated how each application would be used in a school district. Every attempt was made to write the descriptions in language that could be easily understood by persons with no computer background. The descriptions were reviewed by school personnel, ESI designers and other staff members, Scott, Foresman personnel, and by consultants. The school personnel were from school districts located in Somerset, Wisconsin and Wolfeboro, New Hampshire. Because of their representativeness as districts that could benefit from the microcomputer-based applications, these districts served as the pilot sites throughout the development process. Their staff members had a great deal of impact upon the ultimate products. Based upon discussions and written comments from all persons who reviewed the descriptions, a final list of fourteen feasible School Management Applications was produced. This list, along with a brief summary of some of the functions of each application, is included here:

Accounting Assistant — Maintains expenditure accounts, helps monitor payments against budgets, records checks, purchase orders, and vendor data.

Activity Accountant—Records and accumulates building-level activity fund transactions. Activity fund accounts might include Spanish Club, Booster Club, etc.

Attendance Recorder — Records daily attendance from card or keyboard entry. It quickly produces a daily absence list, and also provides cumulative statistics for year-end reports.

Class Data Recorder — Calculates assignment grades and grade distributions. Reports showing student progress are produced.

Data Analyzer — Records, collates, and tabulates survey data, using card or keyboard entry. Various statistical analyses are performed on the data.

Mark Reporter — Records and analyzes course grades for a school year of two, three, or four terms.

Payroll Assistant — Processes salaries, other compensation, deductions, and benefits. It prints the paychecks.

Personnel Data Recorder — Maintains staff records of personal, certification, and payroll data. The payroll data are used and updated automatically by the Payroll Assistant application.

Property Manager — Organizes data on type, location, quantity, cost, age, condition, and ownership of equipment.

Salary Planner — Calculates and compares total cost and average salary for current and proposed salary schedules.

Scheduling Assistant — Summarizes student course requests by priority. Course request conflicts are identified for two to five courses at a time.

School Mailer — Produces self-adhesive mailing labels addressed to parents, guardians, or students, staff, etc. Also, lists of students sorted by grade, homeroom, club membership, etc., are produced.

Student Data Recorder — Maintains individual student records related to student accounting and services. It prints counselor lists, bus lists, locker lists, etc.

Test Scorer — Records, calculates, analyzes, and reports test scores from answer cards that are input through the card reader.

Detailed Specifications

Detailed specifications for each application had to be developed so that the content of each application could be more precisely refined and reviewed, and so that programmers could be given precise instructions to direct their work. The specifications were written such that an individual could follow the complete flow of the application even though no programming had yet been done. All data that could be accepted by the application were defined as to size and type (such as 10 spaces, letters

Streamline your administrative work

School Management Applications

MICROCOMPUTER SOFTWARE PACKAGES

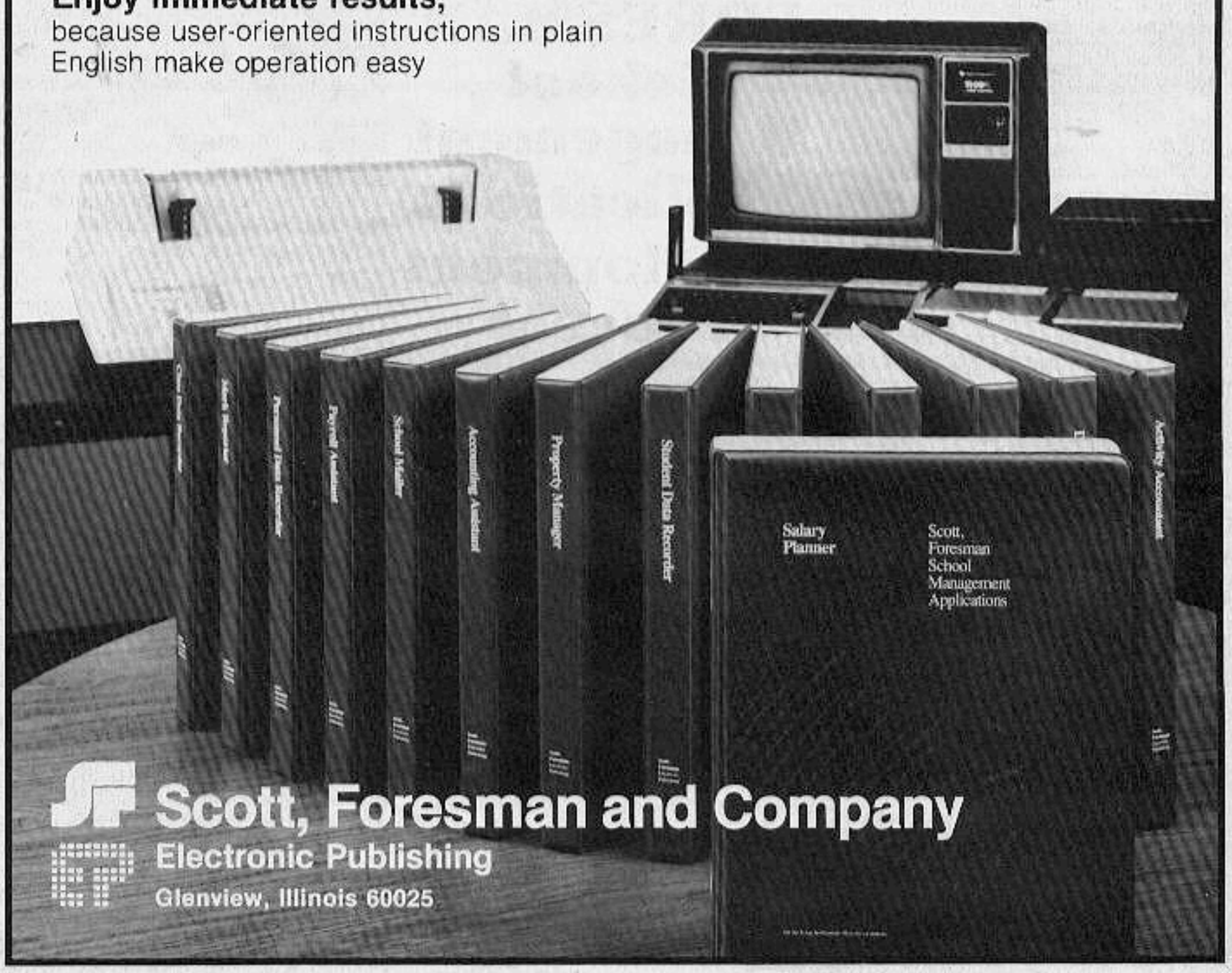
Reduce paper work
with professional, functional microcomputer programs

Simplify record-keeping
through convenient information storage capabilities

Save time and money
by producing reports, permanent records, mailing labels

Enjoy immediate results,
because user-oriented instructions in plain English make operation easy

Each School Management Application helps you handle a specific area of your work—from class scheduling to test scoring to mark reporting. Learn more about them all. Send for free, full-color brochure 30661-5-NN2.



Scott, Foresman and Company
Electronic Publishing
Glenview, Illinois 60025

only). Each separate screen that would appear on the TV monitor was drawn on a grid exactly the way it would look when the finished application was operated. The directions for the programmer were included on each grid sheet, such as "If the user does not use a comma between last and first names, display the message ' INSERT COMMA BETWEEN NAMES ' at line 12 on the screen." All of the input, editing, and report selection screens were developed for each application.

A precise version of each report was then developed on grid paper, along with special instructions for printing different versions of some reports. Sample data were included on each report.

The remaining part of the specifications package for each application was a diagram showing how the input screens, editing screens, and reports were related to each other. These diagrams demonstrated the logical paths that a user could follow through each application.

Several months of time were spent by the ESI staff and consultants in developing the detailed specifications. This large expenditure of time was felt to be necessary so that changes could be kept to a minimum once programming began. Time spent in developing comprehensive specifications facilitates a thorough understanding and review of the applications. The specifications were reviewed by most of the individuals who had previously reviewed the application descriptions. A thorough review of the thirty to fifty pages of specifications for an application gives the reviewers a clear picture of what the final product will look like. After revisions were made to the specifications, programming began on the first few applications.

EPROM Programming

After the specifications for an application were finalized, programming began on a Texas Instruments authoring computer. The TI-99/4A was not used to develop the programs since the larger authoring system was necessary to be able to use Graphics Programming Language (GPL) instead of BASIC when appropriate. Some of the more complex programs are written in this low-level language to conserve space and improve speed.

After a program was completed and tested at ESI, it was transfered from disk to erasable chips. Initially, the chips were "burned" at TI facilities in Texas, although this capability eventually was available at ESI. Up to ten chips were placed in an Erasable Programmable Read Only Memory (EPROM) box that was attached to a module that could be inserted into the TI-99/4A microcomputer. Boxes were then shipped to the two pilot sites for testing by the pilot site coordinator and school district staff. Extensive testing was also conducted at ESI and Scott, Foresman. Errors, omissions, and enhancements were documented, and a revised version of the application was developed and placed in EPROM boxes.

It was essential that *re-usable* chips be employed at this stage of development because it was sometimes necessary to program several versions of each application before a final product was produced. In most cases, actual data from the school districts were used for testing. District personnel evaluated the data input process, editing, report selection and usability, etc. Depending on the type of application, numerous staff members used the applications for several hours to several months. At either a pilot site, ESI, or at Scott, Foresman offices, each application was also tested with the largest amount of data that it could possibly store and process.

"The necessity for a bug-free program is somewhat new to the computer industry, and certainly puts added pressure on software designers and programmers. Realistically, there is no second chance once the final chips are created."

Final Burning of the Chips

It is essential that a program be completely bug-free before the final chips are created. The chips used in the final module are not erasable. Also, since the production run for the chips is quite large each time, any programming error would cause the final production chips to be useless and would result in a loss of several thousands of dollars. The necessity for a bug-free program is somewhat new to the computer industry, and certainly puts added pressure on software designers and programmers. Realistically, there is no second chance once the final chips are created. This fact is a major motivation for designers to develop precise specifications that a programmer can understand, and for thoroughly testing the EPROM versions of the programs.

Writing User Manuals

An important part of any School Management Application is the user manual. Once the programs were nearly in final form, work began on the manuals. It was felt that for each application the users should have one manual that provides all of the hardware and software information they need. There are two parts to the manual that comes with each application: The first part describes hardware unpacking, set-up, and operation; the second part deals with the particular application. The manual tells how the application can help school district personnel, and how to use every option available in the application. The manuals were developed to follow the flow of each application from start to finish. Sample screens and reports are included, along with precise directions about how to enter data, edit, select reports, and interpret reports. When appropriate, full-size data collection forms are included so that users can reproduce them and have an organized approach to data collection and entry. The manuals also contain troubleshooting suggestions, a microcomputer glossary, a complete index, and a diagram illustrating how the screens and reports are inter-related.

User manuals have always been an important part of any computer system. They become even more important *now* because the introduction of microcomputers means that individuals with no previous computer experience may be given the task of providing districts with computer services. The School Management Application manuals were developed with the assumption that the users would not have had experience with traditional large systems. Nor would they be expected to be microcomputer hobbyists who have become accustomed to searching through numerous manuals, or learning from fellow hobbyists the secrets of their microcomputer systems. Educators should not have to use the "discovery approach" to operate a microcomputer system. A great deal of time was spent writing and reviewing the School Management Application manuals so that users can operate their computer systems without the need for endless hours of reading and cross-referencing or for outside assistance.

The six major development stages described above were spread over a period of nearly two years. This type of controlled and thorough development process is essential in order to develop microcomputer-based management applications that are valuable to district personnel, easy to use, and affordable.

Name That Bone . . . from p. 72

```
1370 DATA 97,0001030307070707,98,0F0F0F0F0F0F0F0F,99,07070707030301,100,B1C3C7EFFFFFFFFF
1380 DATA 101,F7F3F0F0F0F3F7FF,102,FFFFFFFFF7F7E3C1,113,0000000000FFFFFF,121,7E7E7E7E007E7E7E
1390 DATA 129,FEFEFEFEFE000000,130,FEFEFEFEFE0000FE,131,FEFEFE0000FEFEFE,137,FFFFFFFE000080C
1400 DATA 138,FFFFFFFFFF0000FF,139,B0C0C0C0800000FE,140,FFFFFFFF0000FFFF,141,FFFFFF0000FFFFFF
1410 DATA 142,FFFFFF0000C0E0E0,143,C0C08,145,000001070F1F7FFF,146,1878F8F8F0E0C08
1420 DATA 64,0000001F1010101,94,101010101010101,95,101010101F
1430 N=23
1440 GOSUB 160
1450 PRINT TAB(21);"@PHLANGE";TAB(12);"RADIUS    ^"
1460 CALL HCHAR(22,31,83)
1470 PRINT :::TAB(12);"ULNA  ^_METACARPU"
1480 CALL HCHAR(23,31,83)
1490 PRINT " HUMERUS";:TAB(18);"_CARPUS";::::::::::
1500 CALL VCHAR(12,20,94,2)
1510 RESTORE 1520
1520 DATA 8,2,97,1,9,2,98,1,10,2,99,1,8,3,96,8,9,3,96,8,10,3,96,8,8,11,100,1
1530 DATA 9,11,101,1,10,11,102,1,8,12,104,8,9,12,113,8,10,12,112,8,8,20,121,1,9,20,121,1
1540 DATA 10,20,121,1,7,21,145,1,8,21,129,1,9,21,130,1,10,21,131,1,7,22,146,1,8,22,138,2
1550 DATA 9,22,140,2,8,24,139,1,9,24,137,1,10,22,141,1,10,23,142,1,10,24,143,1,10,29,32,1
1560 N=27
1570 GOSUB 320
1580 GOSUB 230
1590 N=7
1600 RESTORE 1610
1610 DATA 6,23,32,9,7,14,32,6,7,23,32,1,11,14,32,18,12,4,32,19,13,20,32,1,14,20,32,7
1620 GOSUB 320
1630 R=6
1640 DATA HUMERUS,9,RADIUS,10,ULNA,11,CARPUS,12,METACARPUS,13,PHLANGES,14
1650 RESTORE 1640
1660 GOSUB 370
1670 GOTO 1650
1680 RESTORE 1690
1690 DATA 43,FFFF00FFFFFFFFFF,33,FFFFFFFF1F1F1F0F,34,FFFFFFFFE0F8FCFE,35,FFFFFFFF
1700 DATA 36,FFFFFFFF071F3F7F,37,FFFFFFFFF8F8F8F8,97,0707070707070707,100,E0E0E0E0E0E0E0E0
1710 DATA 98,00C0F0FCFEFFFFFF,101,00030F3F7FFFFFFF,99,FF1F0701,102,FFFBE08,105,3FFFFFFF1F1FFFFF
1720 DATA 106,FFF7E7F7F7F7F7FF,107,FCFFFFFFF8F8FFFF,108,070707FFFFFF0101,109,E0E0E0FFFFFF808
1730 DATA 113,000000FFFFFF,114,060F1F03C0C0FFFF,117,C0F0F8C00303FFFF,115,0000C0E18F1F
1740 DATA 116,06060CF8F8F,119,6060301F1F0F,121,000103070F1F3F7F,125,0080C0E0F0F8FCFE
1750 DATA 123,7F3F1F0F070301,127,FEFCF8F0E0C08,133,EFF7E7E7C381,132,F7EFE7E7C381,145,7D413E3E1C08
1760 DATA 122,000000FFFFFFFFFF,137,7C7E3F3F0F0F0701,141,3E7EFCFCF0F0E08,138,0001010107FFFFFF
1770 DATA 140,00808080E3FFFFFF,139,FFFFFFF7F7E3C1,129,FFC1DDF9F3E7CFC1,151,FFFFFFC1FDFDE1F0
1780 DATA 130,00C1E1F7F7F7F7F7,134,008387EFEFEFEFEF,131,EFEFEFEFEFEFEFEF,135,F7F7F7F7F7F7F7F7
1790 N=42
1800 GOSUB 160
1810 CALL COLOR(1,12,1)
1820 CALL COLOR(2,12,1)
1830 PRINT TAB(18);"CLAVICLE":::TAB(22);"SCAPULA":"1 STERNUM"
1840 PRINT ::TAB(22);"RIBS":::::TAB(16);"SPINE":::"2 SACRUM"
1850 PRINT TAB(20);"ILIUM":"3 COCCYX"::TAB(18);"ISCHIUM":::
1860 RESTORE 1880
1870 CALL VCHAR(2,17,43,14)
1880 DATA 4,11,33,1,4,12,34,1,4,13,35,3,4,16,36,1,4,18,34,1,4,19,35,3,4,22,36,1,4,23,37,1
1890 DATA 5,11,97,1,5,12,96,1,5,13,98,1,5,15,114,1,5,16,105,1,5,17,106,1,5,18,107,1,5,19,117,1
1900 DATA 5,21,101,1,5,22,96,1,5,23,100,1,6,11,97,1,6,12,96,2,6,14,113,2,6,16,108,1
1910 DATA 6,17,104,1,6,18,109,1,6,19,113,2,6,21,96,2,6,23,100,1,7,12,99,1,7,13,113,9
1920 DATA 7,17,104,1,7,22,102,1,8,12,113,11,8,17,104,1,9,12,113,4,9,16,116,1,9,18,119,1,9,19,113,4
1930 DATA 10,11,113,5,10,19,113,5,11,11,113,4,11,15,116,1,11,19,119,1,11,20,113,4,12,10,113,5
1940 DATA 12,20,113,5,13,10,113,4,13,14,116,1,13,20,119,1,13,21,113,4,16,13,121,1,16,14,120,3
1950 DATA 16,16,130,1,16,17,128,1,16,18,134,1,16,19,120,2,16,21,125,1,17,13,120,3,17,16,131,1
1960 DATA 17,17,129,1,17,18,135,1,17,19,120,3,18,13,120,3,18,16,133,1,18,17,151,1,18,18,132,1
1970 DATA 18,19,120,3,19,13,123,1,19,14,120,2,19,17,145,1,19,19,120,2,19,21,127,1,20,14,123,1
1980 DATA 20,15,120,1,20,16,122,3,20,19,120,1
1990 DATA 20,20,127,1,21,15,137,1,21,16,138,1,21,17,139,1,21,18,140,1,21,19,141,1,24,30,32,1
2000 N=84
2010 GOSUB 320
2020 GOSUB 230
2030 RESTORE 2040
2040 DATA 3,20,32,8,7,3,32,9,6,24,32,7,10,24,32,4,14,18,32,5,18,22,32,5,17,3,32,8
2050 DATA 19,3,32,8,21,20,32,7,5,17,104,1,17,17,128,1,18,17,128,1,18,17,128,1
2060 N=13
2070 GOSUB 320
2080 CALL CHAR(145,"7F7F3E3E1C08")
2090 DATA CLAVICLE,1,SPINE,2,SCAPULA,9,STERNUM,10,
     RIBS,11,ILIUM,12,SACRUM,13,COCCYX,15,ISCHIUM,14
2100 R=9
2110 RESTORE 2090
2120 GOSUB 370
2130 CALL VCHAR(16,13,121)
2140 CALL VCHAR(17,13,120,2)
2150 CALL VCHAR(19,13,123)
2160 GOTO 2110
2170 RESTORE 2180
2180 DATA 97,F0F8F8F8F0E0C08,98,071F7F7F7F7F7F7F,
     99,3F1F0F0707030301,100,FFFFFFFFFFFFEFC
2190 DATA 105,7E7F7F7F7F7F7F3E,113,7FFFFFFF7F7F7F7F,
     114,00C0FFFFFFFFFFFF,115,FFFFFFFFFFFFFFE8
2200 DATA 122,FFCF8F8F0F0F0F0F,121,0F0F0F0F0F0F0F0F,
     129,3E7FFFFFFFFFFF7F,137,1C3F1F0F07030F0F
2210 DATA 138,38BEBFBF00FBFEFF,139,FFFF8F0F070707,
     140,FFFF01FEFF01FEFF,145,FCFEFF03F8FE07E
2220 DATA 146,FB0EF678,147,1F0F03,130,
     00F0FEFFFFFFFFFE
2230 N=19
2240 GOSUB 160
2250 PRINT TAB(14);"FEMUR":::::::TAB(14);
     "PATELLA":::TAB(7);"TIBIA"
2260 PRINT :TAB(14);"FIBULA":::::TAB(14);
     "TARSUS"::"  METATARSUS     PHLANGES"
2270 CALL VCHAR(2,14,96,9)
2280 CALL VCHAR(2,15,96,8)
2290 CALL VCHAR(13,14,112,7)
2300 CALL VCHAR(14,15,121,7)
2310 RESTORE 2320
2320 DATA 2,13,98,1,2,16,97,1,3,13,99,1,10,15,100,
     1,11,14,100,1,11,15,105,1,12,14,113,1
2330 DATA 12,15,114,1,13,15,122,1,20,14,115,1,21,
     14,129,1,21,15,130,1,22,14,137,1,22,15,138,1
2340 DATA 23,15,139,1,23,16,140,2,23,18,145,1,24,
     18,146,1,24,17,147,1,24,30,32,1
2350 N=20
2360 GOSUB 320
2370 GOSUB 230
2380 RESTORE 2390
2390 DATA 4,16,32,5,11,16,32,7,14,9,32,5,16,16,32,
     6,21,16,32,6,23,5,32,10,23,19,32,8
2400 N=7
2410 GOSUB 320
2420 R=7
2430 DATA FEMUR,9,PATELLA,10,TIBIA,11,FIBULA,12,
     TARSUS,13,METATARSUS,14,PHLANGES,15
2440 RESTORE 2430
2450 GOSUB 370
2460 GOTO 2440
2470 STOP
2480 CALL CLEAR
2490 PRINT TAB(7);"NAME THAT BONE":::::::::::::::::::
2500 A$="FFFFFFFFFFFFFFFF"
2510 CALL CHAR(96,A$)
2520 CALL COLOR(9,7,1)
2530 CALL HCHAR(7,15,96,3)
2540 CALL HCHAR(8,15,96,3)
2550 CALL HCHAR(9,15,96,3)
2560 CALL VCHAR(10,16,96,6)
2570 CALL HCHAR(11,13,96,7)
2580 CALL VCHAR(15,15,96,6)
2590 CALL VCHAR(15,17,96,6)
2600 CALL COLOR(2,2,1)
2610 IF FLAG=2 THEN 2820
2620 FOR I=1 TO 7
2630 CALL COLOR(9+I,12,1)
2640 CALL CHAR(96+8*I,A$)
2650 NEXT I
2660 FLAG=2
2670 PRINT "INSTRUCTIONS? (Y/N)"
2680 CALL KEY(0,K,S)
2690 IF K=78 THEN 2820
2700 IF K<>89 THEN 2680
2710 CALL CLEAR
2720 PRINT "YOU MAY STUDY THE NAMES OF":
     :"THE BONES AS LONG AS YOU"
2730 PRINT :"WISH. THEN PRESS <ENTER>."
2740 PRINT ::"THE LABELS WILL CLEAR, THEN"
     ::"IT IS YOUR TURN TO NAME"
2750 PRINT :"THAT BONE - CHOOSE THE":
     :"CORRECT NUMBER."
2760 PRINT ::"YOU MUST NAME THE BONES":
     :"CORRECTLY TO CONTINUE.":::
2770 GOSUB 230
2780 FLAG=2
2790 GOTO 2480
2800 DATA 67,72,79,79,83,69,58,49,
     32,72,69,65,68,32,50,32,65,82,
     77,83,32,51,32
2810 DATA 84,79,82,83,79,52,32,76,69,71,
     83,32,53,32,69,78,68,32,32
2820 RESTORE 2800
2830 CALL HCHAR(23,1,32,21)
2840 FOR X=7 TO 17 STEP 2
2850 FOR Y=23 TO 29
2860 READ G
2870 CALL HCHAR(X,Y,G)
2880 NEXT Y
2890 NEXT X
2900 CALL KEY(0,K,S)
2910 IF S<1 THEN 2900
2920 IF K=53 THEN 2980
2930 IF (K>52)+(K<49)=-1 THEN 2900
2940 CALL CLEAR
2950 PRINT "ONE MOMENT PLEASE":::
2960 CALL COLOR(9,12,1)
2970 ON K-48 GOTO 990,1360,1680,2170
2980 CALL CLEAR
2990 END
```

Space Station . . . from p. 50

only complaint I have with an otherwise enjoyable product. An improvement in documentation would definitely be in order.

The actual program is well thought out. For example, you press the ENTER key to start it up. At the same time, pressing the key has told the program whether you have the TI-99/4 or 99/4A. As soon as you are ready to go, just press the ENTER key again (if you are using the keyboard) or the "fire" button (if you use the joysticks). This starts the game and tells the computer whether to read the keyboard or joysticks for the rest of the game.

Once you get the hang of the game, one fact stands out: Assembly Language games are fast! The graphics are very smooth, and there is no hesitation—no delays—while the program is running. The game speed is even fast enough for you to lay down a "spread" of torpedoes. This is possible because you don't have to lift up your finger from the fire button for the program to obey the cross-hair movement commands. In TI BASIC, of course, you cannot get your program to read two keys from the keyboard simultaneously.

The game becomes very intriguing as the pace speeds up. It develops the feel of a real arcade game. In fact, if I had better joysticks, this game would have become addictive to me. [Here at *99'er* headquarters, we noticed quite an improvement with both the *new* TI joysticks, and Atari joysticks with the Denali Data adapter—Ed.] I also had a hard time controlling the direction keys when I used the keyboard instead of joysticks; it seemed to require more manual dexterity than I'm personally capable of—especially when the action began to really speed up.

The graphic characters are well done. The game screen is slightly cluttered with all the combatants moving around—i.e., the space station, alien attackers, mother ships, comets, target cross-hairs etc. It takes a while until you learn to ignore the unimportant clutter. Until you do, it is rather difficult to react to the alien attackers quickly enough to survive very long.

To summarize, this product has some very good points. For one thing, it really shows what the 9900 Assembly Language in the hands of a good programmer is capable of doing. The graphics are well done and the program is well thought out. It has the speed needed to make this kind of game very challenging. The only real complaint I had was with the documentation. Overall, this is a very good arcade-type product. If you like these kind of games, give it a try and see what a joy it is to play an original game instead of just another rehash of *Space Invaders.*

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 PRINT

ADDRESS _____

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, educatio business, professional, and home applications. Also, it is a valuab information resource for those who want to know more about these friendly, value-packed personal computers in order to mak an intelligent purchase decision in the future . . .

A 4-in-1 Resource

As a special bonus, 99'er Magazine includes the bound-in publications—*OnLoCAltion: 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.

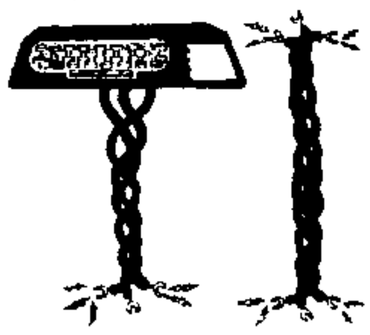
99'er magazine™ A Resource for People Interested in the Enrichment of Personal Computing

Satisfaction Guaranteed

Or the unfilled portion of your subscription will be refunded

℠ A service mark of Emerald Valley Publishing C

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

How can I get an 80-column display for my TI-99/4A?

To date, TI is not offering any system component that will allow the 99/4A to output an 80 column display. However, using the TI RS232 Interface you may connect an 80 column display terminal to the system. This is an expensive alternative. A terminal may cost from $350.00 to $1200.00 depending on the resolution of the screen, and how many extras it may have. I have used this technique with a homebrew text editor program. By "opening a file" for the terminal, I could communicate with it using Extended BASIC statements:

```
LINPUT  #1 : A$
PRINT  #1 : B$
```

Now that TI has introduced the new Peripheral Expansion System, perhaps they (or someone else) may choose to offer an 80-column video display card for a black & white monitor.

Can I use other Disk Controllers with my TI-99/4A?

No. At this time there are no other disk controllers that will directly work with the TI-99/4A system. A disk controller is like a language interpreter at the United Nations. It can speak a "computer dialect" and a "disk drive dialect." The TI Disk Controller speaks the very special "99/4A computer dialect" and a fairly common "5¼ inch disk drive dialect." Although you cannot use a different disk controller with the 99/4, you *can* use several different makes of disk *drives* with the TI Disk Controller. However, if you don't have some background in electronics, don't purchase a non-TI disk drive unless the manufacturer specifically states their drive is compatible with the TI Disk Controller.

Can you tell me more about opening and closing files?

Okay, I'll give that a shot. A "file" as used with the TI-99/4A system (and many other computer systems) can mean many things. It can mean a port on the RS232 interface, a cassette recorder port, a disk drive file, or even a section of special memory that is not normally accessible. Generally, a "file" is some place or thing with which the computer wants to exchange information. To establish this exchange we "open" the file. To terminate the exchange we "close" the file. When the computer executes an OPEN statement, it sets aside a small portion of random access memory. This portion of memory will be used to keep track of the exchange of information between the computer and the file. All the parameters that were specified in the OPEN statement are kept in this block of memory. During the execution of the OPEN statement, the computer tries to exchange information with the file. If it is successful, it puts the current status of the file into the memory block for future use and assigns the "file number" to the block of memory. Now when a statement like PRINT #1 : . . . . . is executed, the computer first goes to the block of memory associated with file #1 to see if that type of exchange is permitted and how it is to be done. After all information exchange is complete, we must close the file. When the CLOSE statement is executed, the computer checks for things like pending prints, pending inputs, and if the file is open—i. e., has a block of memory setup for this file number. After deciding how to take care of any "housekeeping" required, the file number assignment is removed and the block of memory is released for other uses. This is a greatly simplified version of a sophisticated process. [To learn about "random access" files, see the *Getting Down to Business* article in this issue—Ed.]

Can I PEEK and POKE with or without Extended BASIC, and is it really always necessary?

PEEK and POKE are fairly standard functions found in most BASIC language implementations. They are used to see (PEEK) what binary patterns are at specific locations of memory and to write (POKE) binary patterns directly in specific locations of memory. TI BASIC does not have these functions directly available. But with the new *Mini-Memory* Command Module plugged in, TI BASIC will have these functions. TI Extended BASIC does have PEEK and POKE functions in the form of subprograms. CALL PEEK (address, numeric-variable-list) is the PEEK function and CALL LOAD (address, byte1[, . . .]) is the POKE function. The PEEK and LOAD subprograms are generally not needed unless you are using Assembly Language subroutines from BASIC.

Can I use different MODEMs or just the TI MODEM?

There are many MODEMs (MODulator-DEModulator) on the market that will work with the TI RS232 Interface. In my case, I found that my "Princess" style, touch-tone, modern-shaped telephone would not fit well in the foam rubber cups of an acoustic coupler type of MODEM (such as TI's). So I bought a "Hayes Smart MODEM" which is a direct-connect type. This means that it has a cable that plugs right into the telephone line rather than using an acoustic coupler type device. The other connection from the MODEM is cabled to the TI RS232 Interface.

TIP: Some MODEMs will require that one end of the RS232 cable be modified. The modification is to reverse the two wires in pins 2 and 3 followed by reversing the wires between pins 6 and 20. Yes, the Hayes Smart MODEM is one of these.

How can I get my Epson MX-80 to print 132 characters on one line?

First make sure the printer is setup for the 16.5 cpi mode (see the printer's manual for the control sequence). When opening the printer's RS232 port, make sure to disable the automatic generation of carriage return (CR) and line feed (LF). Example: OPEN #1 : "RS232/2.DA=8.BA=9600. CR. LF",VARIABLE 132 It will be necessary to explicitly send the carriage returns and line feeds to the printer:

```
CRLF$= CHR$(10)&CHR$(13) : : PRINT #1 : WORDS
$&CRLF$
```

Buying . . . from p. 21

prices. If you can afford it and desire peace of mind, buy locally.

In the case of TI computers, you can exchange the defective unit for a factory rebuilt unit at one of the exchange centers. It won't matter where you originally purchased the unit. You can check with your local dealer to see if a service center is near you.

Another point to consider is that we really should not abuse the local computer store owner's time by letting him educate us if we have no intention of buying locally. It is fair to expect him to compete with other dealers for our dollar by demonstrating his wares and services, but unfair to sit through an hour or two of free demonstrations if we've already decided to buy through the mail. After all, we want the computer store to succeed, since it will advance personal computing in general.

Miscellaneous Points

Ask the salesperson if the computer you select can perform the graphics, sound, and text functions you desire just as it comes out of the box, or must you buy additional attachments or plug-in devices. You may find the demonstration you witnessed on a "loaded" floor model cannot be performed on a basic unit without adding several hundred dollars of additional equipment. On the other hand, you may find that most of the desirable features are built right into the basic computer.

It is also essential to have clear, concise, easily understood manuals which explain how to use your computer. You should not have to have any knowledge about computers to understand the basic introductory and tutorial manuals for your computer.

If you have not yet bought that first computer, be assured that you are embarking on an exciting adventure. The excitement and pride you'll experience when opening the box on that first day is like a dozen Christmas celebrations combined. Enjoy the experience, and realize that ownership is not only *exciting* but *helpful* and *productive* too.

In the meantime read all you can and shop carefully until you just can't stand it any longer . . . Then take the plunge. Go out and get that computer!

GLOSSARY OF TERMS

BASIC — Beginners All Purpose Symbolic Instruction Code is a program language developed at Dartmouth in the early 60's; it is the most common of all programming languages for small computers. BASIC is relatively easy to learn and is an effective and powerful language for most small computer applications.

bit — The smallest piece of information your computer deals with. It is equivalent to a circuit being turned either on or off. Like a light bulb, a computer logic circuit is either on or off; this equals one bit of information. Most home computers use an 8-bit microprocessor, but Texas Instruments and IBM have a 16-bit microprocessor. The advantages of the 16-bit configuration are too technical for this discussion, but we can generally say that more powerful and accurate computing can be accomplished. It has been predicted that the 16-bit microprocessor will be the future industry standard.

byte — The amount of memory necessary to code a character (a number/letter/punctuation, etc.) A byte has 8 bits in it. A computer which has 16K bytes of memory has 16 thousand bytes and can work with about 16 thousand characters of information in a single program.

chip — The circuits of the computer are fabricated on silicon chips. A chip is typically about ¼ inch on a side. Today's chips are so sophisticated that the basic components of an entire computer can be fabricated on a single chip.

CRT (monitor) — The TV-like screen (cathode ray tube) to which the computer outputs information like numbers/letters/graphs, etc.

disk drive — The accessory which stores and retrieves information on plastic (mylar) diskettes. The DOS (see below) controls the operation of the disk drive.

disk operating system — Sometimes called DOS and sometimes pronounced like "DOSS." It is the set of instructions (software) which controls the storing and retrieving of information with the disk drive.

diskette — A plastic disk coated with an oxide upon which data and programs are stored using the disk drive under control of the DOS. Diskettes come in either of two sizes, 5¼ inch or 8 inch. The TI-99/4A uses the 5¼ inch.

firmware — Generally speaking, firmware is a chip in which a program has been stored permanently. It is "soft" in that it is a program (see software) but "hard" to the extent that it is an electronic chip rather than a diskette or tape. Hence it is "firmware." Firmware is used to store programs which are used repeatedly, and need not be changed or modified. (see **ROM**)

hardware — The actual physical machine, i.e. keyboard, CRT, printer, etc.

integrated circuit (IC) — If you look into the back of an old radio, you will see a lot of resistors, capacitors, and the like. Each component will be discreet—i.e., separate from other resistors, etc. which surround it. Integrated circuits, on the other hand, have many such individual components packed together or integrated in a small area. (see **chip**) If you peer into a computer, you will see rows of little black boxes plugged into circuit boards. Each little black rectangle may have thousands of components integrated into it.

input/output (I/O) — Input is the data that goes into the computer via the keyboard as well as disk drives, tape recorders, etc. Output is what comes back out of the computer to the monitor screen, disk drive, tape recorder, and printer. (**Throughput** is what happens in between).

microcomputer — All computers used to be very large and esoteric and were called "mainframes." But miniaturization with integrated circuits has resulted in very powerful computers of small size coming into being. That is, you could pack a lot of computer into a very small box. These computers were initially called "minicomputers." But as the reduction in size continued, small desktop-size computers were produced with sufficient computing capacity to still be very useful. These are called "microcomputers." The difference in power between the mini and the micro is diminishing rapidly, so that it will soon be difficult to tell a mini from a micro. For now, all home computers are considered microcomputers.

peripherals — All those hardware devices which plug into your computer such as disk drives, tape recorders, printers, and modems.

printer — A peripheral device which will print a copy (called hardcopy) of your computer's output. Very handy to have for correspondence and for program debugging.

program — The set of coded instructions which directs the activities of your computer. Without a program, your computer is just so much metal and silicon junk. (see **software** and **BASIC**)

RAM — Traditionally, the abbreviation for random access memory. But the name is a little misleading. Both RAM and ROM memory are random access. More accurately, RAM should be described as read and write memory (contrast with ROM). RAM is the memory you are using when you program a computer. It is also the memory to which your computer salesperson is referring when he says, "This one has 16K memory." The more RAM you have, the bigger programs you can run. When you turn your computer off, all the contents of RAM is erased. So if you wish to avoid having to type in hundreds of program lines everytime you use your computer, you must save programs on tape or disk for future use.

modem — A device that connects your computer to the telephone so you can communicate with other computers. It works by modulating and demodulating a sound tone.

ROM — This is read only memory. That's right, you cannot "write" anything to a ROM; you can only "read" it. This means that you cannot change the contents of a ROM memory like you can a RAM memory. ROM contents are usually not changed, therefore they are used for firmware. (see above)

RS-232C — A peripheral device which plugs in between (interfaces) the computer and some other device like a modem or a printer.

software — It is not the physical machine (**hardware**) and usually not the permanent programs stored on chips (**firmware**) that instructs the computer on how to perform a task. It is the program stored on disk or tape. You can see that a tape or plastic disk is not as much a part of the computer as a chip (not as "firm"), therefore the programs stored on tape or disk are called "**software**." [See also, "Command Module" in the glossary of the *99'er Road Map* on p. 8]

Rule of 78 . . . from p. 11

```
1380 ADED=DEDUCT/DEN*FC
1900 SAV=RUL 78(AFC)
2090 DEDUCT=DEDUCT-RUL 78
     (FC)
2160 SAV=1/DEN*AFC
2380 DED=RUL 78(FC)
2470 DED=NP/DEN*FC
2530 DED=RUL 78(FC)
2600 DED=1/DEN*FC
```

The subroutine beginning at line 2770 is not required if an ACCEPT statement is used to input the character. Omit line 2730 and modify lines 2720 and 2740 as follows:

```
2720 ACCEPT AT(24, 14): SEL$
2740 IF SEL$="9" THEN 2760
```

And don't forget to omit the subroutines (line 2770-3030). Extended BASIC allows further compression by putting several statements on the same line and by using statements in IF THEN ELSE statements. However, the changes I have suggested provide a significant reduction in the size of the program.

With this program in your computer, you have all the secrets of the Rule of 78 at your disposal. Your computer will tell you everything you ever wanted to know about an installment contract, but didn't ask because you would not have been told.

```
100 REM  ****PAYMENTS****
         *  RULE OF 78  *
110 REM  ********************
110 REM   BY HARLEY M. TEMPLETON
          99'ER VERSION 1.5.1
120 REM   DEFINE FUNCTIONS
130 DEF RND2(X)=INT(X*100+.5)/100
140 DEF INC=28-LEN(X$)
150 DEF RUL78(D)=P*Q/2/DEN*D
160 REM  INPUT DATA ********
170 CALL CLEAR
180 PRINT "    INSTALLMENT PAYMENTS":::
190 INPUT "AMOUNT FINANCED: $":UB
200 INPUT "FINANCE CHARGE: $":FC
210 INPUT "ACQUISITION CHARGE: $":AC
220 AFC=FC-AC
230 INPUT "AMOUNT OF PAYMENT: $":PMNT
240 INPUT "NUMBER OF PAYMENTS: ":NP
250 INPUT "FIRST PAYMENT DATE: ":DATE$
260 N=POS(DATE$,"/",1)
270 MO$=SEG$(DATE$,1,N-1)
280 M=POS(DATE$,"/",N+1)
290 L=M-N-1
300 DA$=SEG$(DATE$,N+1,L)
310 YR$=SEG$(DATE$,M+1,2)
320 MO=VAL(MO$)
330 DA=VAL(DA$)
340 YR=VAL(YR$)
350 DEN=NP*(NP+1)/2
360 PRINT :"CHOOSE ONE:"::
370 PRINT " 1 CONTRACT SCHEDULE"
380 PRINT " 2 CONTRACT STATUS"
390 PRINT " 3 TAX DEDUCTION"
400 PRINT " 4 NEW CONTRACT"::
410 INPUT "ENTER NUMBER: ":SEL
420 IF SEL=4 THEN 170
430 IF (SEL<1)+(SEL>3)<0 THEN 360
440 CALL CLEAR
450 X$=STR$(UB)
460 GOSUB 2900
470 PRINT "AMOUNT FINANCED: $";TAB(INC);X$
480 X$=STR$(FC)
490 GOSUB 2900
500 PRINT "FINANCE CHARGE: $";TAB(INC);X$
510 X$=STR$(AC)
520 GOSUB 2900
530 PRINT "ACQUISITION CHARGE:";TAB(20);"$";
    TAB(INC);X$
540 X$=STR$(PMNT)
550 GOSUB 2900
560 PRINT "AMOUNT OF PAYMENT:";TAB(20);"$";
    TAB(INC);X$
570 X$=STR$(NP)
580 PRINT "NUMBER OF PAYMENTS:";TAB(26);X$
590 X$=DATE$
600 PRINT "FIRST PAYMENT:";TAB(INC);DATE$
610 ON SEL GOSUB 640,1520,2260
620 GOTO 360
630 REM  DISPLAY SCHEDULE **
640 PRINT ::"    CONTRACT SCHEDULE"::
650 PRINT "AFTER PAYMENT ON"
660 PRINT "TOTAL PAID      $"
670 PRINT "BALANCE         $"
680 PRINT "PREPAY AMOUNT   $"
690 PRINT "SAVE BY PREPAY $"::::::
700 CYR=YR
710 CMO=MO
720 TOTPD=0
730 DEDUCT=0
740 BAL=UB+FC
750 N=0
760 FOR I=NP TO 1 STEP -1
770 X$=STR$(CMO)&"/"&DA$&"/"&STR$(CYR)
780 CALL HCHAR(14,21,32,9)
790 C=INC+1
800 R=14
810 GOSUB 2840
820 CMO=CMO+1
830 TOTPD=TOTPD+PMNT
840 X$=STR$(TOTPD)
850 GOSUB 2900
860 C=INC+1
870 R=15
880 CALL HCHAR(15,19,32,11)
890 GOSUB 2840
900 BAL=BAL-PMNT
910 X$=STR$(BAL)
920 GOSUB 2900
930 C=INC+1
940 R=16
950 CALL HCHAR(16,19,32,11)
960 GOSUB 2840
970 P=I-2
980 IF P=1 THEN 1350
990 IF P<0 THEN 1330
1000 Q=P+1
1010 SAV=RND2(RUL78(AFC))
1020 IF SAV<1 THEN 1330
1030 PREPAY=BAL-SAV
1040 R=17
1050 X$=STR$(PREPAY)
1060 GOSUB 2900
1070 C=INC+1
1080 CALL HCHAR(17,19,32,11)
1090 GOSUB 2840
1100 R=18
1110 X$=STR$(SAV)
1120 GOSUB 2900
1130 C=INC+1
1140 CALL HCHAR(18,19,32,11)
1150 GOSUB 2840
1160 N=N+1
1170 DEDUCT=DEDUCT+I
1180 IF CMO>12 THEN 1380
1190 GOSUB 2630
1200 NEXT I
1210 IF DEDUCT=0 THEN 1320
1220 Q=DEDUCT
1230 ADED=RND2(DEDUCT/DEN*FC)
1240 C=2
1250 R=20
1260 X$=STR$(ADED)
1270 GOSUB 2900
1280 CALL HCHAR(20,23,32,9)
1290 X$="DEDUCTION FOR 19"&STR$(CYR)&" $"&X$
1300 GOSUB 2840
1310 GOSUB 2630
1320 RETURN
1330 SAV=0
1340 GOTO 1030
1350 X=1/DEN*AFC
1360 SAV=RND2(X)
1370 GOTO 1020
1380 ADED=RND2(DEDUCT/DEN*FC)
1390 X$=STR$(ADED)
1400 GOSUB 2900
1410 C=2
1420 R=20
1430 CALL HCHAR(20,23,32,9)
1440 X$="DEDUCTION FOR 19"&STR$(CYR)&" $"&X$
1450 GOSUB 2840
1460 DEDUCT=0
1470 N=0
1480 CMO=1
1490 CYR=CYR+1
1500 GOTO 1190
1510 REM  DISPLAY STATUS ***
1520 PRINT ::"CONTRACT STATUS"::
1530 INPUT "ENTER DATE: ":SDATE$
1540 CALL HCHAR(23,1,32,32)
1550 PRINT "STATUS ON ";SDATE$;":";:
1560 N=POS(SDATE$,"/",1)
1570 SMO$=SEG$(SDATE$,1,N-1)
1580 M=POS(SDATE$,"/",N+1)
1590 L=M-N-1
1600 SDA$=SEG$(SDATE$,N+1,L)
1610 SYR$=SEG$(SDATE$,M+1,2)
1620 SMO=VAL(SMO$)
1630 SDA=VAL(SDA$)
1640 SYR=VAL(SYR$)
1650 TMO=MO+NP-INT((MO+NP)/12)*12
1660 TMO=TMO+(TMO=0)*-12
1670 TYR=YR+INT(NP/12)-(TMO<MO)
1680 IF SYR>TYR THEN 2200
1690 IF SYR<YR THEN 2230
1700 IF (SYR=TYR)+(SMO>=TMO)=-2 THEN 2200
1710 IF (SYR=YR)+(SMO<MO)=-2 THEN 2230
1720 N=SYR-YR
1730 N=N*12
1740 X=SMO-MO
1750 N=N+X
1760 IF SDA<DA THEN 1780
1770 N=N+1
1780 TOTPD=N*PMNT
1790 X$=STR$(TOTPD)
1800 GOSUB 2900
1810 PRINT "TOTAL PAID      $";TAB(INC);X$
1820 BAL=UB+FC
1830 BAL=BAL-TOTPD
1840 X$=STR$(BAL)
1850 GOSUB 2900
1860 PRINT "BALANCE         $";TAB(INC);X$
1870 P=NP-N-1
1880 IF P=1 THEN 2160
1890 Q=P+1
1900 SAV=RND2(RUL78(AFC))
1910 IF SAV<1 THEN 2180
1920 PREPAY=BAL-SAV
1930 X$=STR$(PREPAY)
1940 GOSUB 2900
1950 PRINT "PREPAY AMOUNT   $";TAB(INC);X$
1960 X$=STR$(SAV)
1970 GOSUB 2900
1980 PRINT "SAVE BY PREPAY $ ";TAB(INC);X$
1990 DEDUCT=FC-SAV
2000 IF SYR=YR THEN 2100
2010 F=13-MO
2020 AYR=YR+1
2030 IF SYR=AYR THEN 2070
2040 F=F+12
2050 AYR=AYR+1
2060 GOTO 2030
2070 P=F
2080 Q=NP+(NP-F+1)
2090 DEDUCT=DEDUCT-RND2(RUL78(FC))
2100 X$=STR$(DEDUCT)
2110 GOSUB 2900
2120 PRINT "DEDUCTIBLE IN ";SYR$;" $";TAB(INC);X$
2130 PRINT "IF PAID OFF ON ";SDATE$;::
2140 GOSUB 2630
2150 RETURN
2160 SAV=RND2(1/DEN*AFC)
2170 GOTO 1920
2180 SAV=0
2190 GOTO 1920
2200 PRINT "PAID UP";:
2210 GOSUB 2630
2220 RETURN
2230 PRINT "TOO EARLY"::
2240 GOTO 2210
2250 REM  TAX DEDUCTION ****
2260 PRINT ::"IF YOU PAY ALL PAYMENTS"
2270 PRINT "AS SCHEDULED, YOU MAY"
2280 PRINT "DEDUCT FINANCE CHARGE"
2290 PRINT "AS FOLLOWS:"::
2300 PRINT "YEAR                    AMOUNT"::
2310 ND=NP
2320 DYR=YR
2330 P=13-MO
2340 IF P=1 THEN 2470
2350 P=(P<=ND)*-P+(P>ND)*-ND
2360 N=ND-P+1
2370 Q=ND+N
2380 DED=RND2(RUL78(FC))
2390 X$=STR$(DED)
2400 GOSUB 2900
2410 PRINT "19";STR$(DYR);"          $";TAB(INC);X$
2420 ND=ND-P
2430 DYR=DYR+1
2440 IF ND<12 THEN 2490
2450 P=12
2460 GOTO 2360
2470 DED=RND2(NP/DEN*FC)
2480 GOTO 2390
2490 IF ND=0 THEN 2570
2500 IF ND=1 THEN 2600
2510 P=ND
2520 Q=ND+1
2530 DED=RND2(RUL78(FC))
2540 X$=STR$(DED)
2550 GOSUB 2900
2560 PRINT "19";STR$(DYR);"          $";TAB(INC);X$
2570 PRINT ::
2580 GOSUB 2630
2590 RETURN
2600 DED=RND2(1/DEN*FC)
2610 GOTO 2540
2620 REM  NEW SCREEN *******
2630 CALL GCHAR(23,25,X)
2640 IF X<>32 THEN 2720
2650 C=2
2660 R=23
2670 X$="PRESS ENTER TO CONTINUE"
2680 GOSUB 2840
2690 R=24
2700 X$="OR 9 TO QUIT"
2710 GOSUB 2840
2720 C=16
2730 GOSUB 2780
2740 IF SEL=57 THEN 2760
2750 RETURN
2760 STOP
2770 REM  INPUT SUBROUTINE *
2780 CALL HCHAR(24,C,30)
2790 CALL KEY(0,SEL,STAT)
2800 IF STAT=0 THEN 2790
2810 CALL HCHAR(24,C,32)
2820 RETURN
2830 REM  PRINT SUBROUTINE *
2840 FOR J=1 TO LEN(X$)
2850 X=ASC(SEG$(X$,J,1))
2860 CALL HCHAR(R,C+J,X)
2870 NEXT J
2880 RETURN
2890 REM  EDIT ROUTINE *****
2900 Q=POS(X$,".",1)
2910 IF Q=0 THEN 2950
2920 IF Q=LEN(X$)-1 THEN 2970
2930 IF LEN(X$)>6 THEN 2990
2940 RETURN
2950 X$=X$&".00"
2960 GOTO 2930
2970 X$=X$&"0"
2980 GOTO 2930
2990 A=LEN(X$)-6
3000 Y$=SEG$(X$,1,A)
3010 Z$=SEG$(X$,A+1,6)
3020 X$=Y$&","&Z$
3030 RETURN
3040 END
```

3rd-Party NEWS

For Software Developers

The annual West Coast Computer Faire has been repeatedly successful in bringing together more microcomputer software developers in one place, at one time, than any other show or event. It is therefore not surprising that the message conveyed by the Texas Instruments exhibit at the 7th Faire (March 19-21 in San Francisco) fell largely on understanding ears: The Texas Giant was just now starting to play out its hand, and after a slow start, was finally exhibiting "the right stuff"—a winning combination of sophisticated-yet-adaptable user-friendly hardware, serious-yet-inexpensive software development tools, and the most portable of popular operating systems. All of this on top of unbelievably low hardware prices that are designed to attract not tens of thousands, but *hundreds of thousands* of eager consumers. It didn't take trumpets and dancing girls to get the rest of the message across to this dynamic group of enterpreneurs: TI was welcoming the many third-party developers for other popular machines to come on over and claim a piece of the action . . .

TI's booth with its strategic multi-level placement of video monitors attracted over 20,000 visitors during the 3-day event. Game players had the opportunity to experience the fast action and impressive graphics and sound effects of such arcade games as *TI Invaders, Munch-Man,* and *Tombstone City*. The Scott, Foresman CAI software, TI LOGO, and the TEXNET information utility captured the imagination of much of the attendant crowd. TI's new Peripheral Expansion Box (with its capacity to hold 7 peripheral cards plus double sided disk drives) was in active use demonstrating the UCSD Pascal IV.0 System while running *Personal Tax Plan*, the Pascal-written income tax package developed by Aardvark Software. Interested visitors also had a chance to see the new stand-alone Editor/Assembler, the *Mini Memory* Command Module (packed with battery-backed CMOS RAM and powerful Assembly Language macros), and a graphic demonstration of the high-resolution color capabilities of the new enhanced Video Display Processor.

San Francisco Tourist . . . from p. 49

```
1310 CALL CHAR(96,"1C1C087F08142222")
1320 CALL CHAR(97,"10107C101")
1330 CALL COLOR(9,7,1)
1340 CALL CHAR(104,"1038107C10FE101")
1350 CALL CHAR(105,"FF81A59999A581FF")
1360 CALL COLOR(10,13,1)
1370 PRINT ::"MUIR WOODS IS A BEAUTIFUL";
     :"FOREST OF GIANT TREES"
1380 PRINT :"NORTH OF SAN FRANCISCO.";
     :"TAKE A TOUR AND MARK AS";
     :"MANY TREES ON YOUR MAP AS"
1390 PRINT :"YOU CAN.  MOVE BY PRESSING";
     :"THE ARROW KEYS; MARK TREES";
     :"BY PRESSING <ENTER>.":::::
1400 GOSUB 470
1410 CALL CLEAR
1420 CALL SCREEN(12)
1430 PRINT "TIME"::"TREES"
1440 RANDOMIZE
1450 FOR I=1 TO 70
1460 CALL HCHAR(R19,R28,104)
1470 NEXT I
1480 SH=0
1490 P=0
1500 X=2
1510 Y=2
1520 S=32
1530 CALL HCHAR(2,2,96)
1540 CALL SOUND(150,1397,4)
1550 CALL KEY(0,K,S)
1560 IF K=13 THEN 1930
1570 IF K<>69 THEN 1610
1580 DX=-1
1590 DY=0
1600 GOTO 1720
1610 IF K<>68 THEN 1650
1620 DX=0
1630 DY=1
1640 GOTO 1720
1650 IF K<>88 THEN 1690
1660 DX=1
1670 DY=0
1680 GOTO 1720
1690 IF K<>83 THEN 1720
1700 DX=0
1710 DY=-1
1720 IF G>=104 THEN 1740
1730 G=97
1740 CALL HCHAR(X,Y,G)
1750 X=X+DX
1760 IF X>1 THEN 1780
1770 X=1
1780 IF X<20 THEN 1800
1790 X=20
1800 Y=Y+DY
1810 IF Y>2 THEN 1830
1820 Y=2
1830 IF Y<31 THEN 1850
1840 Y=31
1850 CALL GCHAR(X,Y,G)
1860 CALL HCHAR(X,Y,96)
1870 SH=SH+1
1880 FOR II=1 TO LEN(STR$(SH))
1890 CALL HCHAR(21,II+9,ASC(SEG$(STR$(SH),II,1)))
1900 NEXT II
1910 IF SH=100 THEN 2020
1920 GOTO 1550
1930 CALL SOUND(100,-2,0)
1940 IF G<>104 THEN 1870
1950 CALL HCHAR(X,Y,105)
1960 G=105
1970 P=P+1
1980 FOR II=1 TO LEN(STR$(P))
1990 CALL HCHAR(23,II+9,ASC(SEG$(STR$(P),II,1)))
2000 NEXT II
2010 GOTO 1870
2020 PRINT :"TIME IS UP.";
     :"YOU LOOKED AT";P;"TREES."
2030 GOSUB 510
2040 PRINT ::"PRESS ANY KEY"
2050 CALL KEY(0,K,S)
2060 IF S<1 THEN 2050
2070 CALL CLEAR
2080 CALL COLOR(9,2,1)
2090 CALL COLOR(10,2,1)
2100 GOTO 250
2110 CALL CHAR(33,"0101010101010001")
2120 PRINT "HAVE FUN IN SAN FRANCISCO!":::
     "SEE YOU AT TI-FEST!!":::::
2130 END
```

Overland Flow . . . from p. 31

```
1520 PRINT "     HYDROGRAPH PRINTING"::::::::::
1530 FOR I=1 TO HY
1540 PRINT #FILE:TAB(9);"HYDROGRAPH #";STR$(I):
1550 IF FILE=0 THEN 1570
1560 PRINT #FILE:
1570 PRINT #FILE;"  TIME(MIN)","DISCHARGE(CFS)";
1580 IF FILE=0 THEN 1600
1590 PRINT #FILE;"  ---------","-------------";
1600 FOR J=1 TO 22
1610 IF J=11 THEN 1710
1620 IF H(I,2,J)=0 THEN 1710
1630 IL=4
1640 N=H(I,2,J)
1650 FL=1
1660 GOSUB 3210
1670 N=H(1,1,J)
1680 FL=3
1690 GOSUB 3210
1700 PRINT #FILE:TAB(4);E$;TAB(16);D$
1710 NEXT J
1720 IF FILE<>0 THEN 1770
1730 PRINT " PRESS ANY KEY TO CONTINUE";
1740 GOSUB 4230
1750 CALL CLEAR
1760 GO TO 1780
1770 PRINT #FILE:::::
1780 NEXT I
1790 IF TST=1 THEN 1810
1800 GO TO 2030
1810 CALL CLEAR
1820 GOSUB 3720
1830 CALL SCREEN(15)
1840 PRINT " PRESS";TAB(9);"TO";" -----";TAB(9);
     "--";"  1     DISPLAY DATA(GIVEN";TAB(10);
     "AND CALCULATED)"::
1850 PRINT "  2     DISPLAY HYDROGRAPH"::
1860 PRINT "  3     COMPUTE ANOTHER";TAB(10);
     "HYDROGRAPH AND";TAB(10);"COMPARE"::
1870 PRINT "  4     REDIRECT OUTPUT"::"  5     ENTER
     NEW PROBLEM"::"  6     END PROGRAM"::::
1880 CALL SOUND(200,600,1)
1890 GOSUB 3760
1900 GOSUB 4230
1910 IF (KEY<49)+(KEY>54)=-1 THEN 1920 ELSE 1940
1920 CALL SOUND(250,110,1)
1930 GO TO 1900
1940 CALL SOUND(100,666,1)
1950 CALL CLEAR
1960 CALL SCREEN(8)
1970 ON (KEY-48)GO TO 3470,3800,3650,2000,480,1980
1980 CALL CLEAR
1990 END
2000 GOSUB 4260
2010 GO TO 1810
2020 CALL SOUND(150,600,1)
2030 CALL CLEAR
2040 F2=0
2050 QMAX=QW(1)
2060 IF HY=1 THEN 2090
2070 IF QMAX>=QW(2)THEN 2090
2080 QMAX=QW(2)
2090 TMAX=H(1,2,22)
2100 IF HY=1 THEN 2230
2110 IF TMAX>=H(2,2,22)THEN 2130
2120 TMAX=H(2,2,22)
2130 CALL CLEAR
2140 IF QMAX>=10 THEN 2150 ELSE 2160
2150 QMAX=RD(QMAX)
2160 PRINT #F2:TAB(12);"LEGEND":TAB(12);"-------";
     " X = HYDROGRAPH #1";:" 0 = HYDROGRAPH #2";:
2170 PRINT #F2:TAB(2);CHR$(64);" = COINCIDENCE OF
     1 & 2";:" MAX DISCHARGE(CFS)=";RD2(QMAX)::::::::
2180 IF (FILE=0)+(F2>0)<0 THEN 2210
2190 F2=FILE
2200 GOTO 2160
2210 FOR I=1 TO 700
2220 NEXT I
2230 CALL CLEAR
2240 CALL SCREEN(8)
2250 FOR I=9 TO 16
2260 CALL COLOR(I-8,2,1)
2270 CALL COLOR(I,2,16)
2280 NEXT I
2290 CALL SOUND(150,600,1)
2300 FOR I=1 TO 20
2310 CALL HCHAR(I,9,145,20)
2320 NEXT I
2330 CALL VCHAR(1,9,101)
2340 CALL VCHAR(2,9,100,4)
2350 CALL VCHAR(6,9,101)
2360 CALL VCHAR(7,9,100,4)
2370 CALL VCHAR(11,9,101)
2380 CALL VCHAR(12,9,100,4)
2390 CALL VCHAR(16,9,101)
2400 CALL VCHAR(17,9,100,4)
2410 CALL VCHAR(20,9,99)
2420 CALL HCHAR(20,10,102,3)
2430 CALL HCHAR(20,13,103)
2440 CALL HCHAR(20,14,102,4)
2450 CALL HCHAR(20,18,103)
2460 CALL HCHAR(20,19,102,4)
2470 CALL HCHAR(20,23,103)
2480 CALL HCHAR(20,24,102,4)
2490 CALL HCHAR(20,28,103)
2500 GOSUB 3930
2510 FOR J=1 TO HY
2520 FOR I=1 TO 22
2530 IF (I=11)+(I=12)+(KNT=0)<>-2 THEN 2570
2540 P1=TPP(J,I-10)/TMAX*19.99999
2550 P2=QW(J)/QMAX*19.99999
2560 GOSUB 2930
2570 P1=H(J,2,I)/TMAX*19.99999
2580 P2=H(J,1,I)/QMAX*19.99999
2590 GOSUB 2930
2600 NEXT I
2610 NEXT J
2620 IF FILE=0 THEN 2870
2630 NUS$="          ***GRAPH PRINTING***"
2640 FOR I=1 TO LEN(NUS$)
2650 CALL VCHAR(24,I,ASC(SEG$(NUS$,I,1)))
2660 NEXT I
2670 FOR I=1 TO 23
2680 B$=""
2690 FOR J=1 TO 32
2700 CALL GCHAR(I,J,A)
2710 IF (J=9)+(J>20)=-2 THEN 2820
2720 IF (I=20)+(J<9)+(J>28)=-2 THEN 2820
2730 GOSUB 4200
2740 IF (A=91)+(A=93)<>-1 THEN 2760
2750 A=145
2760 IF A<>92 THEN 2780
2770 A=37
2780 IF (A<104)+(A>154)+(A=145)=-1 THEN 2820
2790 A=INT(A/10)+86
2800 IF A<99 THEN 2820
2810 A=98
2820 B$=B$&CHR$(A)
2830 NEXT J
2840 PRINT #FILE:B$
2850 NEXT I
2860 PRINT #FILE:::::
2870 ZZ$="   PRESS ANY KEY TO CONTINUE"
2880 FOR I=1 TO LEN(ZZ$)
2890 CALL VCHAR(24,I,ASC(SEG$(ZZ$,I,1)))
2900 NEXT I
2910 GOSUB 4230
2920 GOTO 1810
2930 T1=P1-INT(P1)
2940 T2=P2-INT(P2)
2950 Y=20-INT(P2)
2960 X=9+INT(P1)
2970 IF Y=20 THEN 3200
2980 IF X=9 THEN 3200
2990 IF T1<.5 THEN 3030
3000 IF T2<.5 THEN 3060
3010 PS=2
3020 GOTO 3090
3030 IF T2<.5 THEN 3080
3040 PS=1
3050 GOTO 3090
3060 PS=4
3070 GOTO 3090
3080 PS=3
3090 IF J=2 THEN 3120
3100 CHAR=PS+110
3110 GOTO 3180
3120 CALL GCHAR(Y,X,CHAR2)
3130 IF CHAR2=145 THEN 3170
3140 IF (CHAR2<108)+(CHAR2>103)=-2 THEN 3200
3150 CHAR=CHAR2*10-990+PS
3160 GOTO 3180
3170 CHAR=103+PS
3180 CALL VCHAR(Y,X,CHAR)
3190 CALL SOUND(100,220*J,3)
3200 RETURN
3210 N2=10^FL
3220 N1=ABS(N)+.5/N2
3230 IP=INT(N1)
3240 FP=INT(N2*(N1-IP))+N2
3250 D$=STR$(IP)
3260 L=LEN(D$)
3270 IF (IL=0)+(IP=0)<>-2 THEN 3300
3280 D$=""
3290 L=0
3300 IF IL<L THEN 3400
3310 IF IL<=L THEN 3350
3320 D$=" "&D$
3330 L=L+1
3340 GOTO 3310
3350 IF FL<=0 THEN 3390
3360 D$=D$&","&SEG$(STR$(FP),2,FL)
3370 IF FL=3 THEN 3390
3380 E$=D$
3390 RETURN
3400 D$="*"
3410 FOR I=1 TO (IL+FL)
3420 D$=D$&"*"
3430 NEXT I
3440 IF FL=3 THEN 3460
3450 E$=D$
3460 RETURN
3470 CALL CLEAR
3480 IF FILE=0 THEN 3500
3490 PRINT "        DATA PRINTING"::::::::::::
3500 FOR I=1 TO HY
3510 PRINT #FILE:TAB(7);"DATA--HYDROGRAPH #";
     STR$(I)::
3520 PRINT #FILE:"GIVEN:";"------":
     " INTENSITY(IN/HR)= ";IN(I)::
     " DURATION(MIN)=   ";DU(I)::
3530 PRINT #FILE:" LENGTH(FT)=       ";LE(I)::
     " WIDTH(FT)=        ";WI(I)::" SLOPE
     (FT/FT)=     ";SL(I)::
3540 PRINT #FILE:" ROUGHNESS FACTOR= ";CR(I)::
3550 PRINT #FILE:"CALCULATED:";"-----------":
     " EQUILIB. TIME(MIN)= ";RD(TE(I))::
3560 PRINT #FILE:" MAX DISCHARGE(CFS)= ";
     RD2(QW(I))::
3570 IF FILE<>0 THEN 3620
3580 PRINT " PRESS ANY KEY TO CONTINUE":
3590 GOSUB 4230
3600 CALL CLEAR
3610 GO TO 3630
3620 PRINT #FILE:::::
3630 NEXT I
3640 GO TO 1810
3650 HY=2
3660 FOR I=1 TO 2
3670 FOR J=1 TO 22
3680 H(2,I,J)=0.0
3690 NEXT J
3700 NEXT I
3710 GO TO 540
3720 FOR I=1 TO 8
3730 CALL COLOR(1,1,1)
3740 NEXT I
3750 RETURN
3760 FOR I=1 TO 8
3770 CALL COLOR(1,2,1)
3780 NEXT I
3790 RETURN
3800 CALL CLEAR
3810 GOSUB 3720
3820 PRINT TAB(6);"HYDROGRAPH DISPLAY":TAB(6);
     "-------------------":::TAB(7);
     "PRESS   FOR"::TAB(9);"1     TABLE":
3830 PRINT :TAB(9);"2     GRAPH"::TAB(9);
     "3     BOTH"::::::
3840 CALL SOUND(200,600,1)
3850 GOSUB 3760
3860 GOSUB 4230
3870 IF (KEY<49)+(KEY>51)=-1
     THEN 3880 ELSE 3900
3880 CALL SOUND(250,110,1)
3890 GOTO 3860
3900 CALL SOUND(150,666,1)
3910 TST=KEY-48
3920 ON TST GOTO 1500,2030,1500
3930 TI3$="100 75 50 25  0"
3940 FOR I=1 TO 5
3950 FOR J=1 TO 3
3960 CALL VCHAR(5*I-4,J+5,ASC(SEG$(TI3$,3*I+J-3,1)))
3970 NEXT J
3980 NEXT I
3990 SC=1
4000 IF TMAX<=132.666 THEN 4020
4010 SC=10
4020 FOR I=1 TO 4
4030 TM$=STR$(I*TMAX/(4*SC)+.5)
4040 CALL HCHAR(21,I*5+8,ASC(SEG$(TM$,1,1)))
4050 IF SEG$(TM$,2,1)="." THEN 4070
4060 CALL HCHAR(21,I*5+9,ASC(SEG$(TM$,2,1)))
4070 IF (SC=1)+(I=4)+(TMAX>99.49999999)<>-3
     THEN 4090
4080 CALL HCHAR(21,30,ASC(SEG$(TM$,3,1)))
4090 NEXT I
4100 TI2$=CHR$(91)&CHR$(92)&CHR$(93)&
     "OF MAX DISCHARGE"
4110 FOR I=1 TO LEN(TI2$)
4120 CALL VCHAR(I+1,5,ASC(SEG$(TI2$,I,1)))
4130 NEXT I
4140 TI4$="  TIME(MIN)"
4150 IF SC=1 THEN 4170
4160 TI4$="TIME(MIN) X 10"
4170 FOR I=1 TO LEN(TI4$)
4180 CALL VCHAR(23,I+11,ASC(SEG$(TI4$,I,1)))
4190 NEXT I
4200 IF (I<>20)+(J<>9)=-2 THEN 4220
4210 A=A+56
4220 RETURN
4230 CALL KEY(0,KEY,ST)
4240 IF ST<=0 THEN 4230
4250 RETURN
4260 CALL CLEAR
4270 GOSUB 3720
4280 PRINT TAB(4);"OUTPUT DESTINATION":TAB(4);
     "-------------------"::
4290 PRINT "   PRESS   FOR":;"     1     SCREEN"::
     "     2     THERMAL PRINTER"::::::
4300 GOSUB 3760
4310 GOSUB 4230
4320 IF (KEY<49)+(KEY>50)=-1 THEN 4330 ELSE 4350
4330 CALL SOUND(250,110,1)
4340 GOTO 4310
4350 CALL SOUND(100,666,1)
4360 FILE=0
4370 IF KEY=49 THEN 4460
4380 FILE=1
4390 DVC$="TP.U.S"
4400 IF KEY=50 THEN 4420
4410 DVC$="RS232"
4420 IF DFG$="" THEN 4440
4430 CLOSE #1
4440 OPEN #1:DVC$,OUTPUT
4450 DFG$="1"
4460 RETURN
4470 PRINT :
4480 CALL SOUND(300,110,1)
4490 PRINT " NUMBER OUTSIDE NORMAL RANGE":
4500 PRINT "  ENTER";
4510 RETURN
```

99'er

The author gives you the option of output to the screen or to the thermal printer. If you have another printer, change Line 4290 to print PRINTER instead of THERMAL PRINTER. Then change Line 4390 to your printer configuration.

As it is,

4390 DVC$="TP.U.S"

For example, change to

4390 DVC$="RS232.TW.BA=110"

Depending on your printer's graphic capabilities, you may also need to change Lines 2000-3200 for the graphic plot.

DEALER DIRECTORY

Granada Hills, 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., Arrada, CO 80004, (303) 421-6361.**

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.**

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., 7216 Boone Ave. North, Brooklyn Park, MN 55428. (612) 535-5330.**

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.**

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.**

Dayton, OH

The only full line TI distributor stocking Semiconductors, DSG (700-800 OMNI Products) TM990 Microcomputer Boards, Calculators, complete in depth inventory including 99/4 Products-Software. Ohio 1-800-762-9510. Surrounding area 1-800-543-9550. Will ship. Call us. **Esco Inc., P. O. Box 1166, 221 Crane St. Dayton, OH 45403, (513) 226-1133**

Portland, OR

Friendly pre-programmed computing systems for the office and home. Featuring T. I. 99/4A's, Hewlett Packard, Xerox, IBM, and most printers. With T. I. 99, discounted packages of all configurations. 24 Hour Home Computer Hotline. **The Electronic Cottage, 2336 SW Osage, Suite 407, Portland, OR 97205, (503) 224-9282.**

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.**

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 99/X to S-100 bus card available system customization. Hard disk, tape. Any AMP connectors. **Interface Technology, P O Box 745, College Park, MD 20740, (301) 490-3608.**

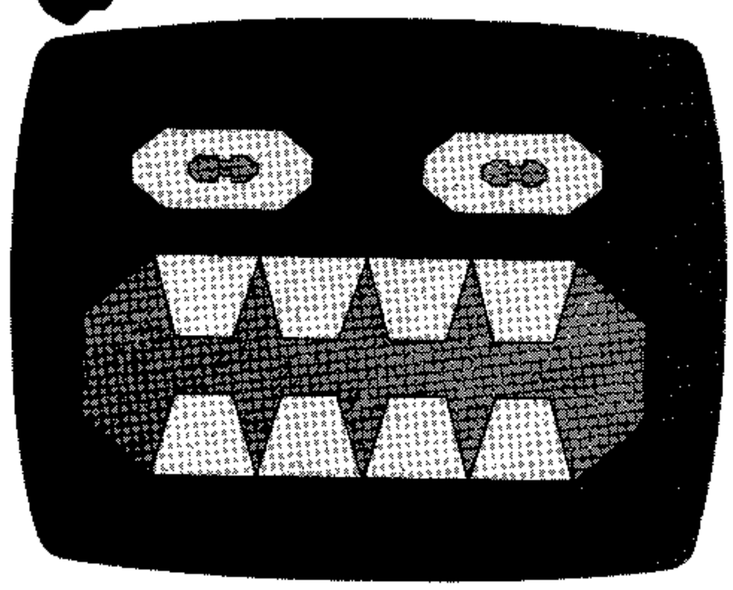
RATES

Listings here are $30 per issue; minimum insertion, 3 issues (6 months). Prepayment of $90 required. Ads include 35 words describing products and services, plus company name, address and phone. (No merchandise prices, please.) Call Pat at 503-485-8796 or write 99'er Magazine, Ad Department, 2715 Terrace View Drive, Eugene, OR 97405.

See page 92

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 February 1 and May 15, 1982, we'll send you the exciting new "Munchman Arcade Game" absolutely free. The modules you 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

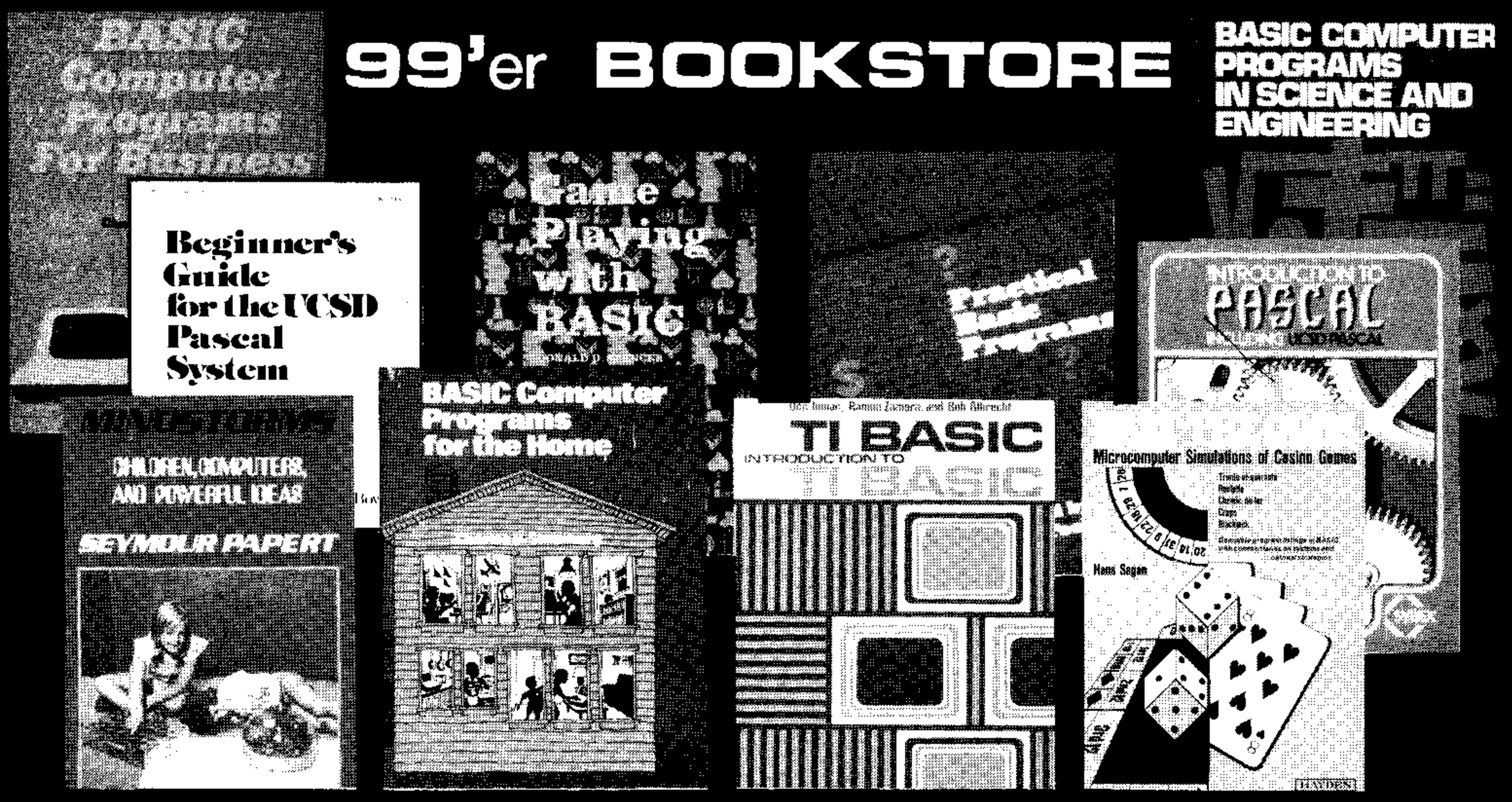
Serial # of TI 99/4 Console Age of Buyer Occupation

No. of children Ages Name of store where purchased

Intended Use Home Personal Finance Education

99'er BOOKSTORE

BASIC COMPUTER PROGRAMS IN SCIENCE AND ENGINEERING



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, **$15.95**
1981, 440 pages, 7 x 9

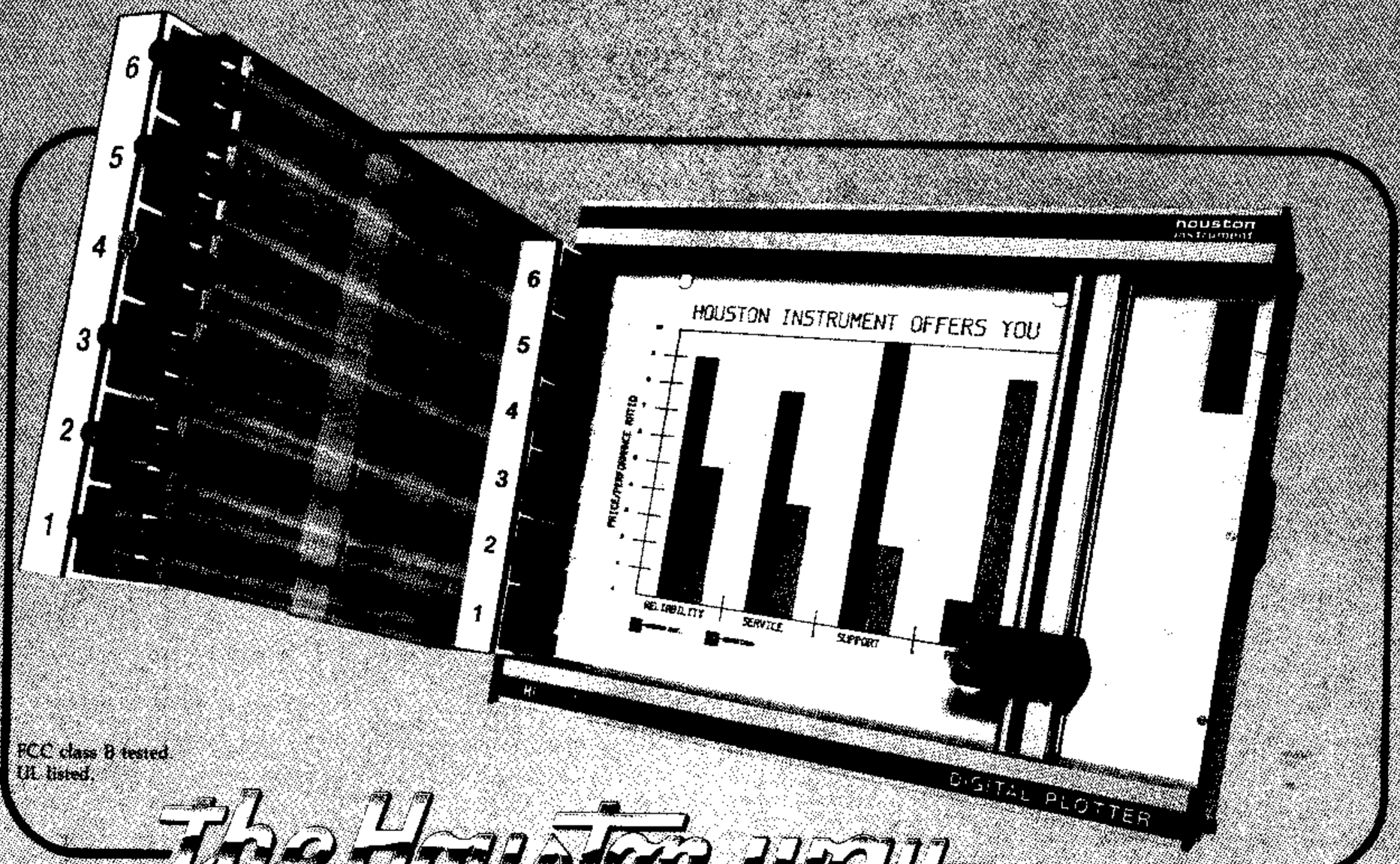
BEAT THE ODDS: MICRO-COMPUTER SIMULATIONS OF CASINO GAMES

By Hans Sagan.
Here's an extremely useful programming guide that provides realistic simulations of five popular Casino games: Trente-et-Quarante (Thirty and Forty), Roulette, Chemin-de-Fer, Craps, and Blackjack. Each of the five chapters has the same structure. It begins with a computer run, displaying facets of the programs, followed by an explanation of the objectives and the physical execution of the game. Acceptable bets and how to place them are discussed and systems and/or strategies laid out. Finally, the computer program is developed and various modifications of the program are detailed.

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.

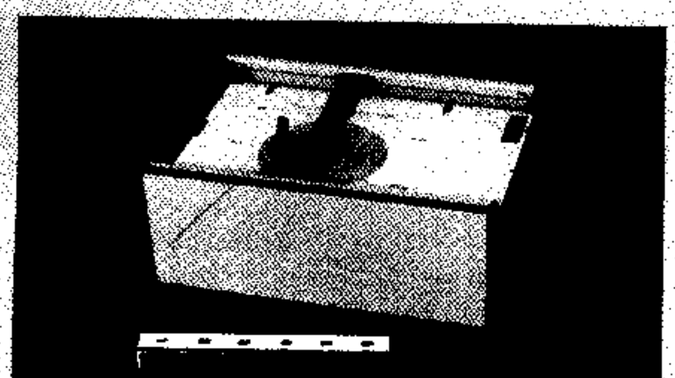
New from HIPLØT™ multi-pen plotting for as little as $1480*.



FCC class B tested.
UL listed.

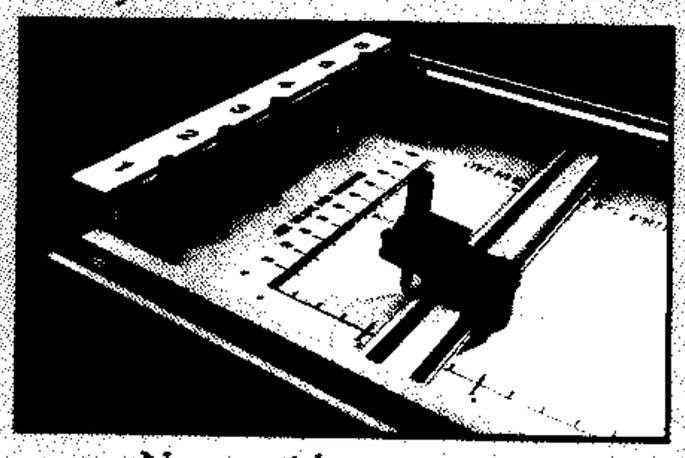
the Houston way

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.

Tex-Thello . . . from p. 67

```
1050 TYPE=3
1060 A(X,Y)=4
1070 GOSUB 2330
1080 NEXT Y
1090 NEXT X
1100 FOR I=3 TO 6
1110 FOR J=3 TO 6
1120 IF A(I,J)<>1 THEN 1140
1130 A(I,J)=0
1140 NEXT J
1150 NEXT I
1160 PL=1
1170 TURN=5
1180 REM  BEGIN MAIN LOOP
1190 FOR I=1 TO 10
1200 CALL HCHAR(23,I+7,PLAY(PL,I))
1210 NEXT I
1220 CALL VCHAR(23,23,120)
1230 CALL VCHAR(23,28,120)
1240 IF COMPLAY=PL THEN 1340
1250 CALL KEY(0,RE,ST)
1260 IF ST<>1 THEN 1250
1270 X=RE-48
1280 CALL VCHAR(23,23,RE)
1290 CALL KEY(0,RE,ST)
1300 IF ST<>1 THEN 1290
1310 Y=RE-48
1320 CALL VCHAR(23,28,RE)
1330 GO TO 1370
1340 GOSUB 2960
1350 CALL VCHAR(23,23,48+X)
1360 CALL VCHAR(23,28,48+Y)
1370 IF X>8 THEN 1460
1380 IF X<1 THEN 1460
1390 IF Y>8 THEN 1460
1400 IF Y<1 THEN 1460
1410 IF A(X,Y)>1 THEN 1460
1420 GOSUB 2530
1430 IF EXTRA>0 THEN 1490
1440 GOSUB 2120
1450 IF SW=0 THEN 1640
1460 CALL SOUND(500,200,3)
1470 IF COMPLAY=PL THEN 1640
1480 GOTO 1220
1490 A(X,Y)=PL+2
1500 FOR I=X-1 TO X+1
1510 FOR J=Y-1 TO Y+1
1520 IF A(I,J)<>1 THEN 1540
1530 A(I,J)=0
1540 NEXT J
1550 NEXT I
1560 TURN=TURN+1
1570 TYPE=2
1580 IF PL=1 THEN 1600
1590 TYPE=3
1600 GOSUB 2270
1610 GOSUB 2530
1620 GOSUB 2840
1630 IF TURN=65 THEN 1810
1640 IF PL=1 THEN 1670
1650 A1=4
1660 GOTO 1680
1670 A1=3
1680 FOR I=1 TO 8
1690 FOR J=1 TO 8
1700 IF A(I,J)<=1 THEN 1720
1710 IF A(I,J)<>A1 THEN 1750
1720 NEXT J
1730 NEXT I
1740 GOTO 1810
1750 IF PL=1 THEN 1780
1760 PL=1
1770 GOTO 1190
1780 PL=2
1790 GOTO 1190
1800 REM  END MAIN LOOP AND
     BEGIN TOTALS REGION
1810 TOT1=0
1820 TOT2=0
1830 FOR I=1 TO 8
1840 FOR J=1 TO 8
1850 IF A(I,J)=3 THEN 1890
1860 IF A(I,J)<>4 THEN 1900
1870 TOT2=TOT2+1
1880 GO TO 1900
1890 TOT1=TOT1+1
1900 NEXT J
1910 NEXT I
1920 IF TOT2<>TOT1 THEN 1950
1930 PRINT "DRAW     32 TO 32"
1940 GO TO 2050
1950 PRINT
1960 PRINT "AT A SCORE OF";
     TOT1;" TO ";TOT2;":"
1970 PRINT "           WINS"
1980 IF TOT1>TOT2 THEN 2010
1990 PWIN=2
2000 GO TO 2020
2010 PWIN=1
2020 FOR I=1 TO 10
2030 CALL HCHAR(23,I+1,
     PLAY(PWIN,I))
2040 NEXT I
2050 PRINT
     :"PLAY AGAIN? (Y/N)";
2060 CALL KEY(0,K,S)
2070 IF K=89 THEN 180
2080 IF K<>78 THEN 2060
2090 STOP
2100 REM
2110 REM  IS THERE A
     LEGAL MOVE?
2120 SW=0
2130 ZZX=X
2140 ZZY=Y
2150 FOR X=1 TO 8
2160 FOR Y=1 TO 8
2170 IF A(X,Y)<>0 THEN 2210
2180 GOSUB 2530
2190 IF EXTRA=0 THEN 2210
2200 SW=1
2210 NEXT Y
2220 NEXT X
2230 X=ZZX
2240 Y=ZZY
2250 RETURN
2260 REM  COLOR ONTO
     BOARD ROUTINE
2270 IF TYPE<>1 THEN 2330
2280 S1=96
2290 S2=97
2300 S3=98
2310 S4=99
2320 GO TO 2430
2330 IF TYPE<>2 THEN 2390
2340 S1=104
2350 S2=105
2360 S3=106
2370 S4=107
2380 GO TO 2430
2390 S1=112
2400 S2=113
2410 S3=114
2420 S4=115
2430 X1=7+2*X
2440 X2=X1+1
2450 Y1=3+2*Y
2460 Y2=Y1+1
2470 CALL HCHAR(Y1,X1,S1)
2480 CALL HCHAR(Y1,X2,S2)
2490 CALL HCHAR(Y2,X1,S3)
2500 CALL HCHAR(Y2,X2,S4)
2510 RETURN
2520 REM  EXTRA SQUARES
2530 EXTRA=1
2540 FOR I=1 TO 3
2550 FOR J=1 TO 3
2560 U=X+DIR(I)
2570 V=Y+DIR(J)
2580 IF U=X THEN 2590
     ELSE 2600
2590 IF V=Y THEN 2790
2600 IF PL=1 THEN 2620
2610 IF A(U,V)=3 THEN 2630
     ELSE 2790
2620 IF A(U,V)=4 THEN 2630
     ELSE 2790
2630 U=U+DIR(I)
2640 V=V+DIR(J)
2650 IF A(U,V)<=1 THEN 2790
2660 IF A(U,V)=2 THEN 2790
2670 IF PL=1 THEN 2690
2680 IF A(U,V)=4 THEN 2700
     ELSE 2630
2690 IF A(U,V)=3 THEN 2700
     ELSE 2630
2700 U=X+DIR(I)
2710 V=Y+DIR(J)
2720 IF A(U,V)=PL+2 THEN 2790
2730 EX(EXTRA)=U
2740 EY(EXTRA)=V
2750 EXTRA=EXTRA+1
2760 U=U+DIR(I)
2770 V=V+DIR(J)
2780 GO TO 2720
2790 NEXT J
2800 NEXT I
2810 EXTRA=EXTRA-1
2820 RETURN
2830 REM  COLOR ADDITION
     SQUARES
2840 XX=X
2850 YY=Y
2860 FOR K=1 TO EXTRA
2870 X=EX(K)
2880 Y=EY(K)
2890 GOSUB 2270
2900 A(X,Y)=PL+2
2910 NEXT K
2920 X=XX
2930 Y=YY
2940 RETURN
2950 REM  GET COMPUTER MOVE
2960 EXTRAP=-100
2970 FOR X=1 TO 8
2980 FOR Y=1 TO 8
2990 IF A(X,Y)<>0 THEN 4110
3000 GOSUB 2530
3010 IF EXTRA=0 THEN 4110
3020 IF TURN<57 THEN 3040
3030 HARD=1
3040 IF HARD=1 THEN 4070
3050 IF X=1 THEN 3380
3060 IF X=8 THEN 3380
3070 IF Y=1 THEN 3260
3080 IF Y=8 THEN 3260
3090 IF HARD=2 THEN 4070
3100 IF X=2 THEN 3180
3110 IF X=7 THEN 3180
3120 IF Y=2 THEN 4070
3130 IF Y=7 THEN 4070
3140 IF X*X+Y*Y+36<>9*(X+Y)
     THEN 3160
3150 EXTRA=EXTRA+1
3160 EXTRA=EXTRA+4
3170 GO TO 4070
3180 IF Y=2 THEN 3210
3190 IF Y=7 THEN 3210
3200 GO TO 4070
3210 U=(7*X-9)/5
3220 V=(7*Y-9)/5
3230 IF A(U,V)>1 THEN 4070
3240 EXTRA=EXTRA-12
3250 GO TO 4070
3260 IF HARD=2 THEN 3360
3270 FOR I=1 TO 8
3280 EDGE(I)=A(I,Y)
3290 NEXT I
3300 FOR I=1 TO EXTRA
3310 IF EY(I)<>Y THEN 3330
3320 EDGE(EX(I))=PL+2
3330 NEXT I
3340 Z=X
3350 GO TO 3570
3360 EXTRA=EXTRA+3
3370 GO TO 4070
3380 IF Y=1 THEN 3520
3390 IF Y=8 THEN 3520
3400 IF HARD=2 THEN 3500
3410 FOR I=1 TO 8
3420 EDGE(I)=A(X,I)
3430 NEXT I
3440 FOR I=1 TO EXTRA
3450 IF EX(I)<>X THEN 3470
3460 EDGE(EY(I))=PL+2
3470 NEXT I
3480 Z=Y
3490 GO TO 3570
3500 EXTRA=EXTRA+3
3510 GO TO 4070
3520 IF HARD=2 THEN 3550
3530 EXTRA=EXTRA+14
3540 GO TO 4070
3550 EXTRA=EXTRA+6
3560 GO TO 4070
3570 CORN=0
3580 FOR I=1 TO EXTRA
3590 IF 2*(EX(I)-2)*
     (EX(I)-7)+(EY(I)-2)*
     (EY(I)-7)=0 THEN 3610
3600 GO TO 3660
3610 U=(7*EX(I)-9)/5
3620 V=(7*EY(I)-9)/5
3630 IF A(U,V)<=1 THEN 3650
3640 GO TO 3660
3650 CORN=1
3660 NEXT I
3670 TYPE=0
3680 IF EDGE(1)=PL+2 THEN 3730
3690 LT=EDGE(1)
3700 IF LT<>0 THEN 3740
3710 LT=1
3720 GO TO 3740
3730 LT=2
3740 FOR I=2 TO Z-1
3750 IF EDGE(I)=PL+2 THEN 3800
3760 IF EDGE(I)<=1 THEN 3790
3770 LT=EDGE(I)
3780 GO TO 3800
3790 LT=1
3800 NEXT I
3810 IF LT=2 THEN 3970
3820 IF EDGE(8)=PL+2 THEN 3870
3830 HT=EDGE(8)
3840 IF HT<>0 THEN 3880
3850 HT=1
3860 GO TO 3880
3870 HT=2
3880 FOR I=7 TO Z+1 STEP -1
3890 IF EDGE(I)=PL+2 THEN 3940
3900 IF EDGE(I)<=1 THEN 3930
3910 HT=EDGE(I)
3920 GO TO 3940
3930 HT=1
3940 NEXT I
3950 IF HT=2 THEN 3970
3960 IF HT<>LT THEN 4030
3970 TYPE=1
3980 IF CORN=0 THEN 4010
3990 EXTRA=EXTRA-8
4000 GO TO 4070
4010 EXTRA=EXTRA+8
4020 GO TO 4070
4030 IF CORN=0 THEN 4060
4040 EXTRA=EXTRA-12
4050 GO TO 4070
4060 EXTRA=EXTRA-4
4070 IF EXTRA>EXTRAP THEN 4080
     ELSE 4110
4080 EXTRAP=EXTRA
4090 ZX=X
4100 ZY=Y
4110 NEXT Y
4120 NEXT X
4130 IF EXTRAP=-100 THEN 4140
     ELSE 4210
4140 FOR X=1 TO 8
4150 FOR Y=1 TO 8
4160 IF A(X,Y)>1 THEN 4190
4170 ZX=X
4180 ZY=Y
4190 NEXT Y
4200 NEXT X
4210 X=ZX
4220 Y=ZY
4230 RETURN
4240 END
```

99'er

Force 1 . . .from p. 36

```
100 REM ************
110 REM * FORCE 1 *
120 REM ************
130 REM BY W.K. BALTHROP
140 REM 99'ER VERSION 1.5.1XB
150 REM
160 CALL CLEAR
170 GOSUB 1510
180 DISPLAY AT(2,10):"FORCE 1"
190 DISPLAY AT(4,3):"LEVEL OF DIFFICULTY:"
200 DISPLAY AT(6,5):"1. BEGINNER" :: DISPLAY AT(7,5):
    "2. NOVICE" :: DISPLAY AT(8,5):"3. INTERMEDIATE"
210 DISPLAY AT(9,5):"4. SEMI-PRO" :: DISPLAY AT(10,5):
    "5. PRO"
220 ACCEPT AT(11,5)VALIDATE(DIGIT)SIZE(1):L1 :: L=L1*4
230 RANDOMIZE
240 CALL CLEAR
250 FOR CO=1 TO 8 :: CALL COLOR(CO,16,1):: NEXT CO
260 COU=0 :: D=1 :: DIS=11000 :: IF SC>=25 THEN L=L*2
270 CALL CHAR(88,"0102040810204080"):: CALL CHAR(89,"8040201008040201")
280 CALL CHAR(90,"03070E1C3870E0C0"):: CALL CHAR(91,"C0E070381C0E0703")
290 CALL CHAR(92,"070F1F3E7CF8F0E0"):: CALL CHAR(93,"E0F0F87C3E1F0F07")
300 CALL CHAR(94,"03060C183060C080"):: CALL CHAR(95,"C0603018C0603018")
310 CALL COLOR(8,1,1):: CALL SCREEN(2)
320 CALL CHAR(96,"0101010101010101"):: CALL CHAR(97,"8080808080808080")
330 CALL CHAR(98,"00000000000000FF"):: CALL CHAR(99,"FF")
340 CALL COLOR(9,16,1)
350 CALL VCHAR(7,12,96,9):: CALL VCHAR(7,21,97,9)
360 CALL HCHAR(6,13,98,8):: CALL HCHAR(16,13,99,8)
370 CALL CHAR(33,"FF"):: CALL CHAR(34,"0101010101010101")
380 CALL VCHAR(12,15,33):: CALL VCHAR(12,18,33):: CALL VCHAR(10,16,34):: CALL
    VCHAR(13,16,34)
390 FOR COL=10 TO 12 :: CALL COLOR(COL,7,1):: NEXT COL
400 GOSUB 410 :: GOTO 500
410 CALL CHAR(104,"00000008"):: CALL CHAR(105,"00000018"):: CALL CHAR(106,"00000
    01C")
420 CALL CHAR(107,"0000003C"):: CALL CHAR(108,"0000183C"):: CALL CHAR(109,"00001
    C3E")
430 CALL CHAR(110,"00003C7E18"):: CALL CHAR(111,"00187EFF3C42")
440 CALL CHAR(112,"000C1E7FFF3F40"):: CALL CHAR(113,"00"):: CALL CHAR(114,"00000
080C00080"):: CALL CHAR(115,"00")
450 CALL CHAR(116,"000000061F7FFFFF"):: CALL CHAR(117,"3F2040"):: CALL CHAR(118,
    "0000000080E0F0F0"):: CALL CHAR(119,"C04020")
460 CALL CHAR(120,"0000000001073FFF"):: CALL CHAR(121,"FF1F1F306040"):: CALL
    CHAR(122,"00000000080E0FCFF")
470 CALL CHAR(123,"FFF8F80C0602")
480 CALL CHAR(124,"02604CD700309C01")
490 RETURN
500 CALL COLOR(12,7,1)
510 GOSUB 700 :: GOSUB 850 :: GOSUB 1390
520 CALL KEY(0,K,S)
530 CALL POSITION(#1,P01,P02)
540 IF K=13 THEN GOSUB 980
550 T=INT(RND*10):: IF T=4 THEN DEV=L/10-INT(RND*L/5):
    : DEU=L/10-INT(RND*L/5)ELSE DEV,DEU=0
560 IF K=69 THEN D1=D1+L/5 :: SA=SA+L/5
570 IF K=88 THEN D1=D1-L/5 :: SA=SA-L/5
580 IF K=83 THEN D2=D2+L/5 :: SB=SB+L/5
590 IF K=68 THEN D2=D2-L/5 :: SB=SB-L/5
600 DIS=DIS-(L*15):: D=11-INT(DIS/1000):: IF DIS<200 THEN GOSUB 1270 :: GOTO 620
610 IF D<9 THEN GOSUB 1230 ELSE ON D-8 GOSUB 1240,1250,1260
620 D1=D1+DEV*(D/11):: D2=D2+DEU*(D/11)
630 CALL MOTION(#1,D1,D2)
640 IF S=0 THEN 660
650 FOR SM=2 TO 15 :: CALL MOTION(#SM,SA,SB):: NEXT SM
660 CALL SOUND(-100,800,15)
670 TIME=TIME+1 :: IF TIME=1000 THEN 980
680 DISPLAY AT(1,3):"SCORE:";SC,"TIME:";TIME :: GOTO 520
690 CALL CHARSET
700 DISPLAY AT(24,2):CHR$(92);"                              ";CHR$(93)
710 DISPLAY AT(23,3):CHR$(92);"                             ";CHR$(93)
720 DISPLAY AT(22,4):CHR$(92);"                            ";CHR$(93)
730 DISPLAY AT(21,5):CHR$(92);"                           ";CHR$(93)
740 DISPLAY AT(20,6):"Z                    ";CHR$(91)
750 DISPLAY AT(19,7):"Z                   ";CHR$(91)
760 DISPLAY AT(18,8):"Z                  ";CHR$(91)
770 DISPLAY AT(17,9):"Z                 ";CHR$(91)
780 DISPLAY AT(16,10)SIZE(1):"^" :: DISPLAY AT(16,19)SIZE(1):"_"
790 DISPLAY AT(15,11)SIZE(8):"^       _"
800 DISPLAY AT(14,12)SIZE(6):"X     Y"
810 DISPLAY AT(13,13)SIZE(1):"X" :: DISPLAY AT(13,16)SIZE(1):"Y"
820 DISPLAY AT(12,14)SIZE(2):"XY"
830 CALL HCHAR(11,16,32,2)
840 RETURN
850 IF SP1=0 THEN D1=INT(L-(RND*L*2)):: D2=INT(L-(RND*L*2)):: CALL SPRITE(#1,
    104,7,INT(RND*256)+1,INT(RND*256)+1,D1/(11/D),D2/(11/D))
860 SP1=1 :: D=1 :: DIS=11000
870 L=L+1 :: RETURN
880 CALL COLOR(8,7,1):: CALL COLOR(8,1,1)
890 CALL COINC(#1,87,124,D,C1)
900 CALL SOUND(20,880,2,990,2,10000,30,-4,2)
910 IF C1=-1 THEN SP1=0 :: CALL DELSPRITE(#1):: GOTO 930
920 RETURN
930 SC=SC+1 :: FOR CS=1 TO 3 :: CALL SCREEN(7):: CALL SCREEN(2):: NEXT CS
940 CALL SOUND(500,110,2,-4,2):: CALL HCHAR(12,16,124,2):: CALL HCHAR(11,16,124,
2):: CALL SOUND(1000,110,2,220,2,330,2,-8,2)
950 CALL SOUND(1,44000,30):: GOSUB 820
960 SA=0 :: SB=0 :: D=1 :: DIS=11000 :: L=L+2 :: GOSUB 850
970 RETURN
980 CALL CLEAR :: CALL SOUND(1000,440,2,550,2,660,2):: CALL SOUND(2000,770,2,
    880,2,990,2)
990 CALL DELSPRITE(ALL)
```

EXPLANATION OF THE PROGRAM
Force 1

Line Nos.	
160-240	Display Levels of difficulty; accept answer.
250-490	Assign variables, color, and characters.
500-590	Read keyboard, branch to subroutine, or adjust variables.
600-610	Adjust distance to alien; branch to display new alien ship.
620-640	Randomly change motion of alien ship.
650	Change motion of stars.
660-680	Display score, time; check for out of time (time >=1000).
700-840	Display laser beams on screen.
850-870	Assign alien space craft to a new location.
880-920	Fire laser, check for hit.
930-970	Alien destroyed. Adjust score, re-initialize variables.
980-1070	Subroutine when hit by alien; branch for bonus, or branch to end-of-game messages.
1080-1190	End-of-game messages.
1200-1220	Check to play again.
1230-1270	Change alien shape.
1280	Check for alien to fire back.
1290-1320	Alien is at max size, and moves off screen faster.
1330-1380	Alien fires and hits your ship; sound effects.
1390-1500	Display star pattern.
1510-1610	Display title page.

Space Patrol . . . from p. 37

EXPLANATION OF THE PROGRAM
Space Patrol

Line Nos.	
150	Clears screen and makes it black.
160	Sets colors of letters and numbers to white.
170-300	Displays game title while defining graphics characters.
310-330	Clears screen; lets user choose joysticks or keyboard.
340-370	Clears screen; initializes energy and ships destroyed; randomizes; lets user choose high or low skill level; sets magnification 3 for high level, 4 for low level.
380-390	Sets colors for stars; randomly places 40 stars on screen.
400	Creates gun sight in center of screen.
410	Displays energy level and number of ships destroyed at bottom of screen.
420	Randomly selects number from 2 to 6; if the number is 5 or 6 branches to the satellite routine, otherwise a supply ship is defined.
430	Changes colors of stars so they will twinkle.
440-450	Randomly sets speed, direction, and location of supply ship.
460	Branches to joystick or keyboard input to move gun sight.
470-510	Checks to see if fire button or key is pressed; if so, stops motion of gun sight, makes laser sound, checks for a hit, and shows a red laser blast; decreases energy by 10.
520-570	If ship is hit, changes sprite #3 to red explosion, makes explosion sound, increments ships destroyed, reduces energy by 10, deletes sprite #3. Checks for 15 ships destroyed or out of energy for end of game or returns sprite #1 to gun sight and branches to beginning of new target loop.
580-590	Sounds and prints warning for satellite launching.
600-650	Creates satellite sprite and gradually increases the size.
660-700	Moves scope if user indicates or tests for hit if fire button or key is pressed. Energy is reduced by 10 for each shot fired.
710-740	If satellite is hit, it explodes; if not there is a larger blast and energy level is reduced by 50.
750-830	Sounds and messages for end of game depending on number of ships or energy level.
840-860	Displays option to play again.
870	Subroutine to move gun sight with joystick.
880-930	Subroutine to move gun sight with arrow keys on keyboard.

Note:
If using joysticks with the TI-99/4A, release the ALPHA LOCK key.

```
100 REM ****************
110 REM * SPACE PATROL *
120 REM ****************
130 REM BY DEAN CLEVELAND
140 REM 99'ER VERSION 1.5.1
150 CALL CLEAR :: CALL SCREEN(2)
160 FOR X=0 TO 8 :: CALL COLOR(X,16,1):: NEXT X
170 DISPLAY AT(9,4)BEEP:"*** SPACE PATROL ***": : : : :"     BY DEAN CLEVELAND"
180 CALL CHAR(112,"0000000000000103010000000000000000000000000080C080000000000000
    000"):!SHOT
190 CALL CHAR(116,"0209200421B80150001280200440080020084400210800248100240094011
    080"):!EXP
200 CALL CHAR(104,"1500400040004000400040004000150054000100010001000100010001005
    400"):!GUN SIGHT
210 CALL CHAR(103,"00000018"):: CALL CHAR(111,"00000001"):: CALL CHAR(95,"0000001
    01"):!STARS
220 TR$(1)="000000003849888F8F98483800000000000000000181412F1F21418100000000"
230 TR$(2)="014141417F717F7F477F4040404000000806282828FE8EFEFEE2FE020202020000"
240 TR$(3)="00000E0E0E0E0E0E3F7FCECE7F3F1F0000007070707070FCFE7373FEFCF800"
250 TR$(4)="0101010207E1919F9F90F1607F000000808080408E08F91F1F1118F06FE000000"
260 CALL CHAR(124,"0000000000000201010200000000000000000000000004080804000000000
    000"):!K1
270 CALL CHAR(128,"0000040D030607030D040000000000000000020B0C0E060C0B020000000000
    000"):!K2
280 CALL CHAR(132,"000C0D03070E07030D0C0000000000000009BD8E070B870E0D898000000000
    000"):!K3
290 CALL CHAR(136,"006061130F0F1E3CAE1D0F0F13616000000686CBF0F0887C3C78F0F0C8860
    600"):!K4
300 CALL CHAR(140,"C0C1271F1F3A3D7A7A3D321F1F27C1C00383E4F85CBC5E5EAC5CF8F8E48330
    3"):!K5
310 CALL CLEAR
320 DISPLAY AT(10,1):"CHOOSE METHOD OF INPUT:": :"  1  JOYSTICK": :
    "  2  KEYBOARD--ARROW KEYS": :"        AND 'Y' TO FIRE"
330 CALL KEY(0,KEY,STAT):: IF (KEY<49)+(KEY>50)=-1 THEN 330 ELSE INP=KEY-48
340 CALL CLEAR :: LB=400 :: SD=0 :: RANDOMIZE
350 DISPLAY AT(10,1):"CHOOSE SKILL LEVEL:": :"  1  FOR HIGH": :"  2  FOR LOW"
360 CALL KEY(0,KEY,STAT):: IF (KEY<49)+(KEY>50)=-1 THEN 360 ELSE SK=KEY-48
370 CALL MAGNIFY(SK+2):: CALL CLEAR
380 CALL COLOR(8,16,1,9,11,1,10,3,1)
390 FOR X=1 TO 40 :: CALL HCHAR(INT(23*RND+1),INT(26*RND+3),
    INT(3*RND+1)*8+87):: NEXT X
400 CALL SPRITE(#1,104,16,98,130)
410 DISPLAY AT(23,1):"SHIPS DESTROYED=";SD:"ENERGY UNITS REMAINING=";LB
420 TAR=INT(6*RND)+1 :: IF TAR>4 THEN 580 :: CALL CHAR(120,TR$(TAR))
430 FOR L=1 TO 4 :: CALL COLOR(INT(2*RND+9),INT(11*RND+3),1):: NEXT L
440 VM=INT(19*RND)-9 :: HM=INT(38*RND)-19 :: IF VM=0 OR HM=0 THEN 440
450 CALL SPRITE(#3,120,INT(14*RND+3),210,INT(256*RND)+1,VM,HM)
460 ON INP GOSUB 870,880
470 CALL KEY(2,K,S):: IF K<>18 THEN 460 :: CALL MOTION(#1,0,0)
480 CALL SOUND(300,1800,2,2300,2,2800,2):: CALL COINC(ALL,HIT)
490 CALL PATTERN(#1,112):: CALL COLOR(#1,7)
500 IF HIT THEN 520 :: LB=LB-10
510 CALL COLOR(#1,16):: CALL PATTERN(#1,104):: DISPLAY AT(24,25):LB :
    : IF LB<10 THEN 780 ELSE 460
520 CALL PATTERN(#3,116):: CALL COLOR(#3,7,#1,1)
530 SD=SD+1 :: LB=LB-10
540 CALL SOUND(200,-7,5):: CALL SOUND(700,-7,2)
550 FOR L=1 TO 80 :: NEXT L :: CALL DELSPRITE(#3)
560 DISPLAY AT(23,17):SD :: DISPLAY AT(24,25):LB :: CALL COLOR(#1,16):
    : IF SD>14 THEN 810 :: IF LB<10 THEN 780
570 CALL PATTERN(#1,104):: GOTO 420
580 FOR L=1 TO 4 :: DISPLAY AT(1,9):"**-WARNING-**" :: CALL SOUND(100,600,0,700,
    0,-2,0):: FOR D=1 TO 30 :: NEXT D
590 DISPLAY AT(1,9):" " :: FOR D=1 TO 20 :: NEXT D :: NEXT L
600 CALL SPRITE(#3,124,4,INT(80*RND+20),INT(100*RND+80),2,2)
610 FOR ATT=1 TO 5 :: CALL PATTERN(#3,ATT*4+120)
620 FOR D=1 TO 4
630 ON INP GOSUB 870,880
640 CALL KEY(2,K,S):: IF K=18 THEN 670
650 NEXT D
660 GOTO 700
670 CALL MOTION(#1,0,0):: CALL SOUND(300,1800,2,2300,2,2800,2):: CALL COINC(#1,#
    3,SK*5,HIT)
680 CALL PATTERN(#1,112):: CALL COLOR(#1,7):: IF HIT THEN 740 :: LB=LB-10
690 CALL COLOR(#1,16):: CALL PATTERN(#1,104):: DISPLAY AT(24,25):LB :
    : IF LB<10 THEN 780
700 NEXT ATT
710 CALL DELSPRITE(#1):: CALL MAGNIFY(4):: CALL PATTERN(#3,116)
720 CALL SOUND(1700,-7,0):: FOR D=1 TO 30 :: CALL SCREEN(7):
    : CALL SCREEN(10):: NEXT D :: CALL SCREEN(2)
730 CALL DELSPRITE(#3):: CALL MAGNIFY(SK+2):: LB=LB-50 :: DISPLAY AT(24,25):LB :
    : IF LB<=0 THEN 750 ELSE 400
740 CALL PATTERN(#3,116):: CALL COLOR(#3,7,#1,1):: LB=LB-10 :: GOTO 540
750 CALL DELSPRITE(#1,#3):: CALL SOUND(1000,110,0,130,0,150,0)
760 DISPLAY AT(8,1):"YOUR SHIP HAS BEEN DAMAGED": :"BEYOND REPAIR.  BECAUSE OF":
    :"YOUR FAILURE YOU HAVE BEEN"
770 DISPLAY AT(14,1):"DEMOTED TO PRIVATE!!!" :: GOTO 840
780 CALL DELSPRITE(#1,#3):: FOR L=1 TO 6 :: CALL SOUND(60,2000,0):: NEXT L
790 DISPLAY AT(8,1):"YOUR POOR SHOOTING HAS": :"CAUSED YOU TO RUN OUT OF";
    :"ENERGY UNITS. RETURN TO BASE"
800 DISPLAY AT(14,1):"AT ONCE!! YOU WILL BE SENT": :"BACK TO THE TRAINING
    CENTER!" :: GOTO 840
810 CALL DELSPRITE(#1):: FOR L=2000 TO 4500 STEP 250 :: CALL SOUND(100,L,0):
    : NEXT L
820 DISPLAY AT(8,1):"YOU GOT 'EM ALL CAPTAIN!!!": :"YOUR MISSION WAS A SUCCESS."
    : :"YOU ARE HEREBY PROMOTED"
830 DISPLAY AT(14,1):"TO FLEET COMMANDER!!"
840 FOR D=1 TO 700 :: NEXT D :: DISPLAY AT(19,1)BEEP:"PLAY AGAIN? (Y OR N)"
850 CALL KEY(0,KEY,STAT):: IF KEY=89 THEN 310
860 IF KEY<>78 THEN 850 ELSE CALL CLEAR :: END
870 CALL JOYST(2,C,R):: CALL MOTION(#1,-R*4,C*4):: RETURN
880 CALL KEY(0,KEY,STAT):: IF STAT=0 THEN CALL MOTION(#1,0,0):: RETURN
890 IF KEY=69 THEN 900 ELSE IF KEY=68 THEN 910 ELSE IF KEY=88 THEN 920
    ELSE IF KEY=83 THEN 930 :: RETURN
900 CALL MOTION(#1,-16,0):: RETURN
910 CALL MOTION(#1,0,16):: RETURN
920 CALL MOTION(#1,16,0):: RETURN
930 CALL MOTION(#1,0,-16):: RETURN
```


Force 1

```
1000 SCO=SCO+SC
1010 IF SCO>=25 AND SCO-SC1>=25 THEN TIME=1000 :: SC1=SCO :: DISPLAY AT(2,3):
     "BONUS GAME" :: GOTO 260
1020 CALL CHARSET :: CALL SCREEN(6):: CALL DELSPRITE(#1,#2,#3)
1030 IF SCO>=40 THEN 1080
1040 IF SCO>=30 THEN 1100
1050 IF SCO>=20 THEN 1120
1060 IF SCO>=10 THEN 1140
1070 IF SCO>=5 THEN 1160 ELSE 1180
1080 DISPLAY AT(4,1):"A VERY GOOD BATTLE.":"YOUR NAME WILL GO DOWN IN":"HISTORY
     AS ONE OF THE"
1090 DISPLAY AT(7,1):"GREATEST STARSHIP CAPTAINS":"OF YOUR TIME.":"YOUR SCORE=";
     SC :: GOTO 1200
1100 DISPLAY AT(4,1):"YOU ARE TO BE CONGRATULATED":"ON YOUR FINE MISSION. FEW"
     :" PILOTS HAVE ATTAINED SUCH"
1110 DISPLAY AT(7,1):"SUCCESS. GOOD LUCK ON FUTURE":"MISSIONS.":"YOUR SCORE=";
     SC :: GOTO 1200
1120 DISPLAY AT(4,1):"A FAIR SHOWING. THE ALIENS":"HAVE BEEN TURNED BACK AND"
     :"YOUR HOME WORLD IS SAFE."
1130 DISPLAY AT(7,1):"YOUR SCORE=";SC :: GOTO 1200
1140 DISPLAY AT(4,1):"YOUR FLEET WAS BADLY DAMAGED":"IN THE FIGHT, BUT YOU":
     "MANAGED TO FIGHT OFF THE"
1150 DISPLAY AT(7,1):"ALIEN ATTACK. BETTER LUCK":"NEXT TIME.":"YOUR SCORE=";SC :
     : GOTO 1200
1160 DISPLAY AT(4,1):"YOUR FLEET HAS BEEN":"DESTROYED. YOU ARE THE":
     "ONLY SURVIVOR."
1170 DISPLAY AT(7,1):"YOUR HOME PLANET IS SAFE":"AT LEAST UNTIL THE NEXT":
     "ATTACK.":"YOUR SCORE=";SC :: GOTO 1200
1180 DISPLAY AT(4,1):"ALL HOPE IS LOST IN TRYING":"TO SAVE YOUR PLANET.":
     "YOUR MISSION HAS FAILED"
1190 DISPLAY AT(7,1):"AND YOU ARE DISGRACED.":"YOUR SCORE=";SC :: GOTO 1200
1200 DISPLAY AT(10,1):"DO YOU WISH TO PLAY AGAIN":"ENTER (Y/N).";
1210 ACCEPT AT(11,14)SIZE(1)VALIDATE("YN"):ANS$
1220 IF ANS$="N" THEN END ELSE SC,SCO,SC1,TIME=0 :: CALL MAGNIFY(1):: GOTO 230
1230 CALL MAGNIFY(1):: CALL PATTERN(#1,103+D):: RETURN
1240 CALL PATTERN(#1,112):: CALL MAGNIFY(3):: RETURN
1250 CALL PATTERN(#1,116):: RETURN
1260 CALL PATTERN(#1,120):: RETURN
1270 CALL MAGNIFY(4):: CALL POSITION(#1,PO1,PO2):: IF PO1>8 AND PO2>8 THEN CALL
     LOCATE(#1,PO1-8,PO2-8)
1280 IF PO1<110 AND PO1>36 AND PO2<148 AND PO2>88 THEN GOTO 1330
1290 IF D1>10 OR D2>10 THEN 1310
1300 CALL MOTION(#1,D1*5+40*SIN(D1),D2*5+40*SIN(D2)):: FOR TD=1 TO 20 :: NEXT TD
1310 CALL DELSPRITE(#1):: SP1=0 :: DIS=11000 :: D=1 :: FOR MO=2 TO 15 :
     " CALL MOTION(#MO,0,0):: NEXT MO
1320 SA=0 :: SB=0 :: RETURN
1330 CALL POSITION(#1,D3,D4):: CALL VCHAR(D3/8+3,D4/8+2,34,21-D3/8):
     : CRASH=CRASH+1 :: CALL SOUND(1000,110,2,220,2,10000,30,-4,2)
1340 CALL VCHAR(D3/8+3,D4/8+2,32,21-D3/8):: GOSUB 1300
1350 CALL SOUND(300,110,2,220,2,20000,30,-8,2)
1360 CALL SOUND(500,440,2,660,2,3000,30,-4,2)
1370 CALL SOUND(600,110,2,220,2,5000,30,-8,2)
1380 CALL SOUND(1000,220,2,330,2,1000,30,-8,2):: SA,SB=0 :: GOTO 980
1390 Z$="B161163BC4241211"
1400 CALL COLOR(13,16,1)
1410 Z1$="0000001000000000"
1420 ST=2
1430 CALL CHAR(128,Z1$)
1440 CALL CHAR(129,"00"):: CALL CHAR(130,"00"):: CALL CHAR(131,"00")
1450 FOR ST=2 TO 15
1460 STA1=INT(RND*256)+1 :: STA2=INT(RND*256)+1
1470 CALL SPRITE(#ST,128,16,STA1,STA2)
1480 NEXT ST
1490 RETURN
1500 END
1510 DISPLAY AT(11,8):"***********"
1520 DISPLAY AT(12,8):"* FORCE 1 *"
1530 DISPLAY AT(13,8):"***********"
1540 GOSUB 410 :: DISPLAY AT(24,1):"PRESS ANY KEY TO CONTINUE"
1550 CALL KEY(0,K,S):: IF S<>0 THEN CALL SOUND(-1,40000,30):: CALL CLEAR :
     : RETURN ELSE CALL MAGNIFY(1)
1560 T1=INT(RND*192)+1 :: T2=INT(RND*256)+1 :: CALL SPRITE(#1,104,2,T1,T2,INT(RN
     D*7)-3,INT(RND*7)-3)
1570 D=0
1580 D=D+1 :: IF D<9 THEN GOSUB 1230 ELSE ON D-8 GOSUB 1240,1250,1260
1590 CALL SOUND(-4000,600,(11-D)*3,400,(D-1)*3)
1600 CALL KEY(0,K,S):: IF S<>0 THEN CALL DELSPRITE(#1):
     : CALL SOUND(-1,40000,30) :: CALL CLEAR :: RETURN
1610 IF D<11 THEN 1580 ELSE CALL DELSPRITE(#1):: GOTO 1550
```

99'er TI Fest

is Coming...

to San Francisco

October 22–24, 1982

DON'T MISS IT!

ATTEND Applications Seminars for Using Personal Computers in Business, Science, Engineering, and for Home Management & Financial Decision-Making!

SEE & OPERATE the Latest Personal Computer Hardware & Software Products from TI & Third Parties!

ATTEND Programming Language Tutorial Seminars in TI BASIC, Extended BASIC, LOGO, TMS9900 Assembly Language, UCSD Pascal, & PILOT!

WATCH Demonstrations of TI's State-of-the-Art Technology & Catch a Glimpse of Products of the Future!

PLAY Spectacular Personal Computer Arcade Games . . . Enter the Competition to Win Computers, Peripherals, and Software!

WITNESS Unique Applications for TI Electronic Learning Aids & Innovations in Computer Assisted Instruction!

MEET with Users Groups and Share Ideas with Others of Similar Interest!

SAVE Money by Purchasing the Products You Need at Special Show Discounts!

PARTICIPATE in the Users Forum to Voice Your Opinions & Ask Questions Directly to TI!

LEARN How to Profitably Produce Software and Hardware for this Dynamic, New Mass Market!

A UNIQUE OPPORTUNITY for Dealers, Manufacturers, & Product Developers to Display & Sell Directly to Thousands of Eager Buyers!

HUNDREDS of Valuable Door Prizes!

RIDE a Cable Car, Sample the Delicacies of China Town & Fisherman's Wharf, and Cross the Famous "Golden Gate"!

TRANSPORTATION, Sightseeing, and Hotel Discount Packages Available!

And MUCH, MUCH, MORE!

See the Many Reasons Why the Versatile Texas Instruments Personal Computers have become the Value & Innovation Leaders.

ATTENTION SHOW EXHIBITORS:
Call or Write for Booth Contracts and Exhibitor Information Kits.

ATTENTION SOFTWARE DEVELOPERS:
Intensive multi-day workshops will be held prior to and after 99'er TI-Fest as follows:

- TMS9900 Assembly Language Programming
  Monday, October 18—Wednesday, October 20, 1982
- UCSD Pascal™ System and Pascal Language Programming
  Monday, October 25—Tuesday, October 26, 1982

99'er TI Fest

A Personal Computer Conference, Trade Show & Seminar
Presented by 99'er Magazine

Special Bind-In Registration Packet in Next Issue

99'er Magazine • 2715 Terrace View Drive • Eugene, OR 97405 • Tel. (503) 485-8796

99'er magazine™ ON TAPE

Selected Programs from 99'er magazine™ Are Now Available on Cassette Tape As A Service to Our Readers.*

- 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.

* Not for Duplication or Resale. This service will exist only as long as it fulfills a useful function and isn't abused.

Use the bind-in card in the back of the magazine for your convenience in ordering.

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
This package available for shipment after June 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 #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

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.

so ORDER TODAY!

Each Only $3.95 postpaid

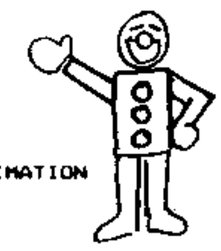
ISSUE # 4 (Partial Contents):
•The Joy of Computer Gaming •Designing Your Own Computer Game •Bit-Plot Printer Graphics •A BASIC Program That Writes BASIC Programs •Micro Bartender •Fundamentals of Assembly Language Programming •Games of Time and Direction •Computer Inventory •A Joystick Video Game in 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.

Index to Advertisers

Spriter . . . from p. 45

```
500 FOR K=0 TO NS :: PRINT #1:ID$(K),
    CHA$(K):: NEXT K :: CLOSE #1
510 END
520 SUB DRAWER(TP$,C$,NS,AN$,CHA$(),
    ID$()):: CALL CHAR(33,RPT$("F",16))
530 IF C$="" THEN 580
540 INPUT "DO YOU WANT TO INITIALIZE WITH A
    PREVIOUSLY DEFINED CHARACTER(Y/N)?":AN$
550 IF AN$="N" THEN C$="" :: GOTO 580 ELSE
    IF AN$<>"Y" THEN 540
560 INPUT "ENTER INDEX OF CHARACTER
    DESIRED,ANY '-' VALUE FOR MOST
    RECENTLY DEFINED":NOS
570 IF NOS<0 THEN 580 ELSE C$=CHA$(NOS)
    : NXX=NOS :: GOTO 590
580 NXX=NS-1
590 M=16 :: IF LEN(C$)=0 THEN
    C$=RPT$("0",64):: F=0 ELSE F=1
600 IF LEN(C$)=16 THEN C$=C$&RPT$("0",48)
610 N=1 :: C1$=SEG$(C$,1,16)
    : C2$=SEG$(C$,17,16)
    : C3$=SEG$(C$,33,16)
    : C4$=SEG$(C$,49,16)
620 PRINT "USE ARROW KEYS AND 'W,R,C,Z' TO
    MOVE CURSOR,OR TO CHANGE POLARITY
    USE'F'FOR DARK AND 'A'FOR LIGHT."
630 CALL KEY(0,K,S):: IF S=0 THEN 630
640 CALL CLEAR :: CALL HCHAR(4,4,30,M+2):: CALL HCHAR(M+5,4,30,M+2)
650 CALL VCHAR(5,4,30,M):: CALL VCHAR(5,M+5,30,M):: X,Y=5
660 IF AN$="Y" THEN CALL EXPANDER(C$,X,Y)
670 IF NXX>=0 THEN DISPLAY AT(2,1):ID$(NXX):: DISPLAY AT(22,1):C$
680 CALL HCHAR(X,Y,30,1):: XO,YO=0 :: CALL ADDPIX(XO,YO,N,C$)
690 CALL KEY(1,K,S)
700 IF S=0 THEN 690 ELSE IF N=1 THEN CALL HCHAR(X,Y,33,1)
    ELSE CALL HCHAR(X,Y,32,1)
710 IF K=1 THEN N=0
720 IF K=12 THEN N=1
730 IF K=5 AND X>5 THEN X=X-1
740 IF K=0 AND X<M+4 THEN X=X+1
750 IF K=2 AND Y>5 THEN Y=Y-1
760 IF K=3 AND Y<M+4 THEN Y=Y+1
770 IF K=4 AND X>5 THEN IF Y>5 THEN X=X-1 :: Y=Y-1
780 IF K=6 AND X>5 THEN IF Y<M+4 THEN X=X-1 :: Y=Y+1
790 IF K=15 AND X<M+4 THEN IF Y>5 THEN X=X+1 :: Y=Y-1
800 IF K=14 AND X<M+4 THEN IF Y<M+4 THEN X=X+1 :: Y=Y+1
810 IF K=18 THEN 900
820 IF X>4 AND X<13 THEN IF Y>4 AND Y<13 THEN P=1 ELSE P=3
    ELSE IF Y>4 AND Y<13 THEN P=2 ELSE P=4
830 IF P=1 THEN XO=X-5 :: YO=Y-5 :: CH$=SEG$(C$,1,16)
840 IF P=2 THEN XO=X-13 :: YO=Y-5 :: CH$=SEG$(C$,17,16)
850 IF P=3 THEN XO=X-5 :: YO=Y-13 :: CH$=SEG$(C$,33,16)
860 IF P=4 THEN XO=X-13 :: YO=Y-13 :: CH$=SEG$(C$,49,16)
870 CALL ADDPIX(XO,YO,N,CH$)
880 IF P=1 THEN C1$=CH$ ELSE IF P=2 THEN C2$=CH$
    ELSE IF P=3 THEN C3$=CH$ ELSE C4$=CH$
890 CALL HCHAR(X,Y,30,1):: C$=C1$&C2$&C3$&C4$ :: GOTO 690
900 DISPLAY AT(22,1):"ENTER SPRITE NAME." :: DISPLAY AT(23,1):
    "" :: DISPLAY AT(24,1):""
910 ACCEPT AT(23,1):ID$(NS)
920 IF TP$="N" THEN SUBEXIT
930 DISPLAY AT(22,1):"WANT TO COPY ON T.P.(Y/N)?" :: ACCEPT AT(23,1):AN$
940 IF AN$="N" THEN SUBEXIT ELSE IF AN$<>"Y" THEN 930
950 DISPLAY AT(2,1):ID$(NS):: DISPLAY AT(22,1):C$
960 CALL SCREEPT :: SUBEND
970 SUB ADDPIX(X,Y,N,C$)
980 DEF B(A)=INT(NH/(2^A))-2*INT(NH/(2^(A+1)))
990 IF Y<4 THEN Z=2*X+1 :: YO=3-Y ELSE Z=2*X+2 :: YO=7-Y
1000 A2$=SEG$(C$,Z,1)
1010 IF Z>1 THEN A1$=SEG$(C$,1,Z-1)
1020 IF Z<16 THEN A3$=SEG$(C$,Z+1,16-Z)
1030 NH=ASC(A2$):: IF NH<=57 THEN NH=NH-48 ELSE NH=NH-55
1040 IF B(YO)=0 AND N=1 THEN NH=NH+2^YO
1050 IF B(YO)=1 AND N=0 THEN NH=NH-2^YO
1060 IF NH<=9 THEN A2$=STR$(NH)ELSE A2$=CHR$(NH+55)
1070 IF Z=16 THEN C$=A1$&A2$
1080 IF Z=1 THEN C$=A2$&A3$
1090 IF Z<>16 AND Z<>1 THEN C$=A1$&A2$&A3$
1100 SUBEND
1110 SUB EXPANDER(C$,XO,YO)
1120 DEF B(A)=INT(NH/(2^A))-2*INT(NH/(2^(A+1)))
1130 FOR I=0 TO 15 :: FOR J=0 TO 15
1140 IF J>7 THEN JO=J-8 ELSE JO=J
1150 IF I>7 THEN IO=I-8 ELSE IO=I
1160 IF I<8 THEN IF J<8 THEN L=1 ELSE L=3 ELSE IF J<8 THEN L=2 ELSE L=4
1170 IF JO<4 THEN Z=2*IO+1 :: Y=3-JO ELSE Z=2*IO+2 :: Y=7-JO
1180 REM
1190 A2$=SEG$(C$,Z+16*(L-1),1)
1200 NH=ASC(A2$):: IF NH<=57 THEN NH=NH-48 ELSE NH=NH-55
1210 IF B(Y)=1 THEN CALL HCHAR(XO+1,YO+J,33,1)
1220 NEXT J :: NEXT I :: SUBEND
1230 SUB SCREEPT
1240 OPEN #255:"TP.U.E.S",OUTPUT :: FOR X=1 TO 24 :: S$=""
1250 FOR Y=1 TO 32 :: CALL GCHAR(X,Y,Z):: S$=S$&CHR$(Z)
1260 NEXT Y :: PRINT #255:S$ :: NEXT X :: CLOSE #255
1270 SUBEND
1280 SUB CASTER(@FILE$,N,I$(),C$())
1290 OPEN #2:@FILE$,INTERNAL,INPUT ,FIXED 128 :: GOTO 1300
1300 INPUT #2:NAM$,N
1310 FOR I=0 TO N
1320 INPUT #2:I$(I),C$(I):: NEXT I :: CLOSE #2
1330 N3=23 :: N1=0 :: IF N<=24 THEN N2=N ELSE N2=23
1340 FOR I=N1 TO N2 :: IF I>N THEN 1390
1350 PRINT I;I$(I):: NEXT I
1360 PRINT "PRESS ANY KEY TO CONTINUE."
1370 CALL KEY(0,K,S):: IF S=0 THEN 1370
1380 IF N>N3 THEN N1=N1+24 :: N2=N2+24 :: N3=N3+24 :: GOTO 1340
1390 SUBEND
```

Listing 3.



```
100 REM **********************
110 REM *   SPRITE DEMO 2    *
120 REM **********************
130 REM
140 REM 99'ER VERSION 1.5.1XB
150 REM DEMONSTRATION OF SPRITE ANIMATION
    USING DATA STATEMENTS
160 CALL CLEAR
170 DIM I$(17),C$(17)
180 GOSUB 250 !CASTER
190 FOR I=0 TO N :: CALL CHAR(136-4*I,C$(I))
200 NEXT I
210 CALL CLEAR
220 CALL SPRITE(#1,136,2,30,30,0,-10):: CALL
    MAGNIFY(4)
230 FOR I=0 TO N :: CALL PATTERN(#1,136-4*I):
    : GOSUB 300 :: NEXT I :: GOTO 230
240 END
250 REM SUBROUTINE CASTER
260 READ NAM$,N
270 FOR I=0 TO N
280 READ I$(I),C$(I):: NEXT I
290 RETURN
300 REM SUBROUTINE DELAY
310 FOR J=0 TO 15 :: NEXT J
320 RETURN
330 DATA MAN&NB1,12
340 DATA MAN#1,000609090600F0F0F1E060F0F190804080000000
    0000000000000205080000000
350 DATA MAN#2,030404030707ZF1303030706060207000080800
    08090D0A080B0B0B0B0B0B0000
360 DATA MAN#2.5,060909060F0F172606060E1EA242010300000
    0000000804080000000000000000 0
370 DATA MAN#3,00070903060F0F172F0606060F090818000000
    0000000000000000000804080
380 DATA MAN#4,00001B241C0C1C2C4E1606070602020600000000
    0000000000040A000000000000
390 DATA MAN#5,0000000000000387FDE9662428100010000000
    0000020508000000000B000
400 DATA MAN#6,00061424140C0C0C0C1C1E1E1E1D0C100000000
    0000600000000000000000B080
410 DATA MAN#6.5,0000002020184C7C0C0C0E0606090E0400000
    0000000000000000000000B080
420 DATA MAN#7,0000000000000040804024F1E376640C00000000
    0000000004080000000804020
430 DATA MAN#8,000000110A06030101010303062A12060000008
    44850A0C0C0B0B0000000000000
440 DATA MAN#1,000609090600F0F0F1E060F0F190804080000000
    0000000000000205080000000
450 DATA MAN#3,00070903060F0F172F0606060F090818000000
    0000000000000000000804080
460 DATA MAN#2.5,060909060F0F172606060E1EA242010300000
    0000000804080000000000000000
```

ALL 99'er-ware PRODUCTS ON THESE PAGES MAY BE ORDERED USING THE BIND-IN CARDS NEAR THE FRONT & REAR OF THIS MAGAZINE

DIGITAL COMPUTER CASSETTES with LIBRARY ALBUM

- High - Quality BASF Tape in a 5-Screw Cassette Housing for Data Integrity
- Secure Storage & Quick Convenient Retrieval with Protective File Album
- 8 Special 50-Foot Cassettes — Ideal for Building Up Your 99'er Program Library
- Economical As It Is Functional

☆ Only $11.95 each; or 3 for $30.00

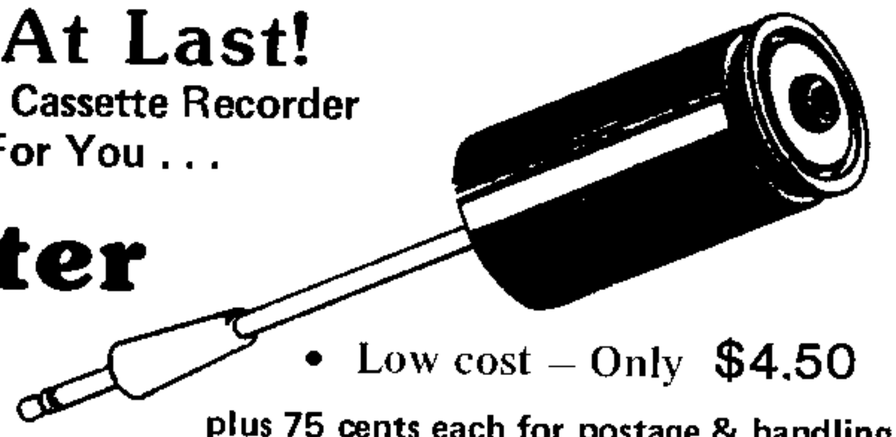
Add $2.00 shipping and handling for the first album; 75¢ for each additional album. Use the bind-in card in the back of the magazine for your convenience in ordering. Telephone orders accepted if charged to credit cards.

Cassette Compatibility At Last!

If The TI-99/4 or TI-99/4A Will Not Control Your Cassette Recorder Through Its Remote Jack, We Have The Solution For You . . .

The TI-SETTE™ Adapter

- Quickly connects between the black lead on the TI Dual Cassette Cable and your recorder's remote jack to establish compatible polarity.

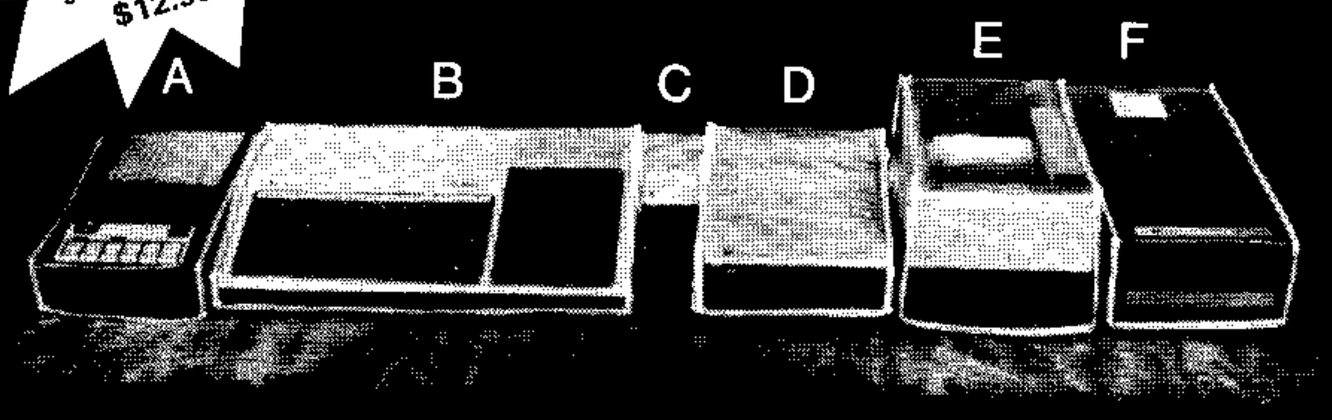


- Low cost — Only $4.50

plus 75 cents each for postage & handling.

★ NEW ★ Peripheral Expansion Box Dustcover Available After June 1, 1982 $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	$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

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