also including ON LoCAItion — The International Journal of Computer Assisted Instruction

$2.95 in USA

99'er magazine

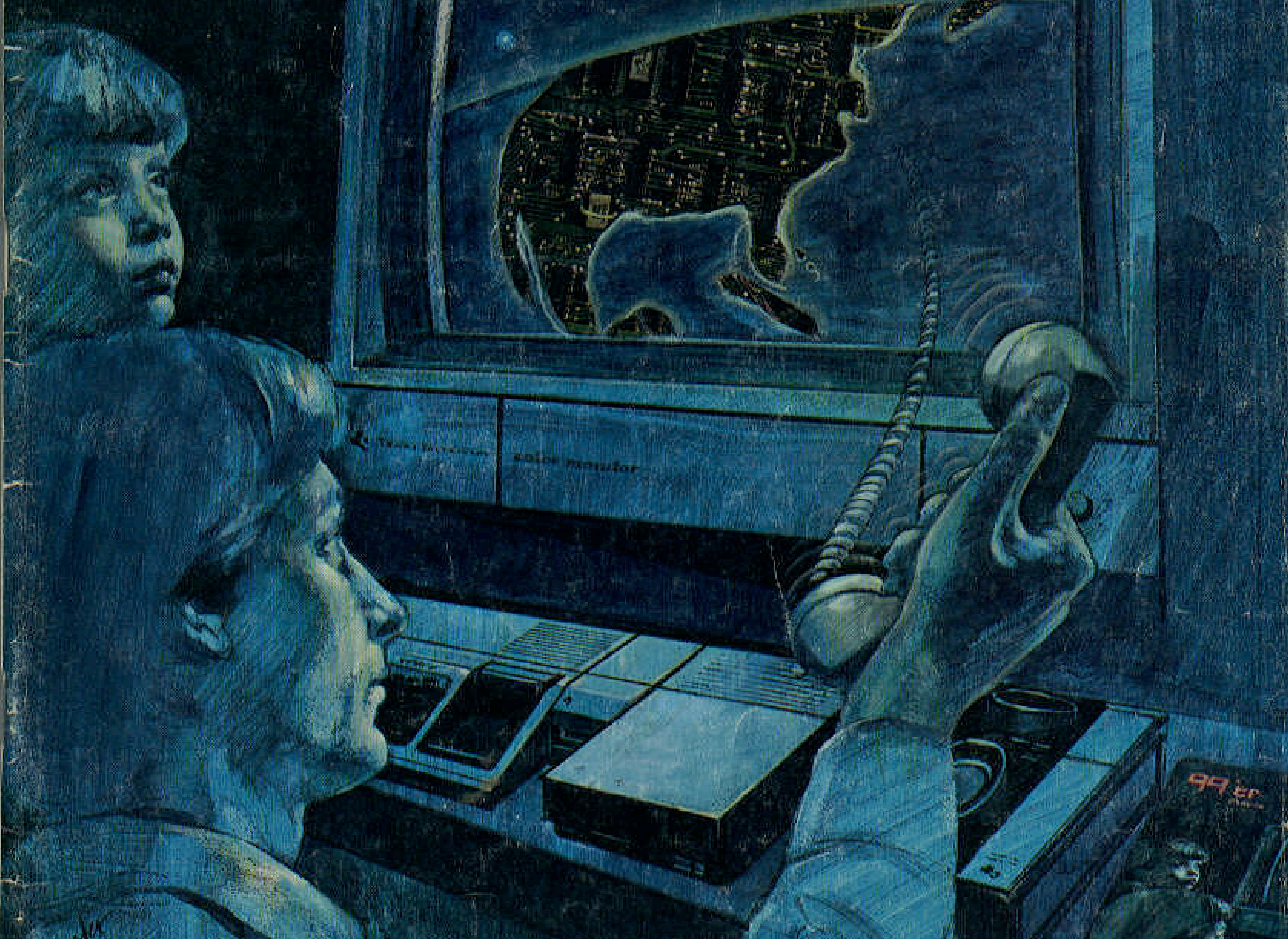
For Users of the Texas Instruments TI-99/4 and other 16-Bit TMS9900-Based Personal Computer Systems

July/August 1981 Vol. 1, No. 2

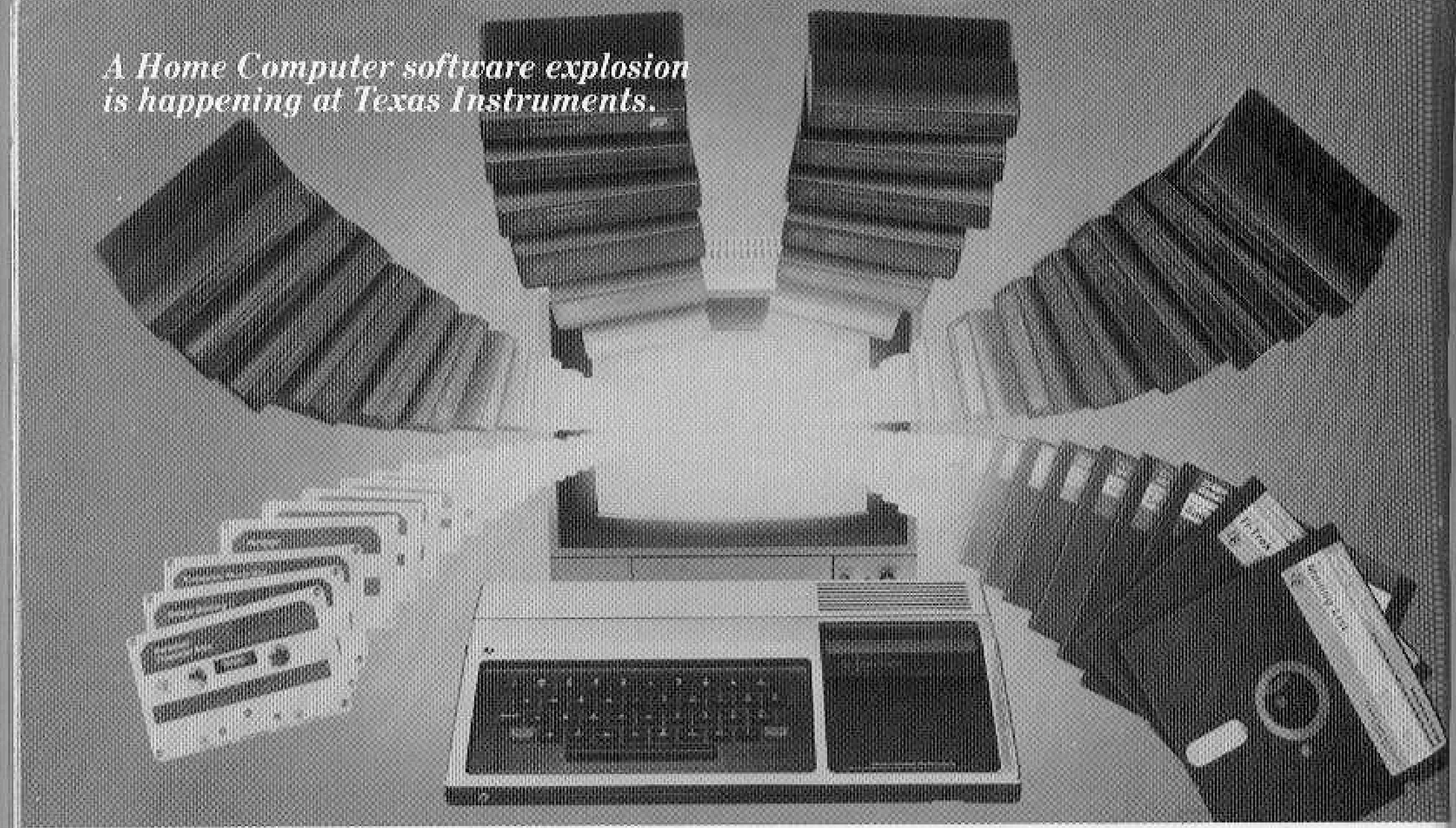
Data Communications, On-Line Retrieval, & Electronic Bulletin Boards

Telephone Dialing By Computer

TI Unveils New Micro



The Home Computer: Your Personal Link to the World of Information



Now there are hundreds of additional new uses for the Texas Instruments TI-99/4 Home Computer—something for every member of your family.

Our library of application programs for the TI Home Computer is growing rapidly. And that's great for TI-99/4 owners. The more programs to choose from, the greater the value and utility of this remarkable home computer.

Programs are available in three different, easy-to-use formats — Solid State Software™ Command Modules, diskette and cassettes — in a wide range of application areas: • Youth Education. • Home Management and Personal Finance. • Entertainment. • And much more.

Many independent software companies are also creating programs for the TI-99/4 Home Computer. Companies like: Scott, Foresman and Company; Micro-Ed, Inc.; Program Design; 99/4 National Users' Group; Creative Computing; Microcomputers Corporation; Image Computer Products; Instant Software; Charles Mann and Associates; and many more.

Get a copy of our Application Program Directory. You can pick up a copy of our latest catalog of TI-99/4 software today at your nearest Texas Instruments Home Computer dealer. To receive a copy by mail, call toll-free **1-800-858-4565.** In Texas call **1-800-692-4279.**

Congratulations to the winners of our Author Incentive Program contest.

We received many innovative and exciting application programs from individuals who entered our recent Author Incentive Program. Our thanks to all those who submitted entries.

First Prize Winner $3000.00
Charles M. Ehninger
Ft. Worth, TX
(Program: Household Inventory)

Second Prize Winner $1000.00/Each
Wayne D. Williams
Mesa, Arizona
(Program: Metric System Tutor)

Fred J. Baker
Motley, MN
(Program: Kepler Planetory Motion)

Norman S. Kerr
St. Paul, MN
(Program: Mendelian Inheritance)

Glen M. Kleinman
Toronto, Ontario CANADA
(Program: Addition Teaching Tools)

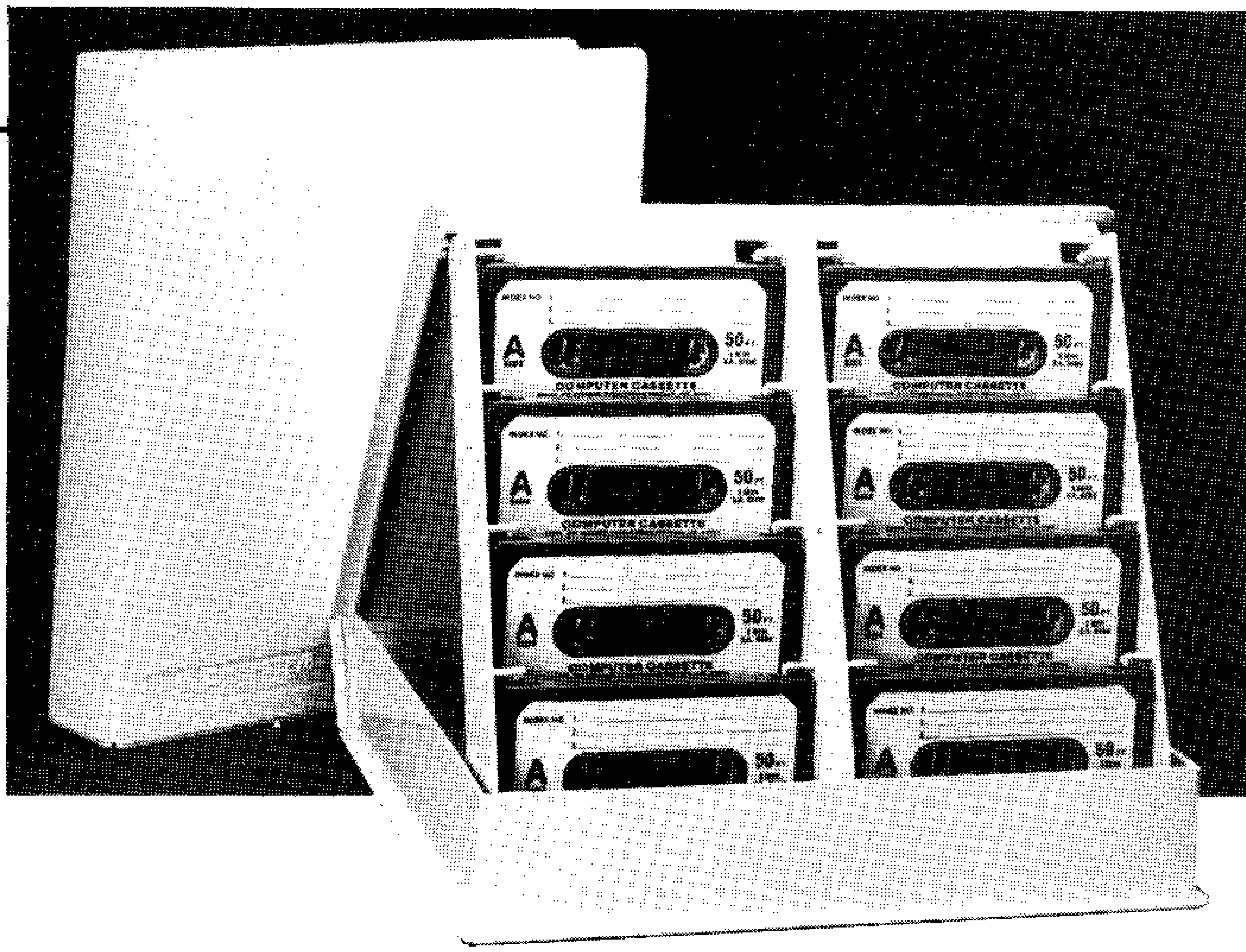
Jack Kirby
Brockport, NY
(Program: Math Remediation)

Third Prize Winners $500.00/Each
Melvin W. Mills, Hancock, MD
Colin James III, Denver, CO
Richard L. Loggins, Albuquerque, NM
Richard A. Ertl, Houston, TX
Alan Winkley, Oakland, CA
Suzanne M. Hawkinson, Lexington, MA
Charles W. Engel, Tampa, FL
Morris C. Burkhart, Alexandria, VA
Robert M. Kiehn, Luling, TX
Chris and Craig Christianson, Tulsa, OK
Williame Q. Christianson, Tulsa, OK
Don C. Rich, Jr., Thomasville, NC
Donald E. Bertrand, Wallingford, CT
Janet Rieker, Los Altos, CA
John M. Anderson, Wilmington, NC
Richard I. Larson, Wilmington, NC
Robert J. Prondzinski, Northbrook, IL
Edward G. Blakeway, Raleigh, NC
Rajeev Jayavant, Lyndhurst, OH
John and Norma Clulow, Perrysburg, OH

Texas Instruments technology—bringing affordable electronics to your fingertips.

TEXAS INSTRUMENTS
INCORPORATED

221140A



99'er magazine™

8 **Information Utilities and the Electronic Cottage**
By Gary M. Kaplan / The Source and TEXNET pave the way.

10 **An Introduction to the Marinchip Systems M9900**
By Duff Kurland / Welcome a new member into our 99'er family.

16 **The Small Investor and the TI-99/4 : A Look at the Dow Jones News Service**
By Lawrence Riley / A profitable way to use your micro.

19 **Data Communications and the TI-99/4**
By F. T. Berkey / Moving bits, bytes, and baud from here to there.

22 **Getting Down to Business**
By George Struble / Evaluating software and return-on-investment.

25 **Kelley's Korner : "Catch & Match" Games in TI BASIC & Extended BASIC**
By W. K. Balthrop, Regena, & Ron Binkowski / 3 super programs that test your memory, agility and luck.

30 **TMS9900 Machine & Assembly Language**
By Dennis Thurlow / The tutorial continues with registers, programming, and the purpose of assemblers.

33 **Software Conversion : TRS-80 to TI BASIC**
By Fred Forster / Learn when it's better to start from scratch.

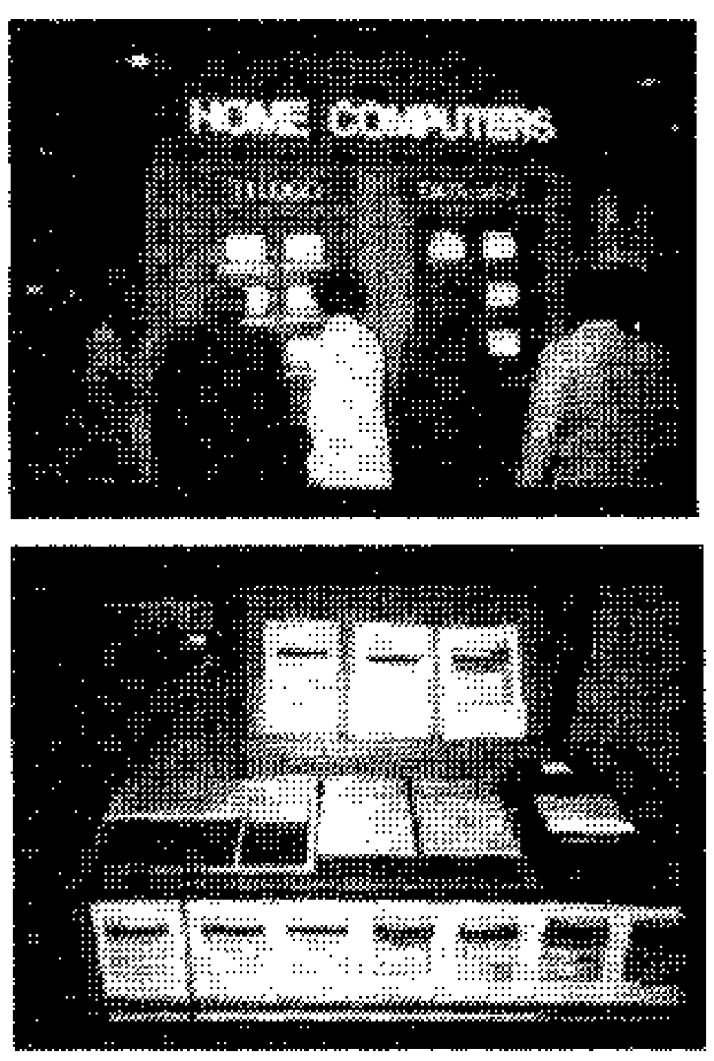
40 **On-Line Information Retrieval**
By Gary M. Kaplan / Database searching is easier than you think.

43 **The Electronic Home Secretary**
By Malladi V. Subbaiah / A computer file system with automatic phone dialer and call timer.

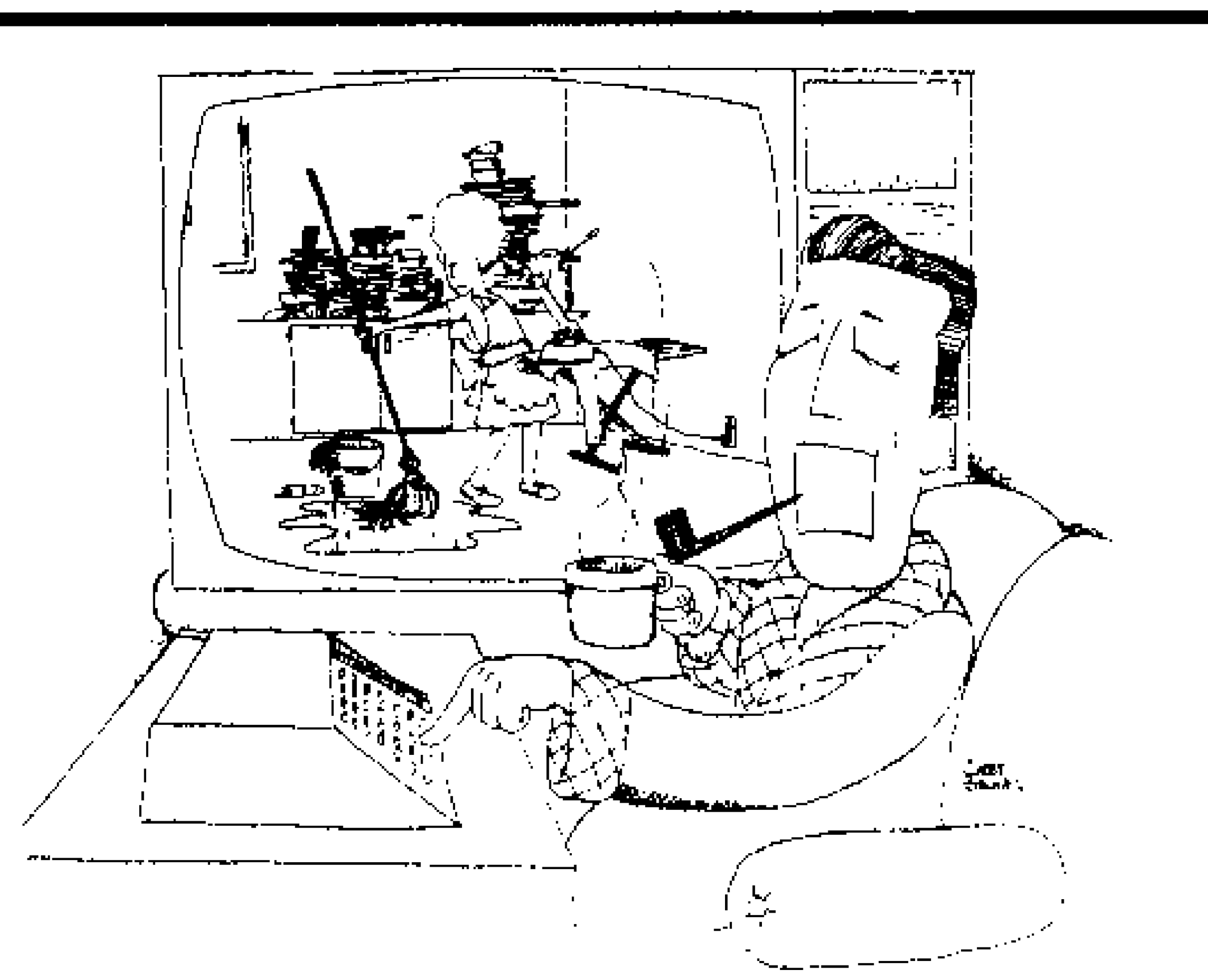
48 **TI at the Consumer Electronics Show**
By Gary M. Kaplan / The new console is finally unveiled.

51 **Bombs Away! (University BASIC Style)**
By Paul R. Roper & Richard T. Adams / An arcade game for the University Board.

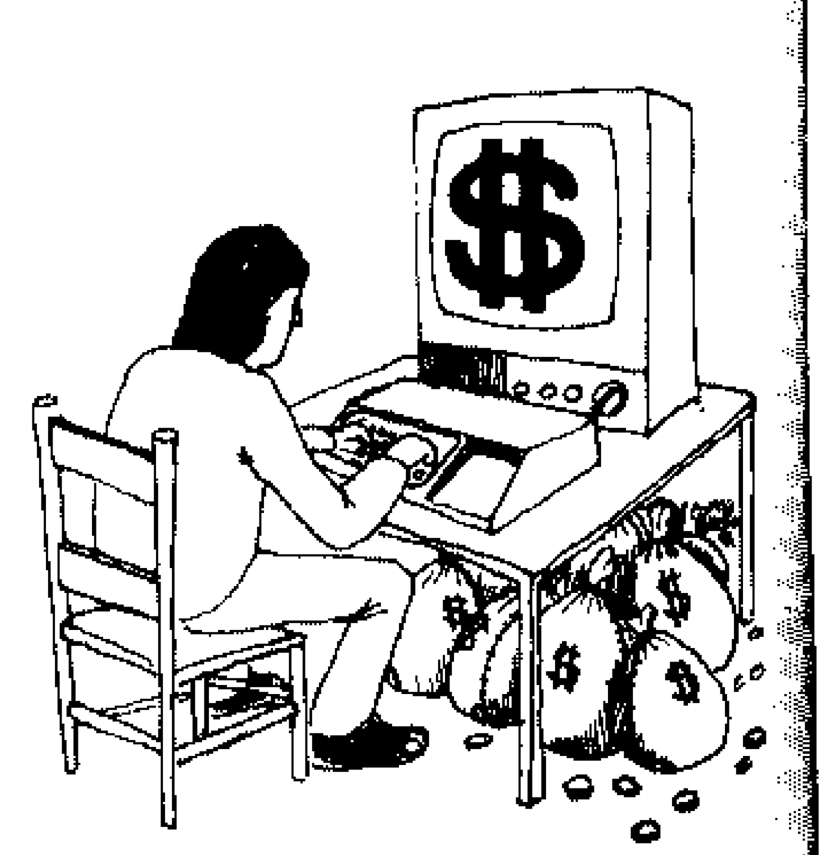
53 **A 99'er Review : HI PAD Digitizer**
By W. K. Balthrop & G. R. Michaels / Enter data the natural way.



Page 48



Page 27



Page 16

Table of Contents: July/August 1981

ON LoCAItion™
THE INTERNATIONAL JOURNAL OF COMPUTER ASSISTED INSTRUCTION.

58 Typing Tutor, Part 1 : Learning the Keyboard Symbols
By Regena / Practice the special keyboard symbols in this tutorial series for learning and improving touch-typing skills.

60 Computer Chess Corner
By Jerry Wolfe / The origin of chess programs plus a worthy opponent for TI's *Video Chess.*

62 Civil Engineering Fundamentals : Simple Beams
By Regena / Solve for reaction forces with this colorful program.

64 TI LOGO and the Space Shuttle
By Henry Gorman Jr. / An update on Lamplighter activities and an introduction to the language itself.

73 Local Networking with Shared Peripherals
By Kathleen M. Swigger / North Texas State University shows us how to make the most of one printer and disk drive.

89 Almost Everything You Ever Wanted To Know About Music . . . But Were Afraid To Ask
By Norma and John Clulow / A lesson in theory and a look at TI's *Music Skills Trainer.*

6 On Screen
66 99'er Bookstore
72 Dealer Directory
85 Programming Hints
94 Letters to the Editor
94 Index to Advertisers

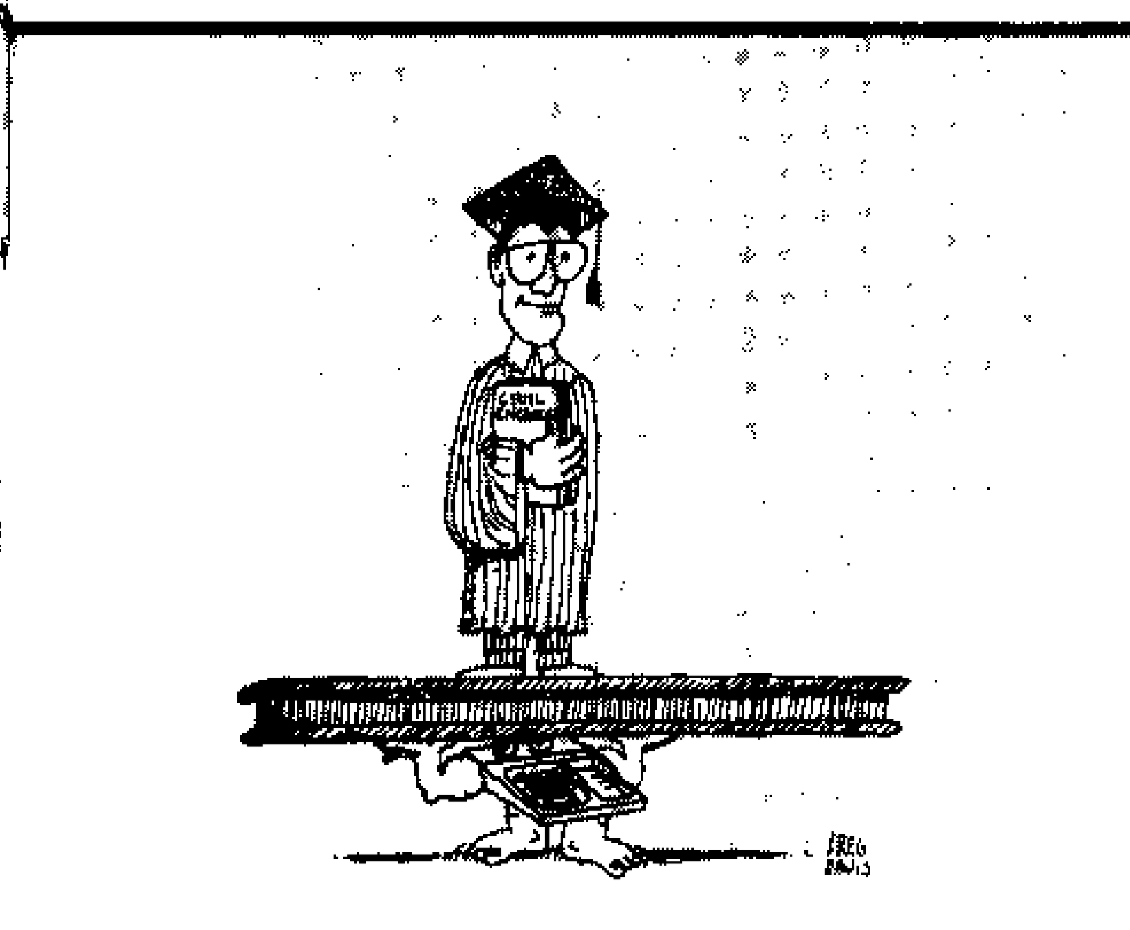
Programming Conventions

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

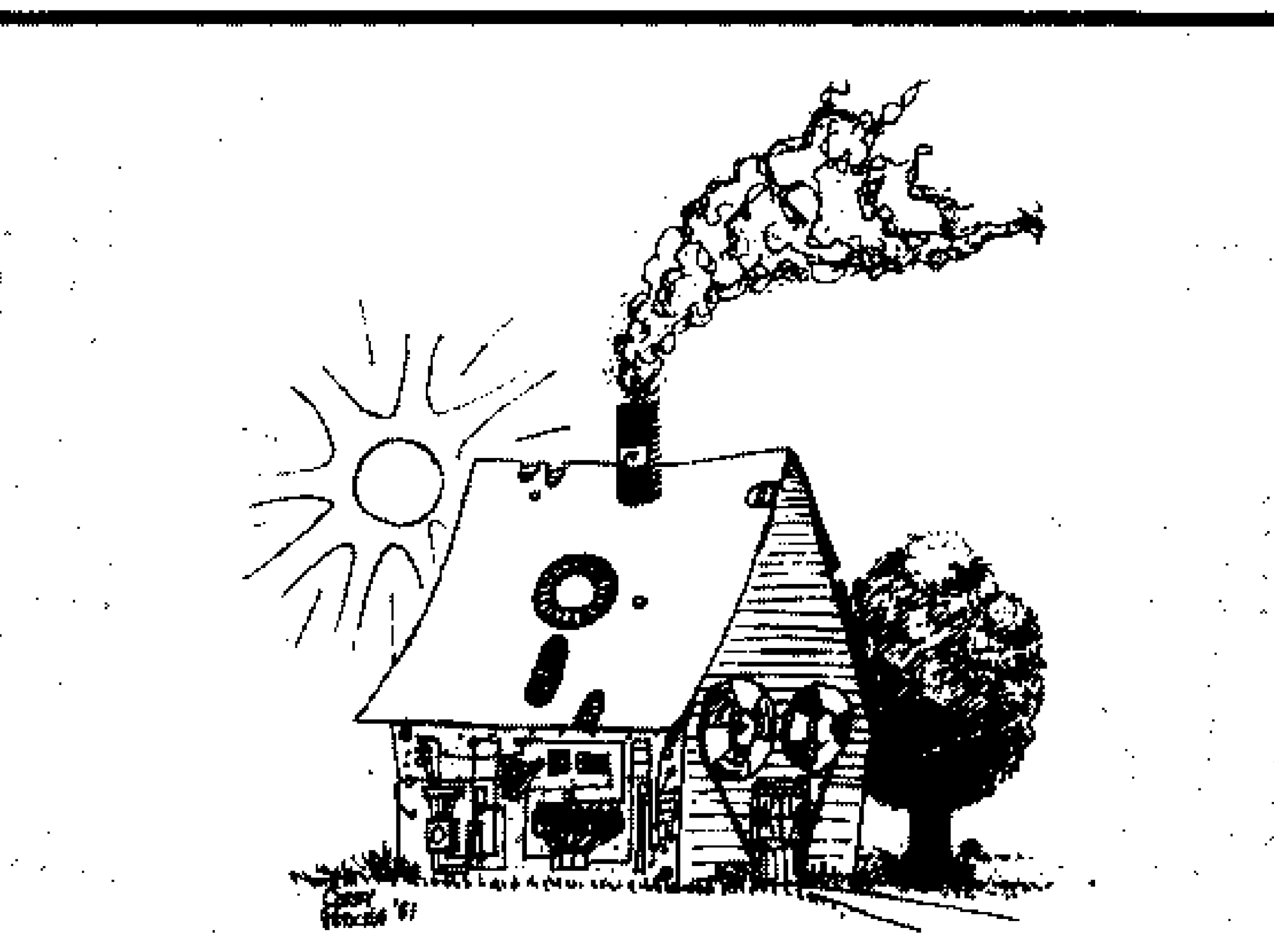
= End of Program or Article

99'ER VERSION 7.81.1XBM

month (7)
year (81)
version (1)
1 = original program
2 … n = no. of update
TI Extended BASIC (XB)
Expansion Memory Required (M)



Page 62



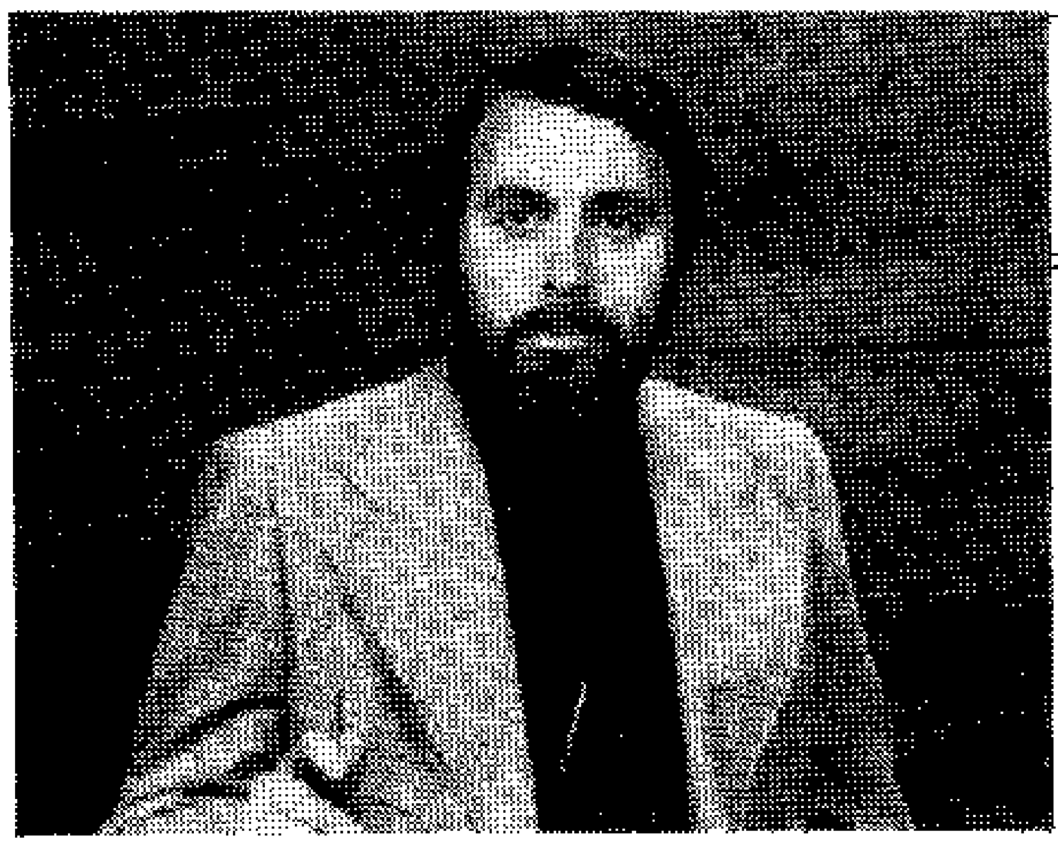
Page 8

Volume 1 No. 2

THIS ISSUE'S COVER:
The vast worldwide network of information resources, depicted in microcosm by the TMS9900 circuit board landmass, is catapulted into the home through the magical link of telephone and home computer. A generation of possibilities separates the son's pensive gaze into the starscape and beyond . . . from that of his father's preoccupation with the astounding reality of the present. The cover art is an original painting by Hayder Amir from a design by Gary Kaplan.

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

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



ON SCREEN

By Gary M. Kaplan
Editor & Publisher

As both an editor and a publisher (and perhaps, somewhat of a mystic), I am well aware that a "theme" can't be *forced* on a particular issue of a magazine without the cohesiveness and "flow" of the magazine taking a telling blow. Time and circumstance dictate successful themes. One either "happens" or it doesn't. In this case, as you've undoubtedly noticed from the cover, we do indeed have a theme—the home computer as our personal link to the world of information resources. It is, however, both pleasing as well as a little surprising for me to admit that *we* didn't select *it* . . . *it* somehow "selected" *us!*

Forgetting for a moment all the other significant things you can do with your own computer, just having a computer to use *solely* as a data terminal is a powerful enough reason for buying one . . . as you'll soon find out in our lead article, *Information Utilities and the Electronic Cottage.*

After you've read all about The Source and TEXNET in the article, Lawrence Riley will show you a profitable way to put your microcomputer to work, as he examines a rather specialized information utility in *The Small Investor and the TI-99/4: A Look at the Dow Jones News Service.* And be sure not to overlook our round-up of Electronic Bulletin Boards that immediately follows.

Later, if you should wonder about what other types of databases exist and how to go about searching them—for your own professional needs, or even for the purpose of launching an information brokering career—look no further than p. 40's *On-line Information Retrieval,* and learn how Boolean algebra works its wonderful magic to make information access as easy as A+B=C.

But all of this talk about information services and methods of database searching brings up an interesting question: How does this bundle of data pass through the telephone lines from one computer to another? Tom Berkey has the answer for you in *Data Communications and the TI-99/4.* He will show you how the bits, bytes, and baud get moved from here to there . . . and back again.

If you've gotten through all the overview and tutorial articles so far, you're about due for a change of pace—perhaps a program. Well then, still keeping in line with our theme, how about giving our *Electronic Home Secretary* a spin. It's a rather special home applications program centered around the use of the telephone. Let Dr. Subbaiah show you how to set up cassette or disk information files, and to have your computer dial the telephone and time your calls.

For the "big news" of this issue, be sure to check out the center photo spread for our on-the-scene, "Windy City" report on *TI at the Consumer Electronics Show,* and a look at the new TI-99/4A console that TI unveiled. The speculation is ended—temporarily.

TI did, of course, display their new software at the Chicago show, but perhaps you can't wait . . . or maybe there's nothing available yet that does what that "neat little TRS-80 program" is supposed to do. If that's the case, let Fred Forster show you the ropes *before* you attempt that language conversion project. You'll save a lot of frustration.

But if, however, converting or writing your own programs does bring on a case of "programmer's headaches," RUN straight-away to *Kelley's Korner* for some more super games in the tradition of our last issue.

After you've had fun and relaxed with Kelley, remember that there's always a time and place for *Getting Down To Business.* This time, George Struble will take you through the basics of evaluating a software package and programming your micro to calculate effective interest rate and return-on-investment.

As George's short program demonstrates, business software does usually require substantial amounts of data to input—the usual input device being the keyboard. Now, in a *99'er Review of the HI PAD Digitizer* you'll learn about a convenient alternative for more natural data entry.

While we're on the subject of alternatives, Duff Kurland shows us an alternative to the single-user, mini-disk TI-99/4 system with *An Introduction to the Marinchip Systems M9900.* This new family member is, of course, also TMS 9900-based, but uses 8-inch drives, has a multi-user operating system available, and is compatible with the S-100 bus. All in all, some versatile hardware and software for those of you needing a larger system.

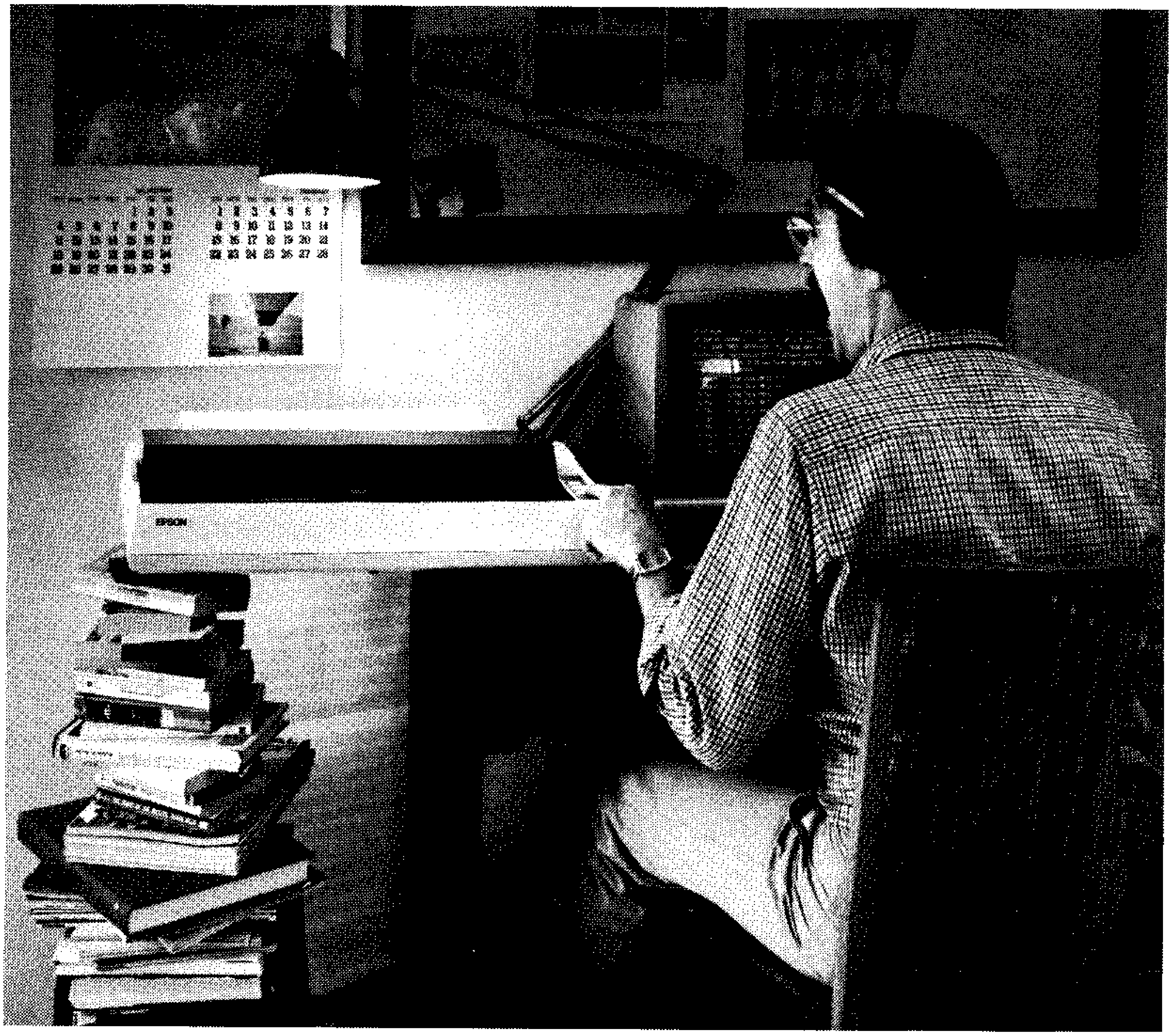
When we talk about a "new family member," we are referring to its use of the TMS9900 CPU or a slightly different version of the chip with the same basic instruction set. The University Board's TMS9980A falls in this family, so we continue our coverage of the board with *Bombs Away!*—a war game program that also makes use of a compatible TMS9918 Video Display Processor board. All you Extended Basic sprite lovers should find the program listing informative. If you should, however, have trouble understanding the accompanying assembly listing for the joystick, be sure to sit in on Dennis Thurlow's continuing class on *TMS9900 Machine & Assembly Language.*

Rounding out this July/August issue is our familiar "magazine within a magazine," *OnLoCAItion*—for 99'ers interested in using their computers for education. As seen from page 57's magazine cover, the 99'er Schools of Music, Engineering, and Typing are ready to help. But lest we concentrate solely on traditional CAI (computer-assisted instruction), we do have a couple of articles for the LOGO and Chess aficionados as well.

And finally, for any schools that need to use multiple micros but can't afford duplicate peripherals, Kathleen Swigger will reveal the North Texas State University solution.

I'd like to take this opportunity to congratulate Ralph Oliva, Alecia Helton and Diane Musha of Texas Instruments, as well as Niky and Susie Murphy, students at the Lamplighter School, for a fine performance on the *Mike Douglas Show* the week of July 6th. Eleven million TV viewers were treated to daily demonstrations of TI learning aids—including the Home Computer with TI LOGO on the last two days. On the last day, the show's guests, Captain and Tennille, were treated to a computerized rendition of their hit song, *Love Will Keep Us Together.* And as the show ended, Toni Tenille sang along with the TI-99/4 computer!

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



The MX-100. Not just better. Bigger.

Epson.

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

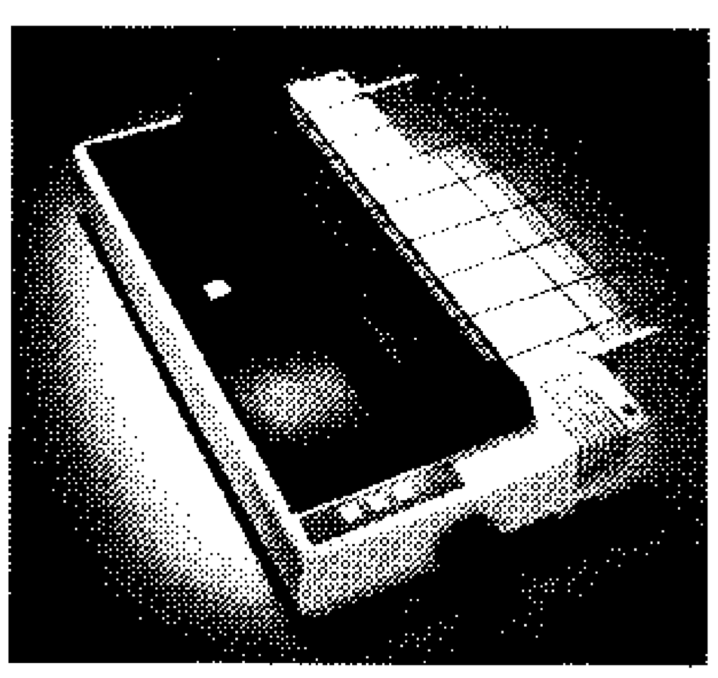
Frankly, it wasn't easy. But the results of all our sleepless nights will knock your socks off.

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

Needless to say, the specs on this machine — and especially at under $1000 — are practically unbelievable. But there's something about the MX-100 that goes far beyond just the specs; something about the way it all comes together, the attention to detail, the fit, the feel. Mere words fail us. But when you see an MX-100, you'll know what we mean.

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

How are we going to top this?



Your next printer.

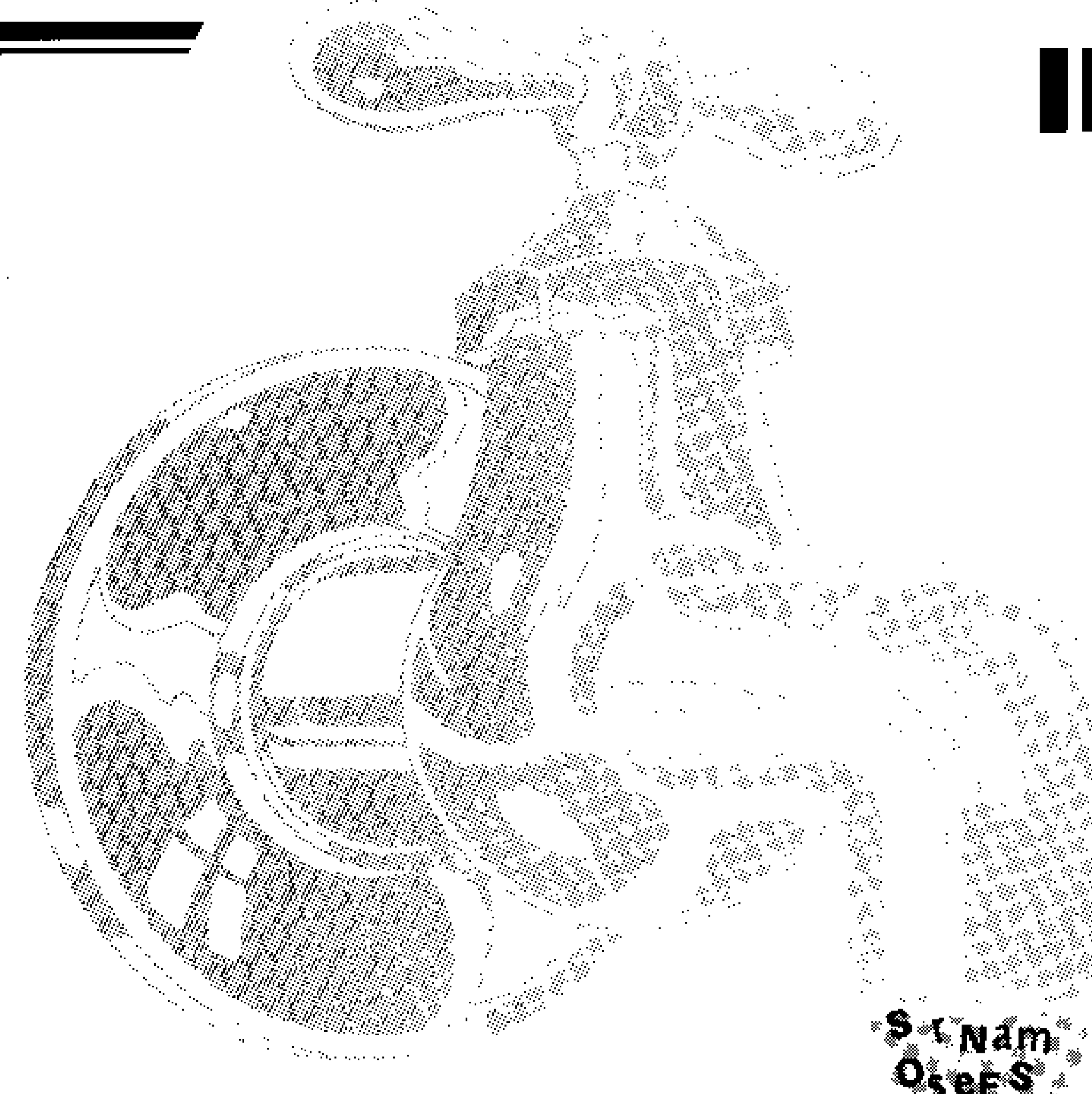
EPSON

EPSON AMERICA, INC.

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

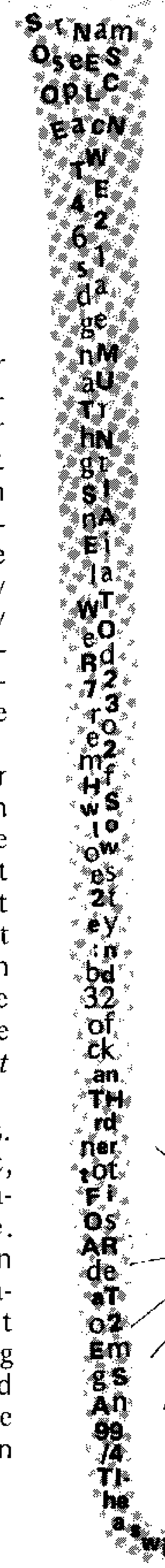
See the whole incredible Epson MX Series of printers at your Authorized Epson Dealer.

INFORMATION UTILITIES AND THE ELECTRONIC COTTAGE



" . . . we find more and more companies that can be described . . . as nothing but 'people huddled around a computer.' Put the computer in people's homes, and they no longer need to huddle."

—The Third Wave
By Alvin Toffler



By Gary M. Kaplan

In his recent book, *The Third Wave,* Alvin Toffler presents a powerful argument that " . . . our biggest factories and office towers may, within our lifetime, stand half empty . . . this is precisely what the new mode of production makes possible: a return to cottage industry on a newer, higher, electronic basis, and with it a new emphasis on the home as the center of society." Toffler goes on to single out many powerful socio-economic forces that are presently fueling this transition and points to the software production industry which has already set an early example as the fastest growing cottage industry of the 1980s.

Within the last year and a half, the microcomputer community has been witnessing the unfolding of an extraordinary event. I say "extraordinary" not because of what has already happened, but rather, for what it portends for the future. What is this event, and what great significance does it hold? Quite simply, the event has been the birth and maturation of "information utilities"—a significant event because of their awesome potential to speed up Toffler's timetable and change the way most of us live and work *within this current decade!*

There's certainly nothing mysterious about utilities. All of us are already familiar with telephone, electric, water and gas utilities. These are necessary and valuable resources delivered to and consumed in the home. If we now add *information* to this list, we create an "information utility"—a service that brings information to a place where the general public can access it and put it to use . . . and where the cost of packaging and delivery is *shared* by the consuming public. And what better, more convenient place is there for the general public to consume this information than in the home—the forthcoming "electronic cottage."

The New Timesharing

Timesharing, the foundation of all information utilities, is certainly not new. It was originally developed to serve the needs of business by providing companies with access to computer power without them having to buy expensive data processing equipment. Custom programming and technical assistance were available at extra cost to those who couldn't use the "canned programs."

What these information utilities have done is add a new wrinkle to the traditional timesharing concept. Using the famous "baking soda technique"—whereby a producer of this unglamorous age-old product continually dreams up and advertises new uses for it—they have repackaged timesharing to make it palatable to a much greater potential market. But lest you jump to the wrong conclusion, I should point out that these utilities are *not* simply pushing an old service to a new market. Rather, what we really have here is the creation of an *entirely new dimension* to timesharing—an attempt to satisfy a mass audience with extremely diverse needs and wants . . . and do it at an *affordable* price.

Information Services for the Masses

To provide you with some appreciation for the great diversity of presently available information services, let's take a brief look at one of the largest, fastest growing utilities, *The Source* (a service mark of Source Telecomputing Corporation, a subsidiary of The Reader's Digest Association, Inc.). At present, The Source offers over 1,200 services in areas such as:

(1) computer-based message services
(2) proprietary databases
(3) business and professional applications packages
(4) personal and corporate services
(5) consumer purchasing
(6) entertainment
(7) education
(8) "classical" timesharing

The E.Y.A.W.T.K.A.T.T.I.P.C.B.D.K.W.T.A.* Questionnaire

Filling in and mailing this questionnaire as soon as possible will help us get you the answers. Please participate. Thank you.

PART 1: THE TI-99/4 & TI-99/4A CONSOLES

(a) What would you like to know about the hardware?

(b) About the internal software & programming languages?

(c) What changes & new languages would you like to see? (Please include prices you'd expect new consoles & programming languages to cost.)

PART 2: THE PERIPHERALS

(a) What would you like to know about the operation, control, & interfacing of existing peripherals?

(b) What would you like to see forthcoming? (Please include prices you expect these to cost.)

PART 3: THE SOFTWARE

(a) What would you like to know about existing software (both TI & compatible non-TI)?

(b) What would you like to see forthcoming? (Please include prices you'd expect the packages to cost.)

PART 4: MISCELLANEOUS

(a) What would you like to know that wasn't covered in Parts 1-3?

(b) Why did you buy the Texas Instruments computer?

(c) What hardware/software (both TI & compatible non-TI) have you already purchased? (Exclude owner-written user-group programs.)

(d) What additional purchases do you expect to make?

(e) How is your computer used? Business? Education? Games? Other? Please explain.

(f) What is your occupation?

(g) What articles would you like to see in 99'er Magazine?

(h) Have you already found us at least one more subscriber? If not, please do so. Thank you.

* *Everything You Always Wanted To Know About The Texas Instruments Personal Computer But Didn't Know Who To Ask*

PLACE STAMP HERE

99'er magazine™

P.O. Box 5537

Eugene, OR 97405

PLACE STAMP HERE

99'er magazine™

P.O. Box 5537
Eugene, OR 97405

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)	☐ $15	☐ $18	☐ $25	☐ $40
2-yr (12 issues)	☐ $28	☐ $34		
3-yr (18 issues)	☐ $39	☐ $48		

Attention Dealers: Please write or call for bulk subscription rates. Also ask about our inexpensive magazine directory listings to publicize your retail locations. Call: (503)-485-8796

99'er BOOKSTORE

QTY	TITLE	PRICE	TOTAL AMOUNT
		POSTAGE	
		TOTAL	

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.

99'er-ware™

*U.S. ONLY—FOR SHIPMENT OUTSIDE THE U.S. INQUIRE SEPARATELY

QTY	ITEM	PRICE	TOTAL AMOUNT
	TI-SETTE ADAPTOR	$3.50	
	SHIPPING — 75¢ ea.*		
	DIGITAL COMPUTER CASSETTES in LIBRARY ALBUM	$9.95 or 3 for $27.00	
	SHIPPING — $2.00 for 1st ALBUM, 75¢ ea. add.*		
	MINI-FLEX DISKETTE FILE	$24.95	
	SHIPPING — $2.50 ea.*		
	FRICTION-FEED KIT FOR EPSON MX-80	$49.95	
	SHIPPING — $2.00 ea.*		
		TOTAL	

Back issues of *99'er Magazine* available — $3.95 ea. (postpaid) issue(s) desired ______ (month/month, year) ______ quantity TOTAL ______

Address shown is: ☐ Business ☐ Home

NAME ______________________ **PLEASE PRINT**

ADDRESS ______________________

CITY ______________ STATE ______ ZIP ______

99'er MAGAZINE SUBSCRIPTION	
TOTAL 99'er BOOKSTORE	
TOTAL 99'er-ware	
TOTAL	

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

Bill my: ☐ VISA ☐ Master Charge

Account No. ______

MC Only — List 4 digits above your name. ______

Expiration Date ______

Signature ______

All these services enter a subscriber's home or business through existing telephone lines (using the packet-switching networks of Telenet and Tymnet). A local toll-free 800 number is available in over 300 U. S. cities for accessing The Source. A subscriber types in (on a computer terminal connected to the telephone line, or a self-contained microcomputer with appropriate software to emulate a terminal) his or her private ID account number, and then chooses from a menu of services. Since subscribers can command the "host" computer in plain English (in a somewhat abbreviated form), very little instruction is necessary to do meaningful things—an extremely important attribute of any information utility.

Although an information utility such as The Source hopes, in the not-too-distant future, to be able to feed millions of inexpensive computer terminals in U. S. households, its present subscriber base of approximately 10,000 is drawn from the business community and a small segment of the vast consumer community—the segment which presently owns microcomputers.

It's not surprising why businesses of all types are attracted to very inexpensive services such as electronic mail, travel arrangements, applications software packages, programming access to mainframes, and business/industry news. It does, however, take some stronger incentives to lure the consumer segment of the microcomputer community—the present-day pioneers who purchased their micros for home use. It's to this group that information utilities like The Source must ultimately cater if they hope to eventually reach the economy of distribution and substantial return-on-investment that are possible in a mass market.

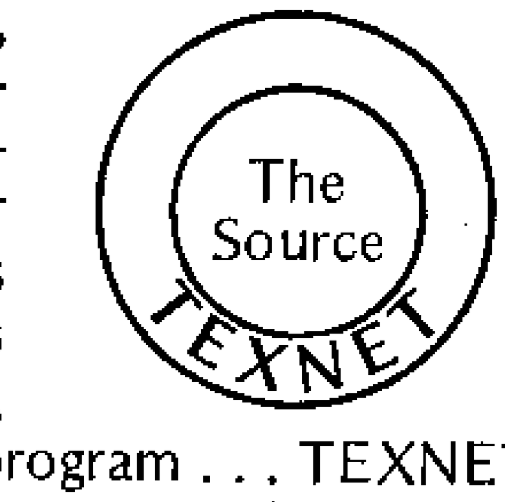
To this end, consumers with microcomputers are presently being wooed with a rapidly expanding array of personal services (such as bookkeeping, correspondence, travel arrangements, and keeping track of investments), educational programs, home economics assistance, plus activities and information that the *whole* family can use—especially games, movie and product reviews, news, and sports reports.

The TEXNET Turn-On

If having the services and activities of The Source in your home isn't exciting enough for you, how about having it together with the following package of special enhancements: color graphics and animation, music and sound effects, a software exchange with hundreds of free programs, plus state-of-the-art synthetic speech—with *all screen text actually "spoken" to you!* No, all this isn't just a "wouldn't-it-be-great-if" speculation of things to come, but rather, embellishments to the basic Source menu that will be available very shortly.

The special services and enhancements I've been describing are available to users of the Texas Instruments TI-99/4 and TI-99/4A microcomputers, and come under the TEXNET (a service mark of Texas Instruments, Inc.) umbrella. Besides the microcomputer, the only additional items that are needed to take advantage of *all* of the special TEXNET features are a plug-in RS232 Interface and modem (for establishing a compatible telephone connection), a plug-in Terminal Emulator II Command Module (the software for the microcomputer), and the plug-in Solid State Speech Synthesizer—the Texas Instruments peripheral that "voices" the synthetic speech. [See *99'er Magazine*, May/June 1981, p. 30]. The synthesizer won't be necessary if speech capability isn't desired.

Just how, exactly, are TEXNET and The Source related? According to Craig W. Vaughan (President, Software Sorcery, Inc.), a systems support consultant to Source Telecomputing Corporation and Texas Instruments, TEXNET *appears* to totally encompass The Source. That is to say, TEXNET subscribers have access to everything Source subscribers do, *plus* additional special services that require the Texas Instruments Home Computer for access and use. Graphically, it would appear like this, with the outer ring of TEXNET including everything within The Source's inner ring, and expanding its own outer ring of special services over time. This is only an *appearance*, however, as Vaughan pointed out: "In reality, TEXNET users will be running a shell program . . . TEXNET will eventually be a collection of programs running on The Source system."



Services on TEXNET fall into two major groups: (1) directory or lookup textual information, and (2) interactive or transfer services. In this first group there will be a product and technical newsletter (TI News), TI Software Directory, TI User Groups, TI Service Centers, and TI Phonetic Dictionary (helpful when programming with text-to-speech). The second group of services is really what TEXNET is all about. First, there are the *transfer* services. Sophisticated error-checking software in the Terminal Emulator II Command Module will permit any of hundreds of user programs from the TI Software Exchange to be downloaded correctly into another user's system. Eventually, we can expect to see on TEXNET the capability for *direct* uploading and downloading between users. The TI Graphics Library and TI Music & Sound Library will work the same way: A TEXNET subscriber will be able to download the color graphics, musical scores, and sound effects into his own system for later use in his own programs.

The *interactive* services on TEXNET are really speech enhancements of services already available on The Source. For example, the electronic mail service—probably the most highly used service, and reason enough for many to be or become Source subscribers—is made even more intriguing when your mail is "read" to you by your machine's electronic voice. And if "electronic voice mail" intrigues you, wait till you experience TI Voice Chat: TEXNET users will be able to participate in "spoken" interactive communication, CB-style. Well almost . . . What actually happens is that one user types in something, and the words get converted back to synthetic speech on the other end; the typed-in reply gets sent back, and then also gets converted to speech. So what we actually wind up with is a real-time verbal conversation *between two speech synthesizers!*

Is it affordable? What does all this cost? You be the judge: There's an initial one-time subscription fee of $100 for The Source/TEXNET registration, and usage fees of $2.75/hour (midnight to 7 a.m.), $4.25/hour (on weekends and on evenings from 6 p.m. to midnight), and $15/hour (weekdays from 7 a.m. to 6 p.m.) for prime-time business usage.

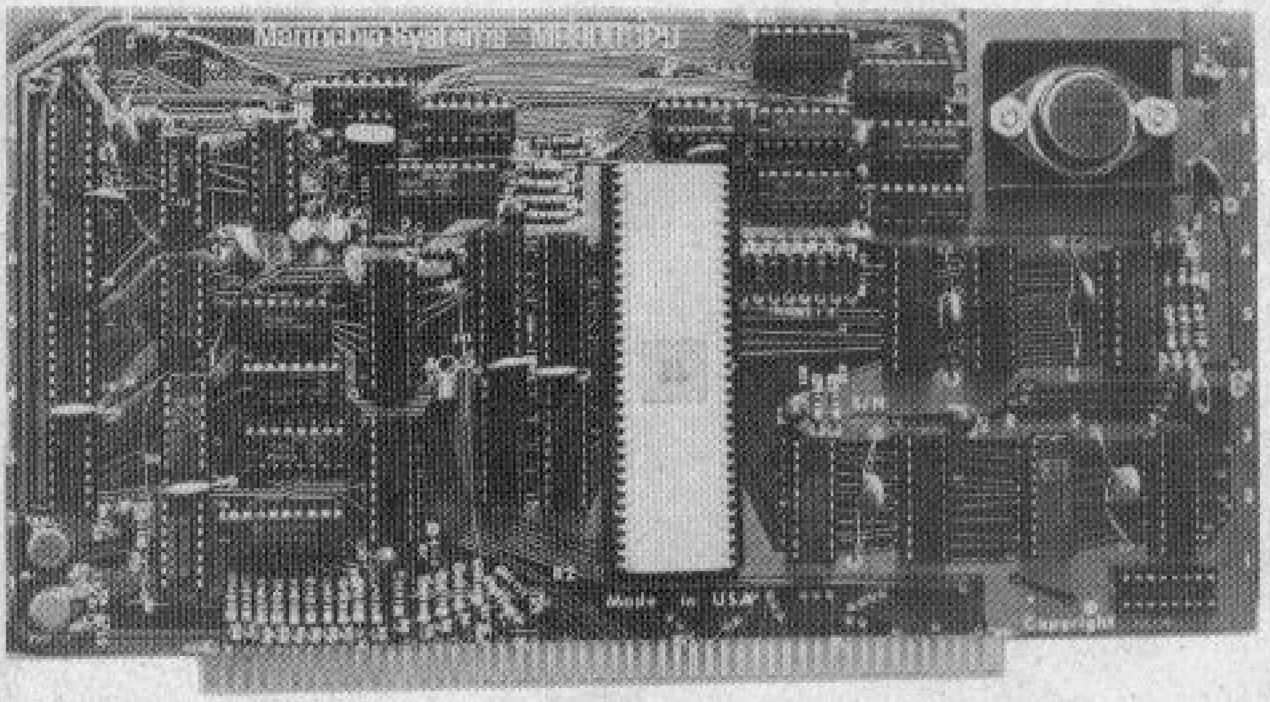
There's one short paragraph in the latest Source brochure that perfectly sums up what's presently happening in the world of information utilities:

> "*This brochure is obsolete. By the time you read this brochure, new information and communication services will have been added to The Source. Old data bases will have been updated, and streamlined "userfriendly" access procedures introduced.*"

Without a doubt, it's an exciting time to be living and learning along the new information frontier.

AN INTRODUCTION TO THE MARINCHIP SYSTEMS M9900

By Duff Kurland



The Marinchip System M9900 CPU Board.
Notice the large TMS9900 16-bit microprocessor in the center.

What? Another 16-bit TMS9900-based personal computer system? You mean the TI-99/4 actually has company out there in the predominently 8-bit micro world — a close relative who shares the same powerful instruction set and architecture in a landscape infested with the likes of Z80s, 6800s, and 6502s? Rejoice 99'ers. . . you are not alone. Welcome your California cousin, the Marinchip Systems M9900. Despite little publicity since its introduction in the spring of 1978, the M9900 has attracted many enthusiastic users. Marinchip's system is built around the 16-bit Texas Instruments TMS9900 microprocessor — the same chip used in the TI-99/4. Beyond that, however, the two computers differ dramatically. This article will serve as an introduction to the M9900 hardware and software. Future articles will explore specific programs and applications in greater depth.

We at *99'er Magazine* would like to take this opportunity to welcome M9900 users into our "99-er Family," and hope that you'll actively contribute by submitting your tips, suggestions, and articles to us. [And as TMS9900 assembly language programming becomes readily available to TI-99/4 users with a soon-to-be-released assembler Command Module, there will be more in common, and more resource sharing between the two user communities — Ed.]

About the Author
Duff Kurland is a senior systems programmer for a computer service bureau, and has had eleven years of experience on large Sperry Univac computers. Having chosen computers as his hobby as well, Duff has assisted in the development of some of the Marinchip Systems software.

The S100 Bus

Unlike the portable, self-contained TI-99/4, the M9900 system consists of several circuit boards which plug into slots in an S100 mainframe. A video terminal is usually employed as the system console, and since Marinchip's software relies heavily on disk storage, a pair of 8-inch floppy disk drives rounds out the basic system.

The S100 bus was the first widely accepted microcomputer bus standard, and sports a broad range of compatible circuit boards from a multitude of vendors. The M9900 thus offers the 16-bit power of the TMS9900 CPU chip, along with the flexibility of the S100 bus — a combination that's hard to beat.

Flexibility? Attach a letter-quality printer, and you've got a word processing system. Connect a high-speed dot matrix printer, and you've got a small business system. Or connect a digitizer and a color graphics display, and you've got an electronic or architectural design system. Or perhaps you're into computer music, speech synthesis, software development; household security, environmental control, or all of the above. It's quite conceivable for one S100 system to be equipped to do *all* these things and more.

With the TI-99/4 system, expansion is presently limited to 48K RAM, three 90K mini-disk drives, and two RS-232 ports. The M9900 system, however, has oodles of expansion room. You can have additional banks of 64K RAM for a multi-user system, have several serial peripherals connected at the same time, and even connect a multi-megabyte hard disk. But expandability isn't the only difference. Although both are based on the TMS9900 chip, the TI-99/4 and the M9900 have vastly different operating systems: Programs designed for one will *not* run on the other unless the differences in system calls and disk storage formats are taken into consideration, and conversion utilities provided.

Hardware

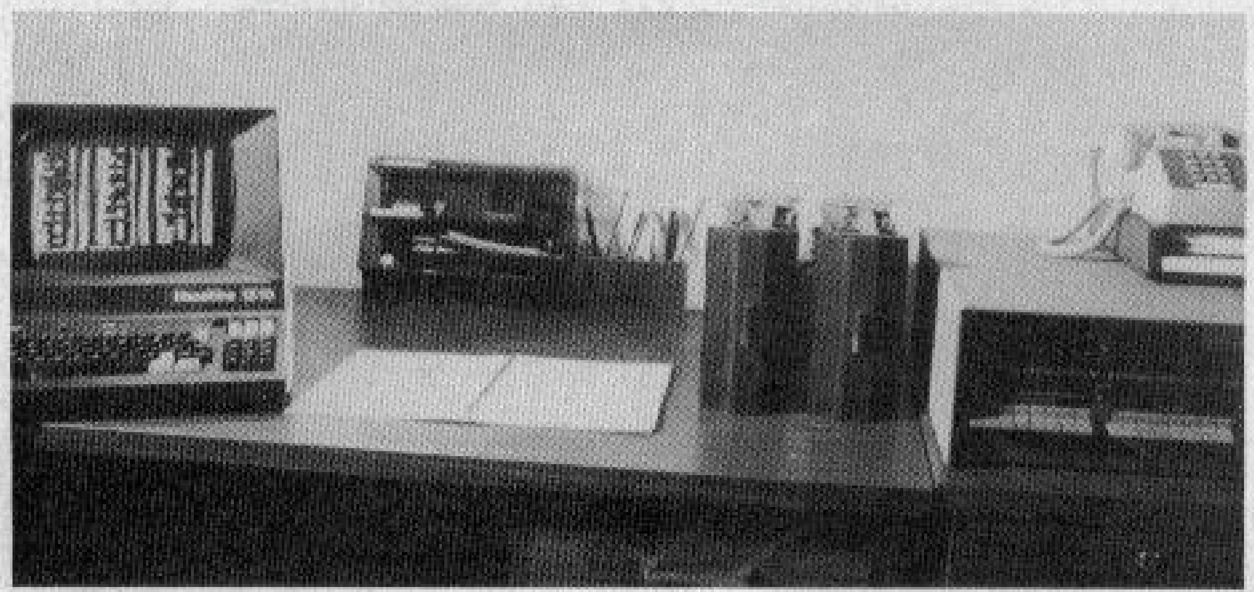
Marinchip Systems manufactures the boards which comprise the heart of the M9900 system, and recommends the products of various other vendors for other functions. A typical M9900 system includes the following Marinchip hardware:

- one M9900 CPU board
- one PROM/RAM board
- one or more 64K RAM boards
- QUAD SIO board (optional)

These boards are assembled, tested, and burned in at the factory, and are supplied with "Theory of Operation" manuals. Also, Marinchip is one of the few manufacturers to provide schematics for their hardware products.

To complete a basic system you'll need:

- a video or printer terminal
- two 8-inch floppy disk drives
- a disk controller board
- an S100 mainframe
- a printer (optional)



The author's system. The printer is not shown.

All items are available through Marinchip Systems, or can be purchased directly from other manufacturers or dealers. Let's take a quick look at each of the Marinchip circuit boards.

M9900 CPU Board

The M9900 CPU board connects the TMS9900 processor chip to the S100 bus, providing all the proper bus control signals. Circuitry to support Marinchip's 16-bit memory boards as well as most 8-bit S100 memory boards is found here, as is the memory mapping logic necessary to support S100-style I/O. (The 1K area from F000 to F3FF is used for this purpose.)

64K RAM Board

As mentioned above, the CPU board supports 8-bit and 16-bit memory. If you already have an S100-based system, you can probably use your existing memory boards with the M9900. Since most memory accesses made by the TMS9900 chip itself are for 16-bit words, logic on the CPU board will automatically make two accesses to adjacent bytes in order to read or write the desired 16-bit word. Use of 16-bit memory such as the Marinchip 64K RAM board, however, will avoid the need for this double memory access, and enable the system to run at least twice as fast.

The 64K RAM board provides 64K bytes organized as 32K 16-bit words. DIP switches permit any combination of the 16 4K blocks of memory to be disabled, to prevent conflicts with other memories or memory-mapped I/O devices. A bank switching capability is provided, permitting several 64K RAM boards to share the same addressing space. This technique is used in multi-user environments, where each user may be assigned his own 64K address space.

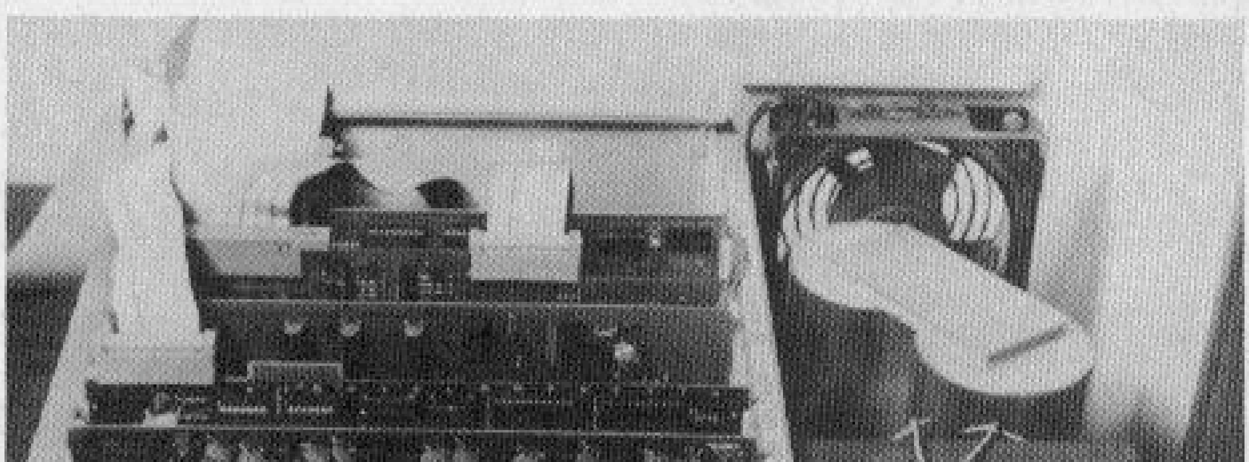
PROM/RAM Board

Marinchip's PROM/RAM board contains 1K bytes of 16-bit PROM (expandable to 32K), one serial I/O port, and a real-time clock. The 16-bit RAM may be used for the workspace registers, speeding up register access if your system has 8-bit memory boards. The PROMs contain the disk boot routine, and a debug monitor capable of dumping memory in several formats. The serial I/O port may be used for the console terminal, and the real-time clock is used by the NOS/MT multi-user operating system.

The debug monitor PROMs normally use the top 2K of the addressing range (F800 through FFFF), while the 16-bit RAM resides at F400 - F7FF.

QUAD SIO Board

Interfacing additional peripherals is easy with Marinchip's recent hardware offering, the QUAD SIO board. As its name implies, this board provides four serial I/O ports, each of which may be used to connect a printer, modem, or additional terminal.



A peek inside the mainframe. The first board is the M9900 CPU, followed by two 64K RAM boards and the PROM/RAM board.

Free Software

Marinchip Systems supplies a complete software package with the purchase (for $700) of their M9900 CPU board. This package includes the Disk Executive operating system, BASIC, an assembler, linker, line-oriented text editor, document formatter, and debug monitor. You also receive a host of utility programs. Much of the software is designed to operate with as little as 32K bytes of RAM in the system, but 64K is recommended.

Disk Executive

The Marinchip Disk Executive provides a simple floppy disk file system. File names and maximum size are chosen by the user, and the operating system keeps track of where the files are located on disk. All disk I/O is file relative, and the system detects any attempts to read or write outside file boundaries.

Input/output operations are hardware independent. All I/O devices are treated simply as files with special names. If you design a program to write to a disk file, you can direct it to write to a hardware device (such as a printer) without any program modification.

The operating system performs all memory allocation for user programs in an address space organized as shown in Figure 2. Programs may expand to fill all available memory automatically, and need not be regenerated when a hardware or software change results in a different user memory space.

Input typed on the system's console terminal is assembled in a buffer that is internal to the operating system, and delivered to the user program only after the carriage return has been entered. The system provides backspace capability for error correction, and special control keys for deleting a whole word at a time and for retyping the current input line if you've become confused. Inclusion of these important features in the operating system means that they need not be implemented in each and every program, and presents a consistent personality to the terminal user. A special system request permits a user program to take complete control of the console terminal in cases where character-by-character input is required.

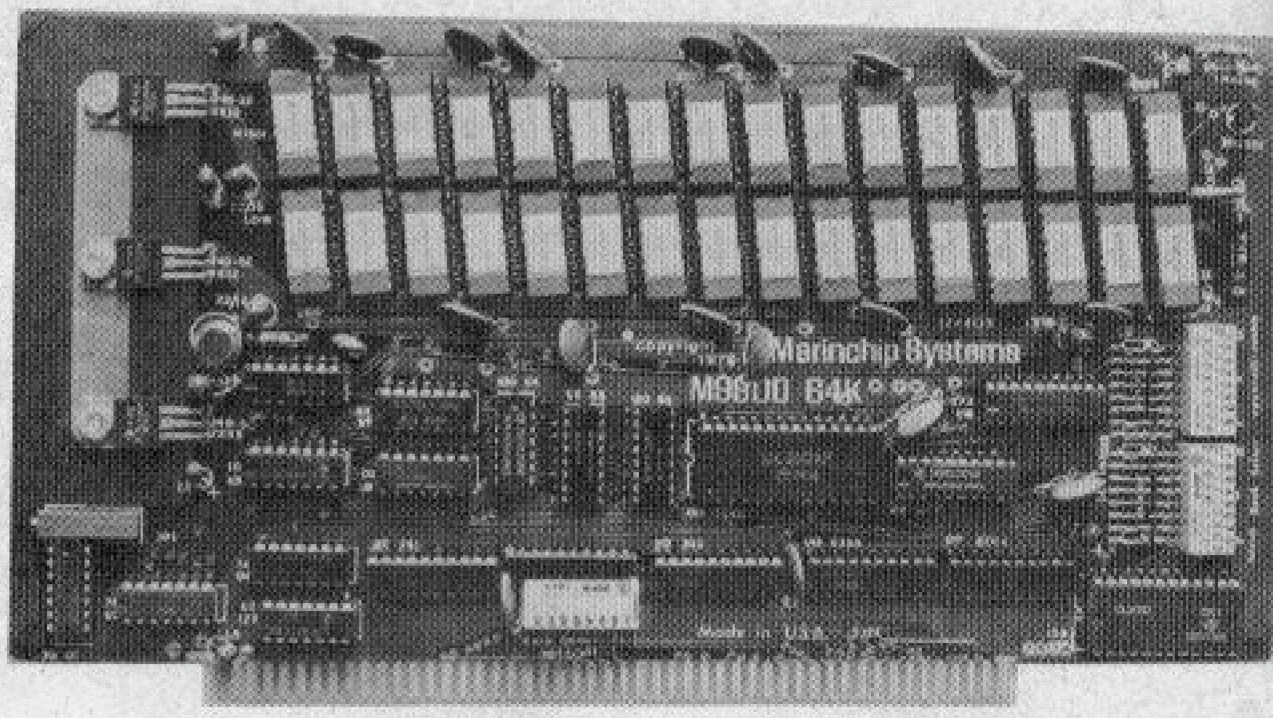
Programs running under the Disk Executive may issue system calls to request various services, or may call upon floating point arithmetic, output editing, buffer pool control, or linked list subroutines built into the system. These features make the Marinchip M9900 system one of the finest systems I've ever seen for the software developer. Disk Executive system calls are upward-compatible with the multi-user NOS/MT operating system, so that your investment in M9900 programs is not wasted if you upgrade to the fancier operating system.

The Disk Executive also provides an easy way to add new system commands. The set of system commands which you enter from the console terminal can be easily expanded by writing a program and giving it the desired command name. You then can execute the new system command by simply typing the name of the file in which the program resides.

BASIC

The version of BASIC included in the CPU board purchase price is very similar to Microsoft BASIC. It offers two-character variable names, IF-THEN-ELSE, multiple-statement lines, single-line functions, disk I/O capabilities, and a statement trace feature.

If you notice a program error while running a program, you may interrupt its execution and enter command mode. While the program is thus suspended, any PRINT statement you type will be executed immediately, so that you may examine the contents of program variables. This feature may also be used to provide a simple "desk calculator." You may list or modify portions of the program, change variable contents, and continue where you left off.



The M9900 16-bit 64K RAM Memory Board

Assembler and Linker

If you're like me, and enjoy developing system utility programs or just getting down to the bare nuts and bolts, an assembler and linker will be essential tools. Well, these are also included free with the M9900 CPU. The assembler produces relocatable object code, and the linker can pull several relocatables together to make an executable program. This allows you to develop a large, complex program in neat byte-sized chunks.

The assembler offers a few extra features worth noting. IF-ELSE-ENDF directives allow you to selectively assemble portions of a program, based on values attached to program labels. A COPY directive lets you place common definitions in a disk file, and easily include them in any programs that need them. Also, the M9900 assembler recognizes several special instructions used by the Marinchip system. For instance, the system call feature is implemented using the TMS9900's extended operation facility (XOP). But, rather than saying "XOP 1" whenever you want the system to do something for you, you may say "JSYS" (meaning "jump to system").

Text Editor and Document Formatter

The line-oriented text editor (EDIT), and the document formatting program (WORD) comprise the final major elements of the basic software package included with the M9900 CPU board. Various file listers and copiers, file directory routines, the debug monitor, and diagnostic utilities round out the package.

Contents of IMSAI Mainframe

- M9900 CPU board
- M9900 PROM/RAM board
- 2 M9900 64K RAM boards
- Teletek FDC-II disk controller
- IMSAI SI02-2 serial I/O board
- Dual Systems Control Corp CLK-24 clock board, with battery backup

Hazeltine 1510 video terminal

2 GSI 110-B 8-inch floppy disk drives

NEC Spinwriter letter-quality printer

Vadic 1200-baud modem

Figure 1 - The Author's System Components

0000	Standard workspace register area
0100	User address area begins here
.	
.	
----	User address area ends here, and
.	operating system address area begins.
.	The exact address is dependent upon the
.	hardware and software configuration.
.	On my system, it's around D100, leaving
.	about 52K bytes for user programs.
.	
EFFF	End of operating system address area
F000	1K input/output memory map area
F400	PROM/RAM board RAM area (1K)
F800	PROM/RAM board PROM area (2K)

Figure 2 - Disc Executive Memory Layout

EDIT allows you to enter a program or save data in a disk file, or to correct, modify, or simply look at the text of an existing disk file. It is "line-oriented" in that the editor "looks" at one line of the file at a time. You can tell EDIT to move up or down in the file by using line numbers, or by searching for a particular string of characters. Once positioned at a line, you may use several commands to change the text on that line (or from that line for a few following lines, or for the rest of the file).

EDIT's most outstanding feature is its file paging capability. You may safely edit a file whose text is much too lengthy to fit in memory all at once: EDIT will page portions of the file out to temporary disk files, and will automatically bring them back into memory when needed.

WORD, the document formatter, reads text files from disk, formats the text as directed by user commands inserted in the text, and writes the formatted version to a disk file or printer. Facilities are provided to handle form letter preparation, program documentation, and many other word processing chores. You control the formatting parameters by using numeric and string variables, macros, and library files.

I often use WORD to document the software I've written. Its automatic page and section numbering, table of contents generation, and heading/footing macro features make documentation a simple — even enjoyable — process.

File Format Commonality

It is important to note that most of the M9900 software was designed and implemented by one person, and that all the programs read and write text files compatibly with one another. Thus, a BASIC program could write a text file which is later edited using EDIT, then processed by WORD and written to a printer. The very article you are reading was written using the WINDOW screen editor. Its spelling was then checked using SPELL, and then was processed by WORD to produce the printed manuscript.

Text file compatibility seems like a simple enough idea, but don't take it for granted! There are numerous systems on the market whose Pascals, BASICs, and word processors all have different file formats.

Additional Software

The free software package described above is all you need to do software development in assembly language, simple BASIC, or to do word processing. However, Marinchip Systems also sells additional language compilers, operating systems, business packages, and other "application" programs.

An Extended Commercial BASIC provides all the features of the standard BASIC, plus PRINT USING, greater floating point precision, a program CHAIN capability, and random-access disk I/O.

Marinchip's Pascal is an implementation of Per Brinch Hansen's Sequential Pascal, and has no connection whatsoever with UCSD Pascal. All the standard structured programming and data structure definition capabilities are provided, but the I/O is nonstandard. The compiler makes seven passes over the program, and produces pseudo-code. Sequential Pascal is well suited to large system development: the compiler itself is written in Sequential Pascal, yet can be compiled without difficulty in a 64K M9900 system! Each program, however, must include a "prefix" defining the system interface. If you need formatted output, you must also provide a set of output editing procedures. For these reasons, as well as the fact that the pseudo-code must be interpreted at run time, you may find QBASIC more suitable for program development.

QBASIC is a language compiler which was actually written by an M9900 user. It is a compiler which accepts programs written in CBASIC-2 (A trademark of Compiler Systems) and produces actual TMS9900 machine code. The resulting program runs much faster than any interpreted BASIC or Pascal program. Many of the recent Marinchip utility programs, such as WINDOW and SPELL, were written in QBASIC.

Just in case you're not familiar with CBASIC-2, its major attributes are long variable names, structured WHILE-WEND constructs, formatted output (PRINT USING), and the nearly total elimination of BASIC's annoying line numbers. QBASIC supports separate compilation of program segments, access to assembly language routines by simply LINKing them in, and several extensions to the CBASIC-2 language.

Marinchip offers a second text editor called WINDOW. This is a screen-oriented editor which turns your terminal into a "window" looking into the text file being edited. You can move the window around in the file vertically (or even horizontally, if the file has lines longer than the width of your terminal's screen).

Control characters (or function keys, if available on your terminal) are used to move the cursor around on the screen and perform the desired editing tasks. A single keystroke puts WINDOW into a special command mode, allowing you to set various modes or request more complicated functions such as moving a block of text from one place to another. Like EDIT, WINDOW is capable of editing a file whose text far exceeds the amount of user memory.

I'm a horrible typist. Maybe you don't spel so gud. Both of us can benefit from Marinchip's SPELL program. SPELL looks up each word of a document in its dictionary, and reports any word that it hasn't heard of. Once you've weeded out the obvious abbreviations and acronyms, the remaining words are probably spelling or typing errors. Utility programs are provided to let you add new words to the dictionary.

A Multi-Tasking Network Operating System (NOS/MT) is offered as an alternative to the Disk Executive. This UNIX-like system can support several users on the same mainframe. Each user has his own terminal and his own memory area (up to 60K each), but all users share the same CPU, printer, disks, and other hardware. To add another user to the system, the only hardware that's needed is another terminal, an I/O port for it, and, perhaps, another memory board. A system generation procedure must be followed to customize the system for your particular hardware and software needs.

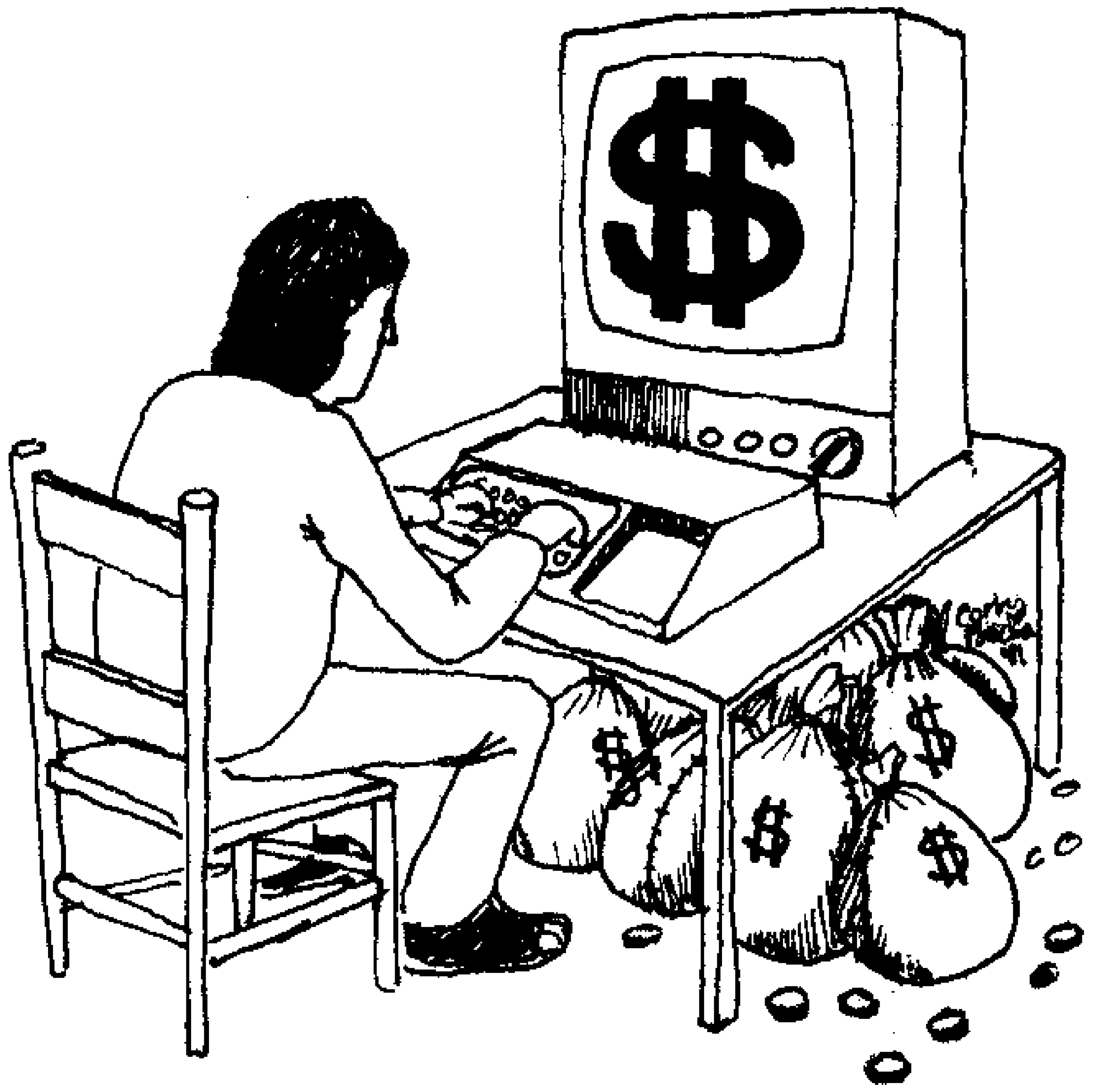
NOS/MT provides all the features of the simpler Disk Executive, plus print output spooling, disk file simulation in memory, background batch capability, dynamic file space allocation up to 4 billion bytes, directory files (hierarchical directory), hard disk drive support, a system clock, and numerous additional features.

If you're looking for off-the-shelf business software to run on the M9900, you're in luck. The folks at Marinchip have converted the well-known Osborne commercial packages (General Ledger, Accounts Payable and Receivable, and Payroll) to QBASIC. These programs are usable singly or as an integrated system. Additionally, there's a computer-aided drafting package called INTERACT, a terminal simulation program for communicating with a remote computer system, and some more exotic languages — FORTH and META — for aficionados of threaded structure and stack operations.

In future articles, we'll take an in-depth look at some of this software.

A Mini Editorial

Now that *99'er Magazine* is bringing together all formerly isolated TMS9900 family members, we would hope to see hardware and software producers adapting their products for these additional related markets. The software from Marinchip systems is a case in point: If TI-99/4 users, for example, would like to see an extremely fast compiled BASIC (e.g., Q-BASIC) available on their fully-configured TI system, the folks at Marinchip should be made aware of this by the interested users—Ed.



The Small Investor & the **TI-99/4**

A LOOK AT THE DOW JONES NEWS SERVICE

By Lawrence Riley

Information utilities such as The SOURCE and MicroNet allow any individual with a microcomputer and modem to tap into a rich vein of information resources. These databases, however, are aimed almost exclusively toward the general consumer population, and as such, cannot adequately cover the needs of serious, small investors. That's where the Dow Jones News Service (DJNS) comes in : the combination of the DJNS and The TI-99/4 may be the most significant advance in investment analysis since the electronic calculator made its debut . . .

Historical stock quotes are available, as are current-day quotes for all listed stocks, bonds, options and U.S. Treasury issues.

For a comprehensive review of a stock or industry, the "Media General" database provides detailed technical and fundamental indicators on the item of your choice.

The conservative investor can access the "Disclosure On-line" database for a profile on most major companies, plus a "10-K" report that lists almost all the important (to the investor) information that can be found in a corporation's financial statement.

"The Money Market Service" database is a new service introduced by Dow Jones in February of this year. Commentary, tables and graphs on the economy are displayed for most of the important indicators used in determining the current business climate. Of course, the ever-popular Dow Jones averages are also available, as are Trading Activity, The Market Diary, Market Volume, and many other valuable market statistics.

With everything there comes a price tag, and the news service is no exception. During the business day (6:00 a.m. to 7:00 P.M. EST) the charge for news is $1 per minute. After 7:00, this rate is reduced 80% ! Until the next morning, news can be accessed for 20 cents per minute, and historical market quotes for 15 cents. The start-up fee for the service is $50, but there are no monthly charges or minimum on-line times. For high-volume users there is pricing option "A". Under this option, there is a $50 monthly fee in exchange for lower prime-time rates during the business day. Pricing option "B" should be satisfactory for most individual investors.

9756 Hadley Drive, Overland Park, KS 66212

In my day-to-day use of the Dow Jones News Service, I have found no problems in accessing the system with my TI-99/4. When using the Terminal Emulator 1 (TE-1) module, the keyboard does occassionally lock up. But I have found that pressing the Shift-C key while simultaneously holding the Shift-V key down, will bring the cursor back. After discovering this trick, I haven't lost any data due to TE-1 lock-up.

After news has been obtained on the News Service, there are really only two things that can be done with it: (1) it can be kept temporarily, or (2) kept permanently. News that is to be kept temporarily is best stored on a disk or printed copy for ease of access and readability. When keeping news permanently, cassette tapes, especially if bought in volume, can be both cost effective and reasonably efficient.

For aspects of the service other than news, there are many different ways to use both the historical and current quote database. The historical quotes are available in either monthly or quarterly format for any given item. While a weekly format would be desirable, the monthly quotes can be used to determine most long and intermediate term trends. For the very short term, one month of daily quotes are always available. These can be used to develop a 10, 15, or 20 day moving average of prices for the item being researched, and if saved over a period of time, can be used in any format.

For the novice investor, the Media General Data Base provides a sufficient amount of both technical and fundamental analysis. *Fundamental Analysis* refers to information concerning the aspects of a particular company or industry such as assets, net worth, or earnings. *Technical Analysis* refers to the study of the chart or graph of a company, industry, or the market in general—in the hope that past behavior as revealed in graphs can be used to predict future price movements.

The serious investor may prefer to develop his or her own analytical tools. One current theory on Wall Street today is that about half of a stock's performance is due to movement of the market in general, and about half of the movement is due to characteristics peculiar to that particular stock. Naturally, anyone that can predict the movement of the market, even for a short time, has a very powerful financial tool.

For this reason, my own analysis tends toward analyzing the leading market indices. This analysis can be facilitated by the use of the TI *Personal Record Keeping* Command Module (PRK). Each page can be set up to represent one day, and the first few lines can label the index to be tracked. The remaining lines can be the 10, 15, or 20 day averages of the aforementioned indices. The use of math transformations in the PRK module allows the average to be computed for each of the indices, but the average must be entered manually with the "change page" option. One by-product of the average that is computed automatically by the PRK module is the standard deviation. I have found this statistic to be a good indicator of market volatility. It too can be entered and tracked with the average. The ability of the PRK module data to be analyzed by the *Statistics* Command Module is a definite plus for analysis. Even though the *Statistics* module is a more sophisticated analytical device, and offers more tools to work with than the PRK module, I do not feel that it is essential to index analysis—only helpful.

Investors with access to a TI-59 programmable calculator as well as a TI-99/4 can perform some rather astounding mathematical computations without a strong math background. Quotes can be obtained through the News Service, and used in a "Least Squares Curve Fit" program detailed in the Texas Instruments publication: *Sourcebook for Programmable Calculators.* This will result in a series of simultaneous equations which can be solved either with the *Master Library-2* program on the TI-59, or with the *Math Library* program on the TI-99/4. In theory, the resulting equation should be a reasonably accurate description of the line the datapoints were taken from, and can be used to predict the future behavior of the line. Naturally, the number and quality of the datapoints chosen determine the accuracy of the predictive equation, and any conclusion drawn from such analysis is at best, highly speculative.

Fundamental analysis using the TI-99/4 also has many applications. Balance sheet and income statement analysis can be programmed, and then compared to an "ideal" or average analysis to determine the variances in an effort to discover the strengths or weakness of a particular company or industry. The information for these analyses can be found in the 10-K section of the Disclosure On-Line Database of the News Service.

Of course, these are only a few of the applications that can be put to use with the TI-99/4 and the Dow Jones News Service. In the past, this mathematical analysis of the market and it's component stocks were inaccessible or simply incomprehensible to the small investor. But now with the help of your TI-99/4, a sophisticated approach is both available and easy to use.

In closing, I would recommend that any investor with a TI-99/4 computer call Dow Jones on their toll free number (800-257-5114 except N.J.) to request their free information packet detailing prices and services.

Good luck 99'ers. If this works for you, your only problem may be writing a suitable income tax program!

FREE COMPUTER INFORMATION NETWORK FOR 99'ers

NOW AVAILABLE IN 16 STATES AND 2 PROVINCES

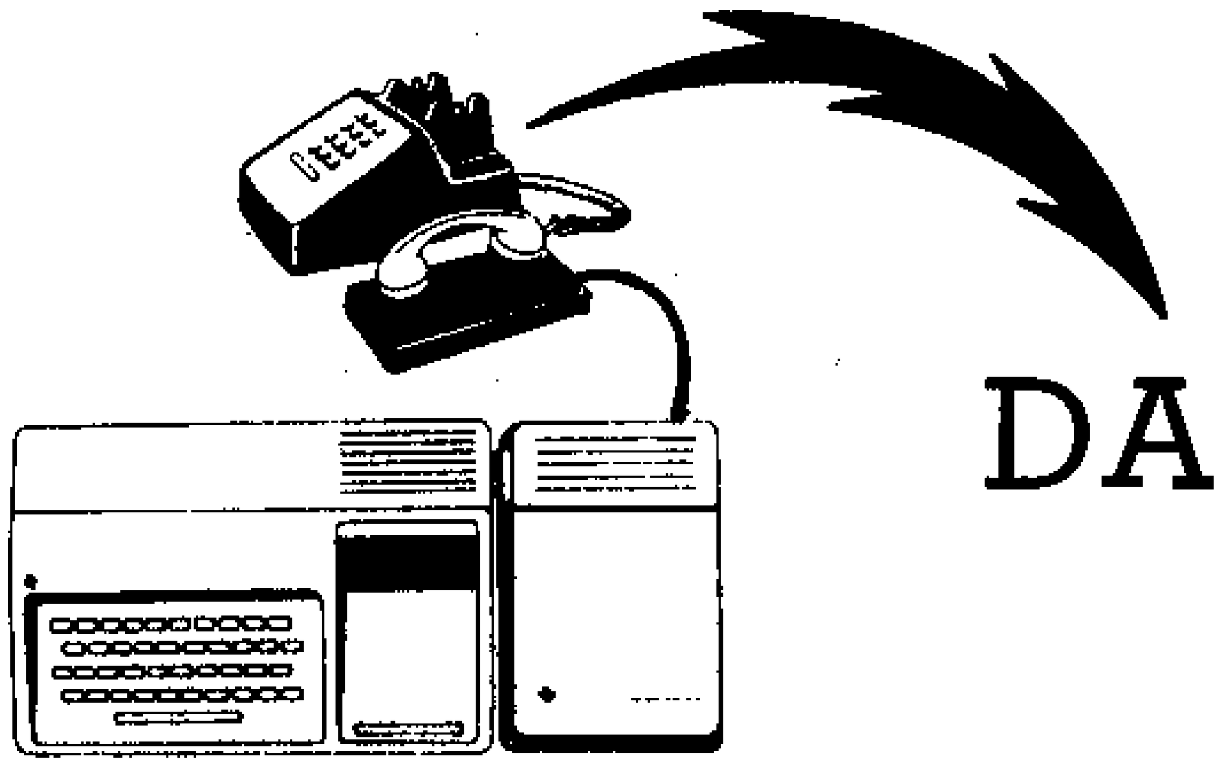
Over three dozen computers with specialized information in such subjects as astronomy, education, amateur radio and games can be accessed free of charge through any computer (equipped with terminal-emulation software) or terminal, by using an ordinary home or business phone and an electronic conversion device called a "modem" (a contraction of MOdulator/DEMdulator—a device that changes digital computer data signals into analog signals sent over the telephone lines.

All computers are free to use and most do not even have time limits. The only costs are for long distance or toll line phone charges. Almost all of them are also available evenings or weekends when phone rates are lower. All operational costs are paid for by the owner/operators (user groups, schools, publishers, commercial businesses, and dedicated hobbyists). Most systems don't request donations, although some present commercial messages. Users also have the opportunity of placing any type of information into these computer files—thus forming a national computer bulletin board.

Novation, Inc., a modem manufacturer, offers a free dial-up directory with up-to-date listings of computers available by modem to any computer or terminal user. Call 213-881-6880, 24 hours a day. When the welcome message appears . . . LOGON PLEASE, type in the word CAT and press the Carriage Return or ENTER key on your computer or terminal. The video screen will display an 18 item menu that contains product information, a glossary of computer terms, and a modem /printer test. Item 18 is the directory of free dial-up computers that is updated each month. Many of the listings are only for Apple or TRS80 users (indicated by "ABBS" and "FOR80". All other listings are accessible with any other computer that is emulating a terminal through a 300 baud asynchronous modem that is compatible with Bell System 100 series specifications

CBBS Wash. DC	(202) 281-2125 (10, 2)
CBBS Wash. DC,	(202) 337-4694 (10)
CBBS Seattle	(206) 723-3282
CBBS Seattle	(206) 246-8983
CBBS New York	(212) 787-5520 (14)
CBBS New York	(212) 997-2186 (7)
CBBS New York	(212) 933-9459
CBBS New York	(212) 245-4363 (10)
CBBS Los Angeles	(213) 954-8582
CBBS Los Angeles	(213) 291-9314
CBBS Los Angeles	(213) 826-0325
CBBS Los Angeles	(213) 881-6880 (10)
(Novation Directory) LOGON with word . . .CAT	
CBBS Inglewood	(213) 673-2206
CBBS Philadelphia	(215) 563-0674
CBBS Macon County IL	(217) 429-5505
CBBS Arlington Hts	(312) 255-6489
CBBS Chicago	(312) 767-0202 (10)
CBBS Chicago	(312) 545-8086 (10)
CBBS Detroit	(313) 288-0335 (10)
CBBS ST Louis	(314) 781-1308
CBBS ST Louis	(314) 227-8495
CBBS Atlanta	(404) 394-4220 (10)
CBBS Atlanta	(404) 939-1520 (5, 10)
CBBS Portlan	(503) 641-8555
CBBS Portlan	(503) 641-9029
CBBS Long Island	(516) 939-9043
CBBS Phoenix	(602) 957-9282
CBBS Vancouver, BC	(604) 687-2640 (10)
CBBS Ottawa	(613) 725-2243
CBBS Nashville	(615) 254-9193 (10)
CBBS Cambridge	(617) 864-3819 (10)
CBBS Las Vegas	(702) 454-3417
CBBS Alexandria VA	(703) 620-4990 (10)
CBBS Orange county	(714) 751-1422 (10)
CBBS Fullerton	(714) 526-3687
CBBS Santee CA	(714) 449-5689 (10)
CBBS Vermont	(802) 879-4981 (7)
CBBS Sacramento	(916) 393-4459 (12)

Codes:
2 = Amateur Radio
5 = Games
7 = Education
10 = 24 hour operation
14 = Astronomy



DATA COMMUNICATIONS & the TI-99/4

By F. T. Berkey

If you have invested in an RS232 interface and a modem in addition to your TI-99/4 system, you have the possibility of tapping a vast information network through existing and planned computer time-sharing services. A variety of information services such as news, financial information, computer games, various data bases, and program exchange, to name just a few, are provided through information utilities such as *The Source* (by Source Telecomputing Corporation) and *MicroNET* (by CompuServe). TEXNET, a collaboration between Source Telecomputing Corporation and Texas Instruments, will enhance data base services with the addition of text-to-speech, color graphics, and music. This service will be available exclusively to users of the TI-99/4. Since it will be some time, however, before the prospect becomes fully implemented [see *Information Utilities & the Electronic Cottage* in this issue], we'll start this series of articles with an examination of basic data communications between the TI-99/4 and other computers.

Data Communications Concepts

A number of coding schemes have been devised to represent characters in order to input information into a computer. The most widely used code is the American Standard Code for Information Interchange—more commonly known as ASCII code. It is a 7-bit code which can represent 128 character configurations. Figure 1 illustrates the bit patterns associated with each of the characters. An eighth bit, called a parity bit, is commonly included in the ASCII code. The parity bit is used to detect errors in the bit stream which might be due to the reading or transmission of the data. Parity of a ASCII coded signal can be odd or even. An ASCII code with even parity must contain an even number of ones; for odd parity the number of ones must be odd (i.e., 1, 3, 5, 7). Examples of other codes are the 5-bit Baudot code, the 6-bit Binary Coded Decimal (BCD) code and the 8-bit Extended Binary Coded Decimal Interchange Code (EBCDIC). These codes use different bit patterns to represent characters than does the ASCII code. The Texas Instruments Terminal Emulator 1 (TE-1) Command Module enables you to tailor your TI-99/4 to fit the characteristics of the remote computer system. With the communications device menu, you can specify the parity of the received or transmitted signal—odd (default), even, or none (no parity bit)—and set the number of data bits at 7 (default) or 8.

The actual number of bits transmitted is larger than the number of bits in the code. "Housekeeping" bits are added

MOST SIGNIFICANT OCTAL DIGIT	LEAST SIGNIFICANT OCTAL DIGIT 000	001	010	011	100	101	110	111
0 000	NUL	SOH	STX	ETX	EOT	WRU	RU	BEL
0 001	BS	HT	LF	VT	FF	CR	SO	SI
0 010	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETC
0 011	CAN	EM	SUB	ESC	FS	GS	RS	US
0 100	SP	!	"	#	$	%	&	'
0 101	(	)	*	+	,	-	.	/
0 110	0	1	2	3	4	5	6	7
0 111	8	9	:	;	<	=	>	?
1 000	@	A	B	C	D	E	F	G
1 001	H	I	J	K	L	M	N	O
1 010	P	Q	R	S	T	U	V	W
1 011	X	Y	Z	[	\	]	↑	←
1 100	_	a	b	c	d	e	f	g
1 101	h	i	j	k	l	m	n	o
1 110	p	q	r	s	t	u	v	w
1 111	x	y	z	{	|	}	~	DEL

NUL	Null or tape feed(control-shift P)
SOH	Start of heading (control A)
STX	Start of text (control B)
ETX	End of text (control C)
EOT	End of transmission(control D)
WRU	Enquiry (control E)
RU	Acknowledge (control F)
BEL	Ring bell (control G)
BS	Backspace (control H)
HT	Horizontal tab (control I)
VT	Vertical tab (control J)
FF	Form feed (control L)
CR	Carriage return (control M)
SO	Shift out (control N)
SI	Shift in (control O)
DLE	Data link escape (control P)
DC1	Device control 1 (control Q)
DC2	Device control 2 (control R)
DC3	Device control 3 (control S)
DC4	Device control 4 (control T)
NAK	Negative acknowledge(control U)
SYN	Synchronous idle (control V)
ETC	End of transmission block (control W)
CAN	Cancel (control X)
EM	End of medium (control Y)
SUB	Substitute (control Z)
ESC	Escape (control-shift K)
FS	File separator (control-shift L)
GS	Group separator (control-shift M)
RS	Record separator (control-shift N)
US	Unit separator (control-shift O)
SP	Space
DEL	Delete, rub out.

Figure 1.

both before and after the bits which represent the character code. The additional bits are called start and stop bits. A single start bit is added at the front of the code as a signal to advise the receiving device to start sampling the incoming signal. Stop bits, added after the character code, indicate when the code is finished, and reset the device for recognition of the next start bit. For an ASCII coded signal, 11 or 12 bits are typically transmitted (see Figure 2).

In data communications terminology, a *full duplex* channel implies that information can flow in two directions simultaneously. On a *half duplex* channel, the information can flow in both directions, but not simultaneously. If you select the half duplex mode from the TE-1 communications device menu, (and set the modem accordingly), the characters you send will be "echoed" back to your monitor or TV set, and appear on the screen. The echoed or extra character does not occur if full duplex is selected.

The public telephone network provides means of communication from your TI-99/4 to another computer or information service. The information or bit stream that your computer sends and receives, travels serially through the network. That is to say that the bits making up a character are sent and received one after another. Serial transmission is not the only mode in which data can be transferred. Inside your TI-99/4, data bits are transmitted in parallel; all eight or sixteen bits are transferred simultaneously from point A to point B. Parallel transmission can also be used to transfer information between computers, however, it is more complex to do so and therefore more expensive than serial transmission.

There are a variety of modes of data transmission. Your modem transmits data *asynchronously*. This means that each character is sent independently of any other character, and that the data bits are preceded by a start bit and followed by at least one stop bit. *Synchronous* transmission requires that both the sending and receiving modems are synchronized by a clock signal. The rate at which data is transmitted (or received) is termed the *baud rate*. The formal definition of a baud is that it is the reciprocal of the length of the shortest pulse used to create a character. Since all the bits of the ASCII code are equal in length, the terms "bits per second" and "baud" can be used interchangeably. A baud rate of 110 requires a minimum of 2 stop bits; at 300 baud a minimum of 1 is required. The TE-1 software allows you to choose between three baud rates (110, 300 or 600), but your modem limits your use to either 110 or 300. The RS232 interface also allows you to use baud rates of 1200, 2400, 4800 or 9600. The higher rates can be used to output data to a printer or to send data to another TI-99/4 connected directly to your system.

The function of your modem is to convert the binary pulse train (1s and 0s) from your computer to some form of analog signal (tones) that can be transmitted over a telephone line. You will note that in the transmit mode your modem emits a continuous tone. This tone is called the carrier signal. When sending data from your TI-99/4, the modem's function is to *modulate* (vary the amplitude or frequency) this carrier signal. It also works in the opposite sense by *demodulating* the carrier, so that the ASCII code sent to your TI-99/4 can be properly interpreted. Thus, the term "modem" is derived from the two words which describe its function: MOdulation and DEModulation. A common modulation technique is called frequency shift keying (FSK). This technique converts the binary pulses from the computer to two tones of different frequency. For example, if the carrier signal has a frequency of 1500 Hz, a 1 would be transmitted at 2000 Hz and a 0 at 1000 Hz.

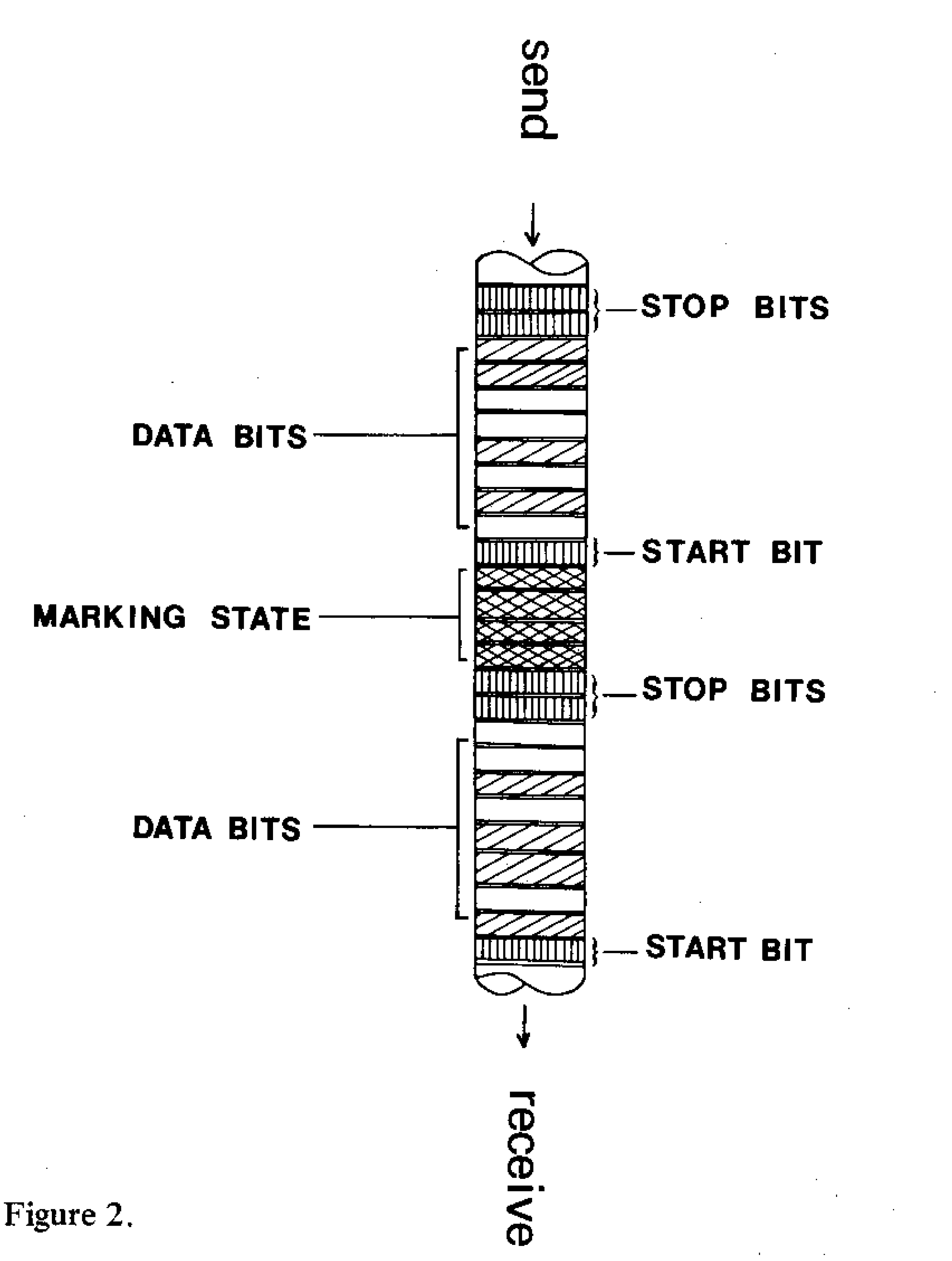


Figure 2.

Terminal Emulator 1 Command Module

The TE-1 Command Module implements all 128 characters of the standard ASCII code which is illustrated in Figure 1. The TI-99/4 keyboard is not encoded with the lower case character set, but lowercase can be easily invoked by depressing SHIFT V, then 1. It's also possible to send any standard ASCII control characters (used for signaling a remote computer or device to perform a predefined function), and display lines containing more than 40 text or program characters by "wrapping" the extra characters onto a second line. The most powerful feature of the TE-1 is the ability it gives users to store received data on tape or disk. You can review this data after logging off the remote computer, and can also send it to a printer or another computer.

Data Communications Using Basic Language Programs

The format of the data stored by the TE-1 is ASCII (display format) and is of variable record length with a maximum of 192 bytes (characters). In order to make further use of the information, it is necessary to write programs using BASIC. A simple example of such a program is shown in Figure 3. Line 130 opens a saved disk file using the OPEN statement. The following line inputs an ASCII character string; if the record denotes the end of file (EOF), the program ends. Otherwise the number of characters is found using the length function (LEN) statement. A new character string is created in line 170 using the string segment function (SEG$) where the last position of the character string is defined as N, the number of characters in the string. The character string is displayed on the monitor or TV, and the program returns to the INPUT statement (line 140) and continues to read the data file until an EOF is detected.

Often data retrieved from another computer or information service contains lower case ASCII characters. Since the TI-99/4 BASIC recognizes only upper case characters, the program of Figure 4 will insert the patterns stored in memory at the locations corresponding to decimal codes 97-122. Using the CHAR subprogram, the lower case characters can be defined as codes 97-122. For example, a lowercase A is given the character code 97 and the pattern identifier "0000300838483C00." The program listing in Figure 4 adds the lower case characters to the program listed in Figure 3. These patterns can be modified or improved by using the character definition program listed on pages 186 and 187 of the *User's Reference Guide* (Note that line 510 of that program in early printings of the book contains an error: B(R,2)*4 should read B(R,6)*4), or the lowercase subroutine in TI's *Programming Aids 1*.

Display formatted files can be sent from your TI-99/4 to another computer under control of the BASIC listing shown in Figure 5. The program assigns file number 1 to the indicated disk filename, and file number 2 to port 1 of the RS232 interface. Each record or character string is input from the disk, displayed on the monitor, and then transmitted to the remote computer. Of course, this assumes that a means of recording this data is resident on the remote computer. This program could be used, for example, to efficiently transmit a pre-recorded message or text file to another home computer.

The program listings in Figures 3, 4 and 5 have a common flaw; if the display file being read contains commas, the character string will be terminated by the first comma encountered. This is due to the fact that BASIC interprets a comma as a separator between character strings or data items in display formatted data. (See page 158 of the *User's Reference Guide.*) This flaw can be overcome only if the file is created from BASIC and the procedure to do so is discussed on page 159 of the *User's Reference Guide.*

BASIC programs can also be transmitted to another computer through the use of the LIST command. After reading the program into the RAM memory of the TI-99/4 using the OLD command, the command LIST "RS232" will send the program listing through port 1 of the RS232 interface to the modem or other device connected to the interface.

In future articles, we'll examine other aspects of data communications, profile various communication services, and review the new Terminal Emulator 2 software along with its exciting reason for existence—TEXNET. Reader input is welcomed.

References

Fitzgerald, J. and T. J. Eason, *Fundamentals of Data Communications,* J. Willey and Sons, 1978.

Healey, Martin, *Minicomputers and Microprocessors,* Hodder and Stoughton, 1976.

Terminal Emulator 1 Manual, Texas Instruments 1980.

RS232 Interface Manual, Texas Instruments 1979.

User's Reference Guide, Texas Instruments 1979.


```
100 REM   ** PROGRAM DECODE **
110 REM  READS IN A DISPLAY DATA FILE SAVED WITH THE TERMINAL EMULATOR AND OUTP
UTS IT TO THE MONITOR
120 INPUT "ENTER FILE NAME: ":FN$
130 OPEN #10:"DSK1."&FN$,VARIABLE 192,DISPLAY
140 INPUT #10:X$
150 IF EOF(10)THEN 200
160 N=LEN(X$)
170 M$=SEG$(X$,2,N)
180 PRINT M$
190 GOTO 140
200 END
```

Figure 3.

```
100 REM   ** PROGRAM DECODE **
105 REM  READS IN A DISPLAY DATA FILE SAVED WITH THE TERMINAL EMULATOR AND OUTPU
TS ITTO THE MONITOR
110 CALL CHAR(97,"0000300838483C00")
115 CALL CHAR(98,"0020203824243800")
120 CALL CHAR(99,"0000182420241800")
125 CALL CHAR(100,"0008083848483800")
130 CALL CHAR(101,"000018243C201C00")
135 CALL CHAR(102,"3028207020202000")
140 CALL CHAR(103,"00001C24241C0418")
145 CALL CHAR(104,"0020203824242400")
150 CALL CHAR(105,"0800180808081C00")
155 CALL CHAR(106,"0800080808281800")
160 CALL CHAR(107,"0020242830282400")
165 CALL CHAR(108,"0018080808081C00")
170 CALL CHAR(109,"00001C2A2A2A2200")
175 CALL CHAR(110,"0000382424242400")
180 CALL CHAR(111,"0000182424241800")
185 CALL CHAR(112,"0000382438202000")
190 CALL CHAR(113,"00001C241C040400")
195 CALL CHAR(114,"0000283420202000")
200 CALL CHAR(115,"0000182010083000")
205 CALL CHAR(116,"0010381010140800")
210 CALL CHAR(117,"0000242424241800")
215 CALL CHAR(118,"0000222222140800")
220 CALL CHAR(119,"0000222A2A2A1400")
225 CALL CHAR(120,"0000221408142200")
230 CALL CHAR(121,"0000221408080800")
235 CALL CHAR(122,"0000380810203800")
240 INPUT "ENTER FILE NAME: ":FN$
245 OPEN #10:"DSK1."&FN$,VARIABLE 192,DISPLAY
250 INPUT #10:X$
255 IF EOF(10)THEN 280
260 N=LEN(X$)
265 M$=SEG$(X$,2,N)
270 PRINT M$
275 GOTO 250
280 END
```

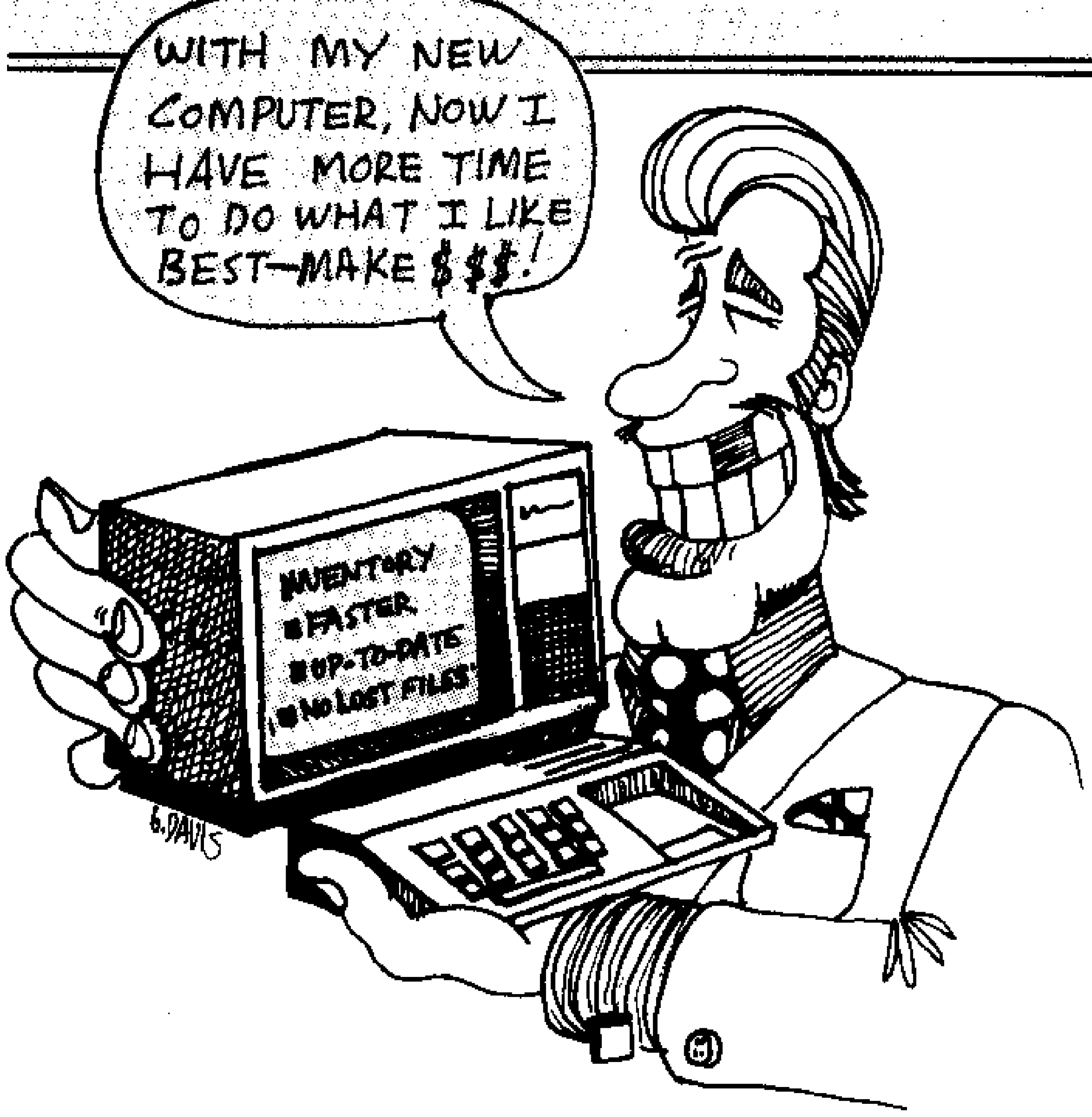
Figure 4.

Note 1:
The problem of using commas (and quotation marks) with the INPUT statement, as discussed in this article, has been remedied by the LINPUT statement of Extended BASIC. With LINPUT, there is no editing of what is input.

Note 2:
The new TI-99/4A console has a lowercase character set already built in.

```
100 REM  ** PROGRAM PRIDISPLAY **
110 REM   **TRANSMIT DISPLAY FILES TO A PDP 11/70**
120 CALL CLEAR
130 INPUT "ENTER FILENAME:":FN$
140 OPEN #1:"DSK1."&FN$,VARIABLE 192,DISPLAY
150 OPEN #2:"RS232.PA=N.DA=8.EC.TW.NU",DISPLAY ,VARIABLE,OUTPUT
160 INPUT #1:X$
170 N=LEN(X$)
180 M$=SEG$(X$,2,N)
190 NREC=NREC+1
200 PRINT NREC;N;M$
210 PRINT #2:M$
220 IF EOF(1)THEN 240
230 GOTO 160
240 END
```

Figure 5.



Getting Down to Business

By George Struble

Evaluating A Software Package

In last issue's column I defined two categories of computer applications for business: (1) **planning**—concerned mostly with projections, and not having to be done at particular moments at peril to a business; and (2) **integrated use**—applications such as invoices, accounts payable and receivable, mailing list maintenance, general ledger, inventory, or any of many others upon which a business crucially depends at particular times. In this article, we'll explore some of the implications of integrated use.

Programs for integrated use are likely to be rather extensive. After all, most such applications involve organization and management of significant quantities of data. This means that the programs must help you with the data entry, help you monitor the validity and correctness of the data, and help you update the data. The programs must also be able to retrieve data for processing, summarization, and answering inquiries. Depending on the application, the programs may also have to generate auditable controls and provide tax reports.

The programs for an integrated use application must be well-designed and form what we would call an **information system.** To develop such a system takes a substantial amount of work—probably several *months,* if not *years,* of programmer time. If your application is small enough for you to think about doing it on a TI-99/4 or other micro, it would be quite a mismatch of investment for you to pay for even six months of a programmer's time to develop a system. Therefore, you will want to buy a system that is already developed, packaged, and ready to install and use. You actually have a better chance of getting a good working product by buying a package, than by having it done to your specifications by a programmer.

OK, you're in the market for a package. Besides cost, the most obvious criterion is whether a proposed package will meet your needs. Now is the time —even before seeing the details of a proposed package—to make yourself a checklist of the features you want your package to include. List each processing action that you think would be necessary in your system. Consider the data elements you think would have to be stored and related to each other in order to provide the information you will need at any moment. If done in a detailed and comprehensive way, this would be close to what we would call a **systems analysis** of your application.

Great detail and comprehensiveness are not needed; the idea is to give you a starting point for judging the adequacy of a package you may be offered. You will probably find that a particular package is organized differently and does its activities differently from your outline. There's nothing wrong with that. Concentrate on the *results* produced and whether they are appropriate: Does the proposed package provide the information you consider essential? Then, of course, you can also judge whether the proposed package is convenient or awkward, and flexible or rigid.

A second suggestion is to talk to other users of the proposed package, and get their opinions of the package's strengths and weaknesses. You may be surprised how willing other users are to share their experiences. Even if you have to phone a couple of users long-distance, it will be well worth the trouble and cost.

You should not expect your needs in an information system to always remain the same. Your business changes; auditors make new demands; federal or state regulations change. This is where flexibility of a system comes in. Chances are, that there will come a time when you will want your system to do something it was not designed to do. Then, you will need help in modifying the system. The supplier of the package is in the best position to know how to modify your system. But will he be around when you need him? Find out whether the *source* program is supplied and accessible to you. If it is, then you have a chance of getting someone near you to modify it when needed. Try to find out from the supplier and users how much trouble a minor modification would be. You may not be absolutely able to trust an answer you get, but a judgement of modifiability is a hard problem, and this is the best suggestion I can make.

In my next article I will review some business-related software. This will provide an opportunity for some more specific suggestions in the analysis of a package.

Now let us turn our attention to something more tangible—a program

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.*

that should be of practical use to many of you.

Effective Interest Rate or Return On Investment

Suppose you have an opportunity to buy an investment for $1500. The investment is expected to pay $140 at the end of each of the next five years, and at the end of five years return a lump sum of $2000. What is the effective interest rate or total yield on this investment? Or, put another way, what is the return on this investment? This problem can be stated in terms of capital in your business: If you invest some amount in a certain piece of equipment or in a higher level of inventory or . . . , you expect some estimated improvement in revenues. What is your expected return on this investment?

Since you have many opportunities and a limited amount of capital, you need to compare the expected rates of return on each of several opportunities in order to be able to make the best decision. Of course, there are usually intangible benefits too, as well as variations in the risks of different investments. A return on investment calculation is, therefore, not the only—or necessarily the deciding—criterion in your decisions. Nevertheless, it will certainly provide valuable input to your decision-making process.

The program presented here is a relatively simple one. I define a **component** of the investment as one or more payments of equal amount made at regular intervals. An investment will have two or more components; they are the main input to the program. Each component is described by:

(a) the amount of each payment (there may be only one).
(b) the time at which the first of these payments is made. Time is measured in months from the current moment, which is understood to be time zero.
(c) the number of months between payments. This is irrelevant if there is only one payment in a component, but we require a number anyway.
(d) the number of payments in this component.

For instance, the example above includes three components:

	(a)	(b)	(c)	(d)
1st Component	1500	0	1	1
2nd Component	– 140	12	12	5
3rd Component	–2000	60	1	1

Note that the investment amount is given as a positive number, but the **returns** on the investment are given as negative numbers. The second component represents the five annual payments (12 months apart) starting 12 months after the current time. The first and third components represent single payments: the initial payment and the final payoff after five years (60 months).

The program makes provision for up to ten components; the number of components is the first input the program asks for.

The program strategy is to compute the residual present value at an interest rate higher and one lower than the effective interest rate. We use an interpolation formula to produce a better estimate to the effective interest rate, then narrow the range of possible effective interest rates, and repeat the process. The program stops when the residual value is less than some fraction of the total of the numbers used in computing the residual value, or when the range of possible effective interest rates is less than some tolerance. There are four parameters set in statements 200-230 of the program that you may want to change, depending on your requirements:

U9 = starting upper bound for effective interest rate, set now at 30%.
L9 = starting lower bound for effective interest rate, set now at 0%.
T9 = tolerance for range of effective interest rate, set now at .05%. When the possible range is less than this, we conclude you have the rate closely enough.
P9 = tolerance for residual present value, set now at .0001. Because of round-off error during the calculations, this tolerance should not be reduced much below this value.

Figure 1 shows a transcript of the execution of the program with the sample data given above.

Note that the program uses a subroutine starting at line 720; a parameter R is supplied to the subroutine, and parameters V and V3 are returned. If you have Extended BASIC, you can make these parameters explicit in the subroutine call. You can also rephrase some of the control structures using IF-THEN-ELSE and multi-line statements, and make the program much more readable. I leave this for you to explore.

Figure 1

```
ENTER NUMBER OF PAYMENT
COMPONENTS? 3
ENTER AMOUNT OF PAYMENT? 1500
ENTER TIME OF FIRST OF THESE
PAYMENTS? 0
ENTER PERIOD BETWEEN THESE
PAYMENTS, IN MONTHS? 1
ENTER NUMBER OF THESE
PAYMENTS? 1

ENTER AMOUNT OF PAYMENT? -140
ENTER TIME OF FIRST OF THESE
PAYMENTS? 12
ENTER PERIOD BETWEEN THESE
PAYMENTS, IN MONTHS? 12
ENTER NUMBER OF THESE
PAYMENTS? 5

ENTER AMOUNT OF PAYMENT? -2000
ENTER TIME OF FIRST OF THESE
PAYMENTS? 60
ENTER PERIOD BETWEEN THESE
PAYMENTS, IN MONTHS? 1
ENTER NUMBER OF THESE
PAYMENTS? 1

RESIDUAL PRESENT VALUE AT
  0% IS -1200
RESIDUAL PRESENT VALUE AT
  30% IS 731.7656652
RESIDUAL PRESENT VALUE AT
  18.63580073% IS 290.8235145
RESIDUAL PRESENT VALUE AT
  15.00040794% IS 93.29345296
RESIDUAL PRESENT VALUE AT
  13.91833345% IS 27.69506322
RESIDUAL PRESENT VALUE AT
  13.60435554% IS 8.02691232
RESIDUAL PRESENT VALUE AT
  13.5139594% IS 2.310160891
RESIDUAL PRESENT VALUE AT
  13.48799321% IS .6635205027
RESIDUAL PRESENT VALUE AT
  13.48053936% IS .1904640003
EFFECTIVE INTEREST RATE,
COMPOUNDED MONTHLY, IS
  13.48053936
```

Lease vs. Purchase Analysis

Quite complex programs are available to do an analysis of whether leasing or purchasing some piece of equipment is more advantageous. The effective interest rate program can be used for a lease vs. purchase analysis, though it requires you to do some side calculation. One way to do the analysis would be essentially to calculate the return on *purchasing* the equipment and *leasing* it back to someone else. You would include the

- cost of purchase (+)
- tax benefits from claimed depreciation (–)
- lease payments (–)
- maintenance cost, if maintenance is provided under the lease (+)
- any difference in insurance or other costs between purchasing and leasing (+ or –)
- expected cost of purchase at the end of lease period (–) *or* trade-in value at the end of lease period (–)

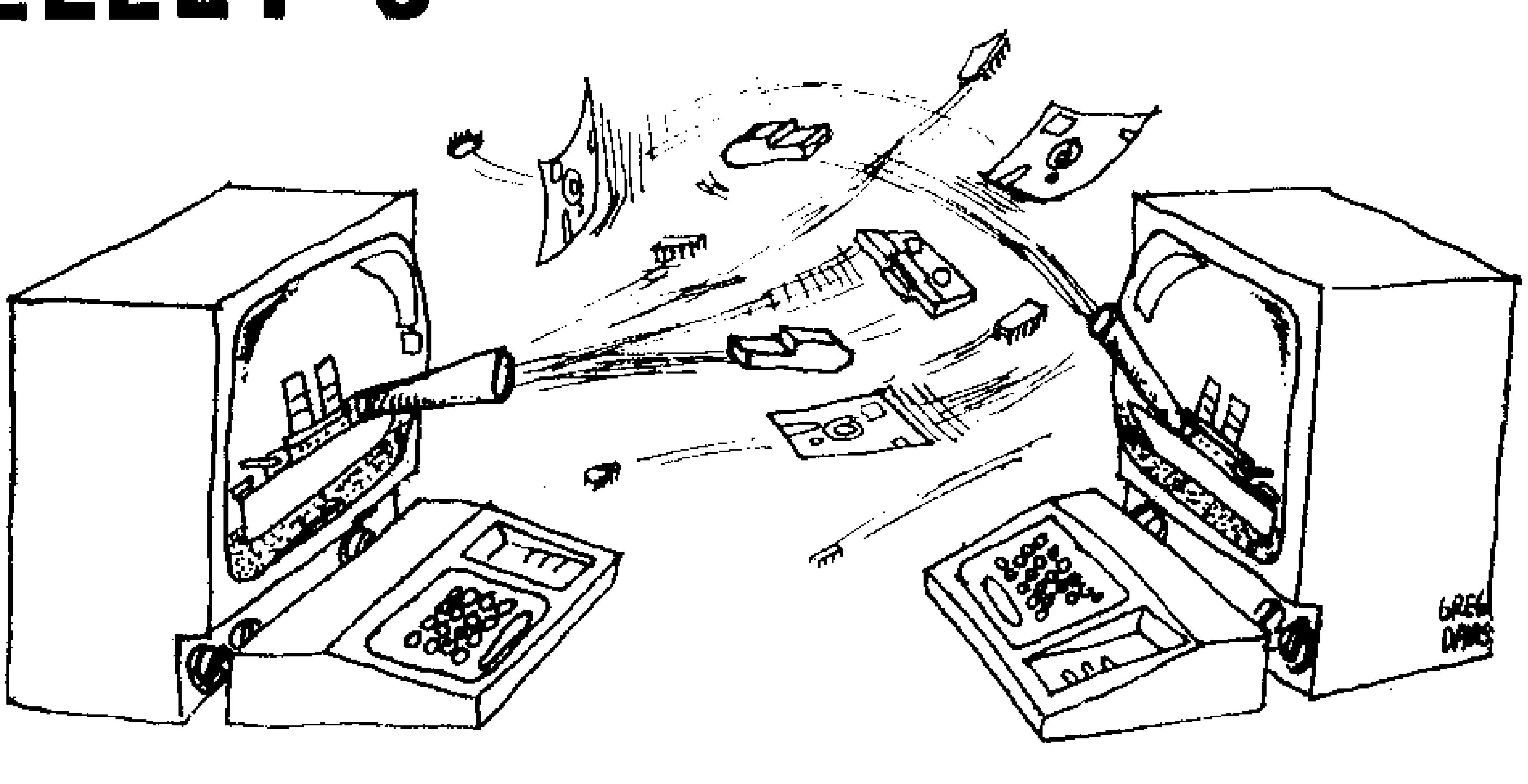
The rate of return indicated by this analysis can be compared with your borrowing cost, and the comparison would give you an indication of whether purchase or lease would be more advantageous to you.

As a small example, suppose you are going to get a widget-grinder. You can buy it for $12,000, or lease it for three years at $300 per month. No maintenance is involved, and the insurance cost

Continued on p. 75

KELLEY'S KORNER

CATCH & MATCH GAMES in TI BASIC & Extended BASIC



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

For this issue, the official travel brochure says that I'm supposed to lead you on an excursion into the world of "static pattern matching" and "dynamic coincidence." But 'tween you and me, what we're really gonna do is hatch a batch of "catch & match" games.

So get ready to battle your computer for supremacy of the seven seas, help the harried housewife handle her housework, and sprint after those spectacular shifting sprites. It's time to begin . . .

Battle Stations ! Battle Stations !

Battle At Sea
By W.K. Balthrop
99'er Magazine

Harried Housewife
By Regena
99'er Magazine

Sprite Chase
By Ron Binkowski
428 Adams
Dolton, IL 60419

Battle At Sea
By W.K. Balthrop

"Damn the torpedoes! Full speed ahead . . . " Get ready, all you "armchair admirals" out there in 99'erland. You're about to do battle with the most crafty enemy of all—the Imperial TI Fleet. If you're old enough to remember those rainy Saturdays in the pre-TV age, you've probably spent many an hour with pencil and paper playing *Battleship*. In the intervening years, *Battleship* has been dressed up as a consumer item in many forms: First it was "cardboardized," then "plasticized," and finally "electronicized." Well gang, as it happened, one rainy Saturday afternoon a few months ago, I had this mad urge to play *Battleship* . . . The expensive electronic version looked really enticing in a local toy store display, but I sure wasn't going to spring for it—especially when I had my trusty TI-99/4 personal computer waiting to carry out my every command. So program it I did. The result: *Battleship* has now been "99'erized" into a 16K TI BASIC version, which I call *Battle At Sea.*

Two 10 x 10 grids are displayed on the screen along with the row and column designations. The computer will ask you to enter coordinates for the placement of each of your ships on the grid at the right. Each coordinate must be entered separately; for example A 5 then A 6 for the destroyer. Since the ships occupy different numbers of grid squares, I've put in a counter for each ship to indicate how many remaining squares that must be entered.

After all the coordinates for a ship have been entered, that ship will be displayed on the screen. Once all five ships are set up, the computer will secretly set up its own ships on the grid to the left. You won't be able to see the computer's ships, since the whole idea of the game is to try to find them.

Once the computer has set up its ships, it will ask you for the coordinates of your shot at its grid (on the left). You must enter your shot as a row letter, then a column number. Valid coordinates are from A-J and from 0-9. Any other entry will result in having to enter the coordinates again. Your hit or miss will be marked on the grid and displayed at the bottom of the screen as a MISS or **HIT**. The computer will then take a shot at your grid. It cannot see your ships, but it does keep track of where the hits and misses are.

After a hit, any ship that has been sunk will be displayed at the bottom of the screen. The score is also updated at this time: one point for each ship sunk. The first player to sink all five ships will win the game.

Because there are no moving objects in this game, speed was not the most important factor in the game design. The action happens to be fairly fast, but the critical factor was programming the computer to make intelligent decisions. With no limit on available memory, I might have been able to write a program with flawless logic. But here that wasn't the case—with having to stay within the confines of standard 16K TI BASIC.

I started by giving the computer a set of rules and several variables to test for a given situation. First, if a ship has been hit only once, the computer will take random shots around that hit until the direction is determined. It will then continue in that direction until either the ship has been sunk, or it misses a shot, or it runs up to the edge of the grid. It will then reverse and shoot at the other end if the ship was not sunk. If you put ships adjacent to one another, the computer sometimes gets "confused" and gets caught in an endless loop. Therefore, keep *your* ships separated by at least one square in both horizontal and vertical directions (its ships can be adjacent) if you want to give the Imperial TI Fleet a fighting chance.

EXPLANATION OF THE PROGRAM
Battle At Sea

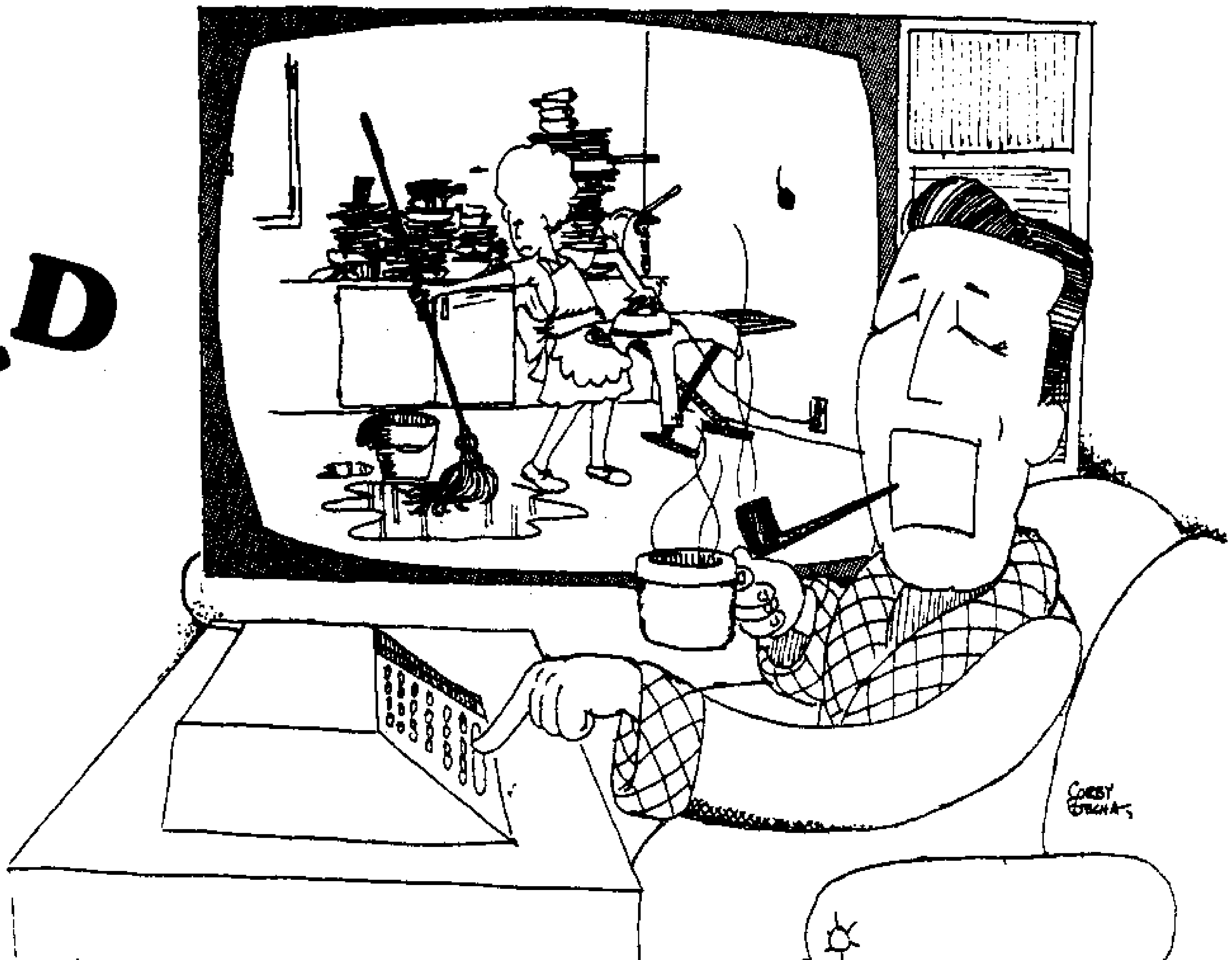
Line Nos.	
100-630	Initialization: Set up variables, character definition, and color assignments.
640-870	Instruction page.
880-1010	Display two 10 x 10 grids.
1020-1100	Control loop for setting up your ships on the 10 x 10 grid.
1110-1360	Subroutines holding data on each ship.
1370-1380	Branch to subroutine: computer sets up its ships.
1390-1530	Display message for ship coordinates to be entered.
1540-1710	Read keyboard; INPUT coordinates of ships.
1720-1950	Put the coordinates in order.
1960-2050	Check that all coordinates are valid.
2060-2220	Display ship on the 10 x 10 grid.
2230-2380	Control loop holding data for computer to set up its ships.
2390-2600	Subroutine to set up computer's ships, at random.
2610-2860	Set up variables for messages; subroutines for displaying those messages.
2870-2910	Keep track of which turn it is. Branch to either user's shot, or computer's shot.
2920-3170	Computer takes random shot at your grid if no ships are hit.
3180-3340	Read keyboard; INPUT user's shot at computer's grid.
3350-3570	Check for valid INPUT, hit, or miss.
3580-3710	Check for direction of hits on your ships.
3720-4150	Take random shot around last hit if only one hit on the ship.
4160-4450	If more than one hit on a ship takes another hit in proper direction.
4460-4620	Adjust variables when computer gets a hit.
4630-4770	Find out how many hits on each ship; used for both computer, and user.
4780-4980	Calculate score, and number of ships hit, but not sunk.
4990-5020	Display any ships that have been destroyed after every hit.
5030-5090	Display scores.
5100-5190	End of game message.
5200-5320	Re-initialize variables for next game.
5330-5340	END of game.
5350-5460	Subroutine to make sure ships are in line.

```
100 REM  *****************
110 REM  * BATTLE AT SEA *
120 REM  *****************
130 REM  99'ER VERSION 7.81.1
140 REM  BY W.K. BALTHROP
150 REM
160 REM
170 REM
180 RANDOMIZE
190 CALL SCREEN(12)
200 CALL CLEAR
210 PRINT TAB(6);"BATTLE AT SEA"
220 PRINT TAB(12);"BY"
230 PRINT TAB(7);"W.K. BALTHROP"
240 PRINT :::::::::::
250 OPTION BASE 1
260 DIM P(10,10),O(10,10),SH(5,5,2)
270 CALL COLOR(14,7,1)
280 CALL COLOR(15,11,1)
290 CALL CHAR(96,"000000FF7F3F1F")
300 CALL CHAR(97,"000000FFFFFFFF")
310 CALL CHAR(98,"3C7EFFFFFFFFFF")
320 CALL CHAR(99,"000000FFFEFCF8")
330 CALL CHAR(100,"1030707070707070")
340 CALL CHAR(101,"7070707070707070")
350 CALL CHAR(102,"787C7E7E7F7F7C78")
360 CALL CHAR(103,"7070707070701010")
370 CALL CHAR(104,"00080403FF7F3F")
380 CALL CHAR(105,"8C4C3CFEFFFFFF")
390 CALL CHAR(106,"01023C3FFFFFFF")
400 CALL CHAR(107,"000204F8FFFEFE")
410 CALL CHAR(108,"1030727478787878")
420 CALL CHAR(109,"7C7C70717A7C7C7C")
430 CALL CHAR(110,"7F7F787C7C7C7A79")
440 CALL CHAR(111,"7078787C7C727110")
450 CALL CHAR(112,"00108867FF7F3F")
460 CALL CHAR(113,"09C5C3F3FFFFFF")
470 CALL CHAR(114,"000204F8FFFEFE")
480 CALL CHAR(115,"103072747879787A7C")
490 CALL CHAR(116,"797A7C7C7F7F787C")
500 CALL CHAR(117,"7C7C7A79787C7C1A")
510 CALL CHAR(118,"0000003FFF7F3F")
520 CALL CHAR(119,"067EFEFFFFFFFF")
530 CALL CHAR(120,"000000E0FCFFFE")
540 CALL CHAR(121,"1030787878787878")
550 CALL CHAR(122,"787C7C7E7E7E7F7F")
560 CALL CHAR(123,"7C7C787878703020")
570 CALL CHAR(124,"03030F1FFF7F3F")
580 CALL CHAR(125,"006060F0FFFEFE")
590 CALL CHAR(126,"10307070787B7E7F")
600 CALL CHAR(127,"7C78787070707010")
610 CALL CHAR(128,"FF8181818181 81FF")
620 CALL CHAR(136,"815E2C366A3C2442")
630 CALL CHAR(144,"81667E3C3C7E6681")
640 CALL SOUND(-3000,220,30,554,20,1047,20,-8,30)
650 PRINT "         BATTLE AT SEA"
660 PRINT :"YOU MUST DESTROY THE ENEMY"
670 PRINT "SHIPS BEFORE THE COMPUTER"
680 PRINT "DESTROYS YOUR SHIPS."
690 PRINT :"TO SET UP YOUR SHIPS YOU"
700 PRINT "MUST ENTER COORDINATES ON"
710 PRINT "THE 10 X 10 GRID ON THE      RIGHT."
720 PRINT "ENTER THE ROW, THEN THE      COLUMN."
730 PRINT "EXAMPLE: A5"
740 PRINT "AFTER YOUR SHIPS ARE SET UP"
750 PRINT "YOU WILL TAKE A SHOT AT THE"
760 PRINT "ENEMY SHIPS BY ENTERING ONE"
770 PRINT "PAIR OF COORDINATES ON THE"
780 PRINT "ENEMY GRID."
790 PRINT "THE COMPUTER WILL THEN"
800 PRINT "TAKE A SHOT AT YOUR SHIPS."::
810 PRINT "THE COMPUTER CANNOT SEE"
820 PRINT "YOUR SHIPS. YOU CANNOT SEE"
830 PRINT "THE COMPUTER'S SHIPS."
840 PRINT "ENTER ANY KEY TO BEGIN."
850 CALL SOUND(1,-2,30)
860 CALL KEY(0,K,S)
870 IF S=0 THEN 860
880 CALL SCREEN(6)
890 CALL CLEAR
900 PRINT "  COMPUTER           YOU"
910 PRINT ::::::::::::::::::::
920 FOR X=5 TO 14
930 CALL VCHAR(X,5,X+60)
940 CALL HCHAR(X,6,128,10)
950 CALL HCHAR(X,18,128,10)
960 CALL VCHAR(X,17,X+60)
970 NEXT X
980 FOR X=6 TO 15
990 CALL VCHAR(15,X+12,X+42)
1000 CALL VCHAR(15,X,X+42)
1010 NEXT X
1020 S1$="CARRIER"
1030 S2$="BATTLESHIP"
1040 S3$="CRUISER"
1050 S4$="SUBMARINE"
1060 S5$="DESTROYER"
1070 FOR S=1 TO 5
1080 ON S GOSUB 1110,1160,1210,1260,1310
1090 GOSUB 1390
1100 GOTO 1360
1110 PR$=S1$
1120 LE=5
1130 S=1
1140 OS=0
1150 RETURN
1160 PR$=S2$
1170 LE=4
1180 S=2
1190 OS=8
1200 RETURN
1210 PR$=S3$
1220 LE=3
1230 S=3
1240 OS=16
1250 RETURN
1260 PR$=S4$
1270 LE=3
1280 S=4
```

Continued on p. 36

HARRIED HOUSEWIFE

By Regena



"HONEY — WOULD YOU BRING ME MY SLIPPERS?"

This matching game is dedicated to tired housewives everywhere who face the daily battle of keeping their houses clean amidst the unrelenting attacks from their kids, husbands, dogs, cats, visiting relatives, unexpected friends, and even home computers—those new family additions that seem to be forever spawning dust, out-of-place furniture, and loose papers.

Harried Housewife uses the color graphics of TI BASIC to depict eight household chores: dusting, sewing, washing clothes, doing dishes, cooking, vacuuming, shopping, and ironing. It is a matching game that even your young children will enjoy playing. The rules are simple: An array of 16 squares is displayed on the screen. Each square represents one of the eight chores, and there are two of each chore somewhere in the array. The object of the game is to find each pair. You do this by choosing two squares at a time and entering the corresponding two letters. As a letter is entered, the chore for that square is shown. If a match is made, the chore is considered finished and is listed on the right side of the screen. If a match is not made, the two selections are covered, and two more letters may be chosen.

When all eight pairs are matched, the housework is complete; you have a clean house and the game is over. But you mustn't take too long, because when the kids come home (determined by the counter in line 1420), everything gets scrambled and the harried housewife must start over . . . And as all harried housewives undoubtedly know: It's not easy to get a completely clean house. Often the goal has to realistically become somewhat more attainable—just seeing *how much* can be accomplished before the kids come home.

If you get too harried and want to quit, press "S" for stop. The arrangement of the current array will be displayed. After you have examined it, Shift C (BREAK) to end the program. If you really feel you must win more often — that is, winding up with everything matched to signify that elusive "clean house" — you can keep the kids out of the house longer by increasing the number in line 1420. Then enjoy the fantasy of a completely clean house all the time. What? Why can't your home computer make this fantasy actually come true? Be patient. It's just a matter of time . . . Anyway, in the words of a once-popular song: "Such are the dreams of the everyday housewife. . ."

Programming Techniques

This program illustrates the capabilities of TI-99/4 color graphics. Characters are defined in each of the eight user-defined character sets, and each set has a different color scheme. These eight sets are used for the eight chores; and for ease in programming, they are numbered 1 through 8.

Two characters in Set 2 are also redefined with a blue foreground and a red background ("FFFFFFFFFFFFFFFF" and "0") to draw the 16-square checkerboard array. It is drawn with a triple-nested FOR-NEXT loop (Statements 2040-2150).

The eight chores to be drawn are called in subroutines (Statements 2290 to 3060). The subroutines use x- and y-coordinates to define the placement of the special characters. The coordinates are specified before the subroutine is called. The coordinates for the chore for each of the sixteen squares where the chore may be drawn are listed in subroutines also (Statements 5350-5980).

To set up the array of 16 squares, two arrays are actually used: WORK(16) and HH(16). The WORK array is given the numbers of the eight chores: WORK(1)=1; WORK(2)=2; . . . WORK(9)=1; WORK(10)=2; etc. (statements 3370-3400). For the HH array, a subscript RR is chosen as a random number from 1 to 16. HH(RR) is then set equal to WORK(RR), and then WORK(RR)=0 so it won't be chosen again. This process continues until all 16 numbers of the HH array have been filled randomly with the numbers from the WORK array (statements 3410-3470). These numbers are the chore numbers for the squares. For example, HH(4)=7 means behind the 4th square(D) would be chore number 7(shopping).

The WORK array is then reset equal to the HH array so the chores can be printed in order on the squares for a "clean house" or for "stop".

As the game is being played, the HH elements are set equal to zero if a match is made, so the match can only be scored once. If a player chooses a square which has previously been part of a matched pair, the word "DONE" appears across the square.

EXPLANATION OF THE PROGRAM

Harried Housewife

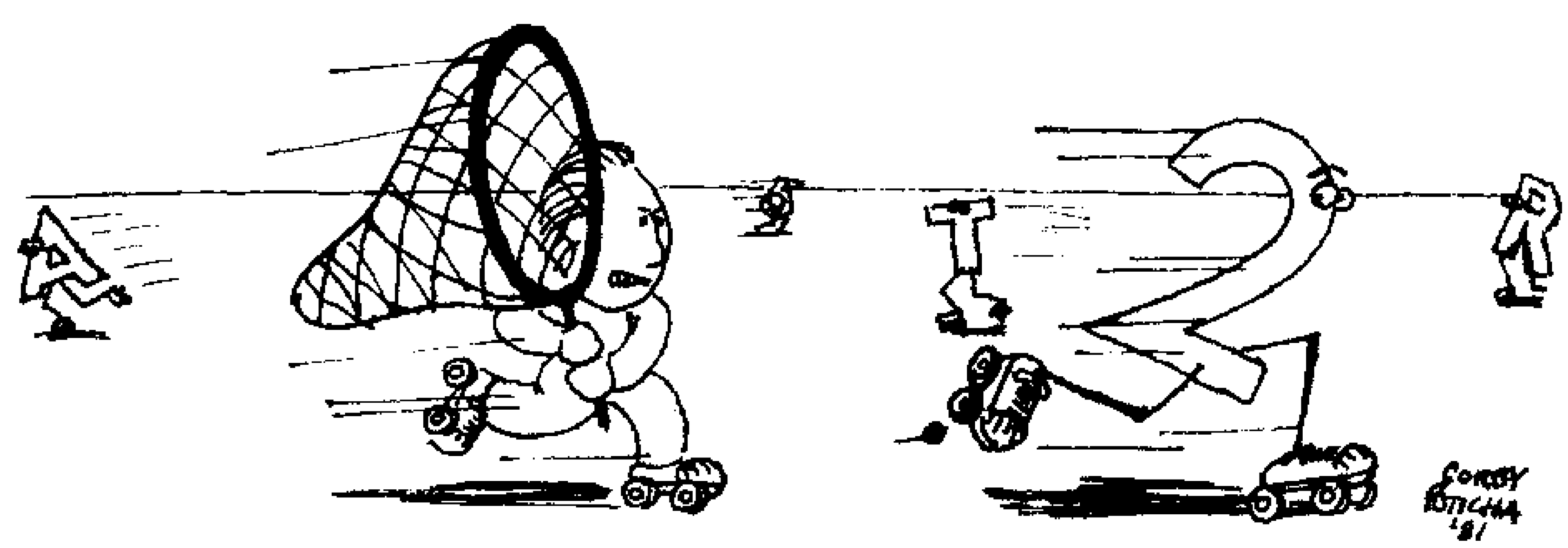
Line Nos.	
130-160	Prints title screen.
170-240	Defines colors for eight household chores.
250-800	Defines special characters for drawing the chores.
810	Displays the eight chores on title screen.
820-830	Sets counters for the number of trial guesses and the number of successful matches.
840	Dimensions arrays to handle 16 elements.
850-860	Redefines characters for checkerboard.
870-880	Delays for title screen.
890-900	Clears screen and makes it yellow.
910	Defines colors for checkerboard.
920	Draws checkerboard and labels it.
930	Assigns the chores for each square in array.
940-990	Prints "HOUSEWORK".
1000-1110	Prints "MATCH 2 LETTERS".
1120-1140	Prints two red lines for the letters chosen.
1150-1180	Waits for letter A-P to be pressed.
1190	Prints the chosen letter.
1200-1210	Finds chore number and coordinates for square chosen.
1220-1240	If the square has been previously matched, prints "DONE".
1250	Draws first chore on square.
1260-1310	Waits for second letter to be pressed and prints it.
1320-1330	Finds the chore number and coordinates for that square.
1340-1370	Prints "DONE".
1380	Draws second chore on square.
1390	Checks for a match.
1400	Increments the number of trials.
1410	If TIME=10 prints message to hurry.
1420	If TIME=12, kids come home.
1430	Branches if TIME is less than 10.
1440	Clears previous message.
1450-1500	Prints "OH NO! KIDS ARE HOME!"
1510-1530	Reprints checkerboard and scrambles chores for a new game.
1540	Prints "PRESS ENTER TO CONTINUE" and waits for response, covers squares for next choice.
1550-1600	Prints "SPEED-KIDS WILL BE HOME SOON!"
1610	Same as 1540.
1620-1730	Correct match is made, sounds tone of A, prints finished chore.
1740-1750	Sets elements matched to zero so they can't be scored again.
1760	Returns for next choice.
1770-1820	If all eight matches have been made, prints "CLEAN HOUSE!!"
1830	Prints S if player wants to stop.
1840	Resets HH array to current arrangement.
1850-1890	Shows all chores in array.
1900-1960	Clears all other printing.
1970-2020	Prints "HOUSEWORK NEVER ENDS".
2030	Holds screen until Shift C (BREAK) is pressed.
Subroutines	
2040-2150	Prints checkerboard.
2160-2210	Prints letters A to P on squares.
2220-2280	Prints "S=STOP" and returns.
2290-2390	Draws feather duster.
2400-2470	Draws sewing machine.
2480-2580	Draws T-shirt for washing.
2590-2670	Draws cup and saucer for dishes.
2680-2760	Draws pan for cooking.
2770-2860	Draws vacuum cleaner.
2870-2960	Draws shopping basket.
2970-3060	Draws ironing board.
3070-3280	Places symbols on title screen.
3290-3360	Plays music for title screen.
3370-3400	Puts two sets of chore numbers in WORK array.
3410-3470	Randomly arranges chores in HH array, two of each chore.
3480-3500	Resets WORK array equal to HH array.
3510	Restarts number of matches.
3520-3560	Clears printed list of matches made.
3570-3600	Resets HH array to original WORK array for printing.
3610-4470	When a match is made, blinks the picture and prints the chore in the "Finished" list; prints labels under pictures in the squares.
4480-4550	Prints "PRESS ENTER TO CONTINUE" and waits for response.
4560-4570	Clears messages.
4580-4610	Covers squares again and relabels them.
4620	Return for next choice.
4630-5260	Subroutines for covering particular square.
5270-5300	Colors blue square.
5310-5340	Colors red square.
5350-5980	Designates the chore number and coordinates for the square chosen.

```
100 REM  HARRIED HOUSEWIFE
110 REM  99'ER VERSION 7.81.1
120 REM  BY REGENA
130 CALL CLEAR
140 PRINT TAB(10);"HARRIED"
150 PRINT ::TAB(9);"HOUSEWIFE"
160 PRINT :::::::::TAB(9);"BY REGENA"
170 CALL COLOR(9,7,15)
180 CALL COLOR(10,13,12)
190 CALL COLOR(11,14,11)
200 CALL COLOR(12,16,3)
210 CALL COLOR(13,7,12)
220 CALL COLOR(14,5,8)
230 CALL COLOR(15,15,16)
240 CALL COLOR(16,3,16)
250 CALL CHAR(96,"0000040EBEBEFFFF")
260 CALL CHAR(97,"0000000020E0C0C")
270 CALL CHAR(98,"0201010703010101")
280 CALL CHAR(99,"FFFFFFFFFFFFFFFF")
290 CALL CHAR(100,"F0E0C0F0F1E080C")
300 CALL CHAR(101,"03070E1C3870E0C")
310 CALL CHAR(102,"FF0E03")
320 CALL CHAR(103,"0")
330 CALL CHAR(104,"FFFFFFFFFFFFFFFF")
340 CALL CHAR(105,"FCFCFCFFFFFFFFFF")
350 CALL CHAR(106,"FCFCF8F8F0E0C")
360 CALL CHAR(107,"FFFFFFFFFCFCFCFC")
370 CALL CHAR(108,"FFFFFFFFFF")
380 CALL CHAR(109,"0")
390 CALL CHAR(112,"000000000F0F0F0F")
400 CALL CHAR(113,"0000000081C3FFFF")
410 CALL CHAR(114,"00000000F0F0F0F0")
420 CALL CHAR(115,"0F0F")
430 CALL CHAR(116,"FFFFFFFFFFFFFFFF")
440 CALL CHAR(117,"F0F")
450 CALL CHAR(118,"FFFFFFFF")
460 CALL CHAR(119,"0")
470 CALL CHAR(120,"00000000000C0F0F")
480 CALL CHAR(121,"000000000000FFFF")
490 CALL CHAR(122,"0000000000C0C0FC")
500 CALL CHAR(123,"0F0F0F0F0F0F0F0F07")
510 CALL CHAR(124,"FFFFFFFFFFFFFFFF")
520 CALL CHAR(125,"FEC6C6C6DCF8E08")
530 CALL CHAR(126,"FFFF")
540 CALL CHAR(128,"1F1F1F1F1F1F1F1F")
550 CALL CHAR(129,"FFFFFCFCFCFCFCFC")
560 CALL CHAR(130,"FFFF")
570 CALL CHAR(131,"1F1F0F")
580 CALL CHAR(132,"FCFCF8")
590 CALL CHAR(133,"0")
600 CALL CHAR(136,"1F0F010000000303")
610 CALL CHAR(137,"80C0C0C0C0F0F8F8")
620 CALL CHAR(138,"0303030303030101")
630 CALL CHAR(139,"F8F0F0F0F0F0E0E")
640 CALL CHAR(140,"E0C7CF7FFFFFFFFF")
650 CALL CHAR(141,"0080C0C0F0F0E")
660 CALL CHAR(142,"0")
670 CALL CHAR(144,"000000000040C1A19")
680 CALL CHAR(145,"090F09090F09090F")
690 CALL CHAR(146,"FE252424FF2424FF")
700 CALL CHAR(147,"00E09E92FE9292FE")
710 CALL CHAR(148,"06090906")
720 CALL CHAR(149,"0")
730 CALL CHAR(152,"000000001F1F1F1F")
740 CALL CHAR(153,"00000000F0FFFFFF")
750 CALL CHAR(154,"000000000F0FEFE")
760 CALL CHAR(155,"1F1F1F040201")
770 CALL CHAR(156,"FFFE2040810A04")
```

Continued on p. 38

Sprite Chase

By Ron Binkowski



Wait . . . Wait . . . Wait . . .
When will they get here?
Wait . . . Wait . . . Wait . . .

"Hi dear, anything in the mail today? Did you look between the doors? Oh. Shucks."

"Hello Ginny. What? You accepted a package from UPS for me? Great! Could you get it for me? Thanks."

"See you later dear, I'll be downstairs."

They're here . . .
The SPRITES are here . . .
NOW, WHAT CAN I DO WITH THEM?

Skim through the manual, page 25. Uh huh. OK. Yea. This looks great! Let's get a little deeper. Page 64. Oh oh. COINC looks like the ALL option doesn't tell you which SPRITES "coincidenced." I hope someone can find out where to PEEK for this.

Now, what shall I do with them? Something simple. Design some cute characters? No, let's just get those SPRITES moving. Since COINC ALL might take some time to figure out which SPRITES are coincidental, I'll stick to one SPRITE versus another. How about a series to chase? Numbers . . . Letters . . . ROTATION . . . That's it . . .

A short game chasing the 10 numbers or a longer game chasing the 26 letters. I'll try the MAGNIFY too. I'll need a numeric variable for the COINC tolerance for that. I guess 8 for normal size and 16 for double size. I'll generate the number or letter SPRITES to go any which way at some speed between –25 and 25. I'll stick to the 8 directions around the arrows for the chaser or else I'll get so tangled up in the math that I'll never move a SPRITE. Wish I had Joysticks. I guess some kind of clock would be good for scoring.

Well, here we go:

EXPLANATION OF THE PROGRAM
Sprite Chase

Line Nos.	
170-200	Instructions.
210-280	Set up variations for play.
290-300	Reset for start of game.
310	Make clock numbers reverse image.
320-330	Put the Chaser somewhere in middle of the screen.
340-360	Create the Chasees.
370-390	The chase has begun.
400-450	While waiting for a direction key to be pressed, keep the clock going and check for a coincidence when the Chaser is stationary.
460-530	Start the Chaser in the direction of key pressed.
540-590	While awaiting release of direction key, check for a coincidence when the Chaser is moving; keep clock going.
600-610	Stop the Chaser; wait for another key to be pressed.
620-650	Caught one; go for the next one.
660-710	End of game.
720	That's it.

A FEW POST SCRIPT NOTES:

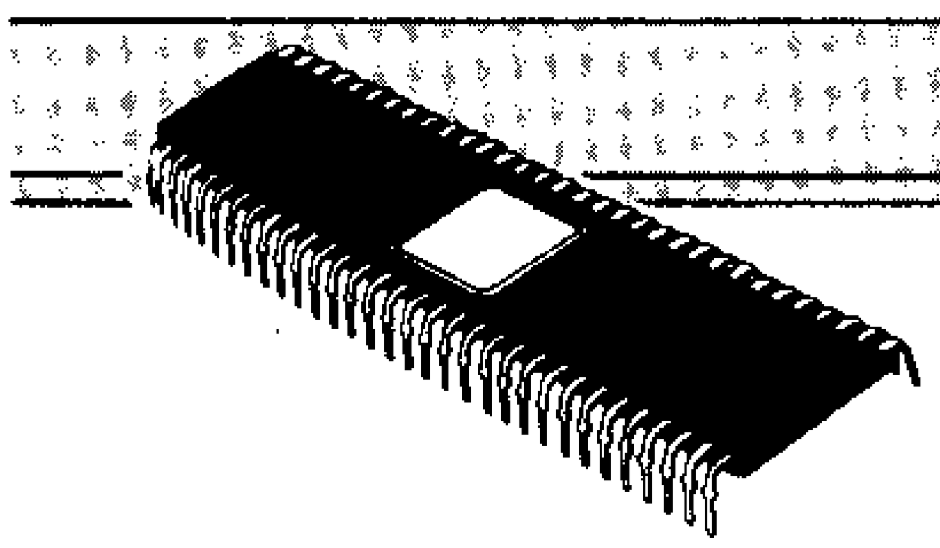
If a SPRITE is moving slowly in a vertical direction, it might go off the top or bottom of the screen for a while, but can be caught there.

If you insert COINC statements between a lot of the instructions and check the HIT field, you probably would reduce the times a coincidence is missed.

If you leave the Chaser in its original position, all targets will eventually pass through it. I wonder how long this would take?

If it sounded like I was talking to myself, *I was!* Doesn't everyone???

```
100 REM ****************
110 REM * SPRITE CHASE *
120 REM ****************
130 REM 99'ER VERSION 7.81.1XB
140 REM BY RON BINKOWSKI
150 REM
160 REM
170 CALL CLEAR
180 PRINT "USE THE FOUR ARROW KEYS AND W,R,Z,C KEYS
    TO CHASE THE   LETTERS OR NUMBERS." :: PRINT
190 PRINT "YOU MUST CATCH THEM IN ALPHAOR NUMERIC
    SEQUENCE." :: PRINT
200 PRINT "PRESS 'L' FOR LARGE TARGET. 'S' FOR
    SMALL TARGET." :: PRINT
210 CALL KEY(0,GOT,STATUS)
220 IF STATUS=0 THEN 210
230 IF GOT=76 THEN T=16 :: CALL MAGNIFY(2)ELSE
    IF GOT=83 THEN T=8 ELSE 210
240 PRINT "FOR NUMBERS PRESS 'N',":"FOR LETTERS
    PRESS 'L'." :: PRINT
250 CALL KEY(0,GOT,STATUS)
260 IF STATUS=0 THEN 250
270 IF GOT=78 THEN TARGS=10 :: CH=47 ELSE IF GOT=76
    THEN TARGS=26 :: CH=64 ELSE 250
280 CALL CLEAR
290 RANDOMIZE
300 COUNT=0
310 CALL COLOR(3,2,9):: CALL COLOR(4,2,9)
320 CALL CHAR(96,"FFFFFFFFFFFFFFFF")
330 CALL SPRITE(#28,96,2,90,120,0,0)
340 FOR A=1 TO TARGS
350 CALL SPRITE(#A,A+CH,2,90,120,INT(RND*50-25),
    INT(RND*50-25))
360 NEXT A
370 CALL SOUND(100,555,0)
380 FOR A=1 TO TARGS
390 CALL COLOR(#A,16)
400 CALL KEY(0,GOT,STATUS)
410 COUNT=COUNT+1
420 DISPLAY AT(24,1)SIZE(6):COUNT
430 CALL COINC(#28,#A,T,HIT)
440 IF HIT=-1 THEN 620
450 IF STATUS=0 THEN 400
460 IF GOT=69 THEN CALL MOTION(#28,-30,0):: GOTO 540
470 IF GOT=88 THEN CALL MOTION(#28,30,0):: GOTO 540
480 IF GOT=68 THEN CALL MOTION(#28,0,30):: GOTO 540
490 IF GOT=83 THEN CALL MOTION(#28,0,-30):: GOTO 540
500 IF GOT=87 THEN CALL MOTION(#28,-30,-30):
    : GOTO 540
510 IF GOT=82 THEN CALL MOTION(#28,-30,30):: GOTO 540
520 IF GOT=90 THEN CALL MOTION(#28,30,-30):: GOTO 540
530 IF GOT=67 THEN CALL MOTION(#28,30,30):: GOTO 540
540 CALL KEY(0,GOT,STATUS)
550 CALL COINC(#28,#A,9,HIT)
560 IF HIT=-1 THEN 620
570 COUNT=COUNT+1
580 DISPLAY AT(24,1)SIZE(6):COUNT
590 IF STATUS=-1 THEN 540
600 CALL MOTION(#28,0,0)
610 GOTO 400
620 CALL DELSPRITE(#A)
630 CALL SOUND(100,-7,0)
640 CALL MOTION(#28,0,0)
650 NEXT A
660 CALL CHARSET
670 PRINT "YOUR SCORE IS ";COUNT
680 PRINT "ENTER 'Y' TO PLAY AGAIN: "
690 CALL KEY(0,GOT,STATUS)
700 IF STATUS=0 THEN 690
710 IF GOT=89 THEN 280
720 PRINT "      BYE"
```



TMS9900 Machine & Assembly Language

Part 2 : Registers, Programming, & The Need For Assemblers

By Dennis Thurlow

We left off last time explaining registers—devices which act like Random Access Memory (RAM) in that they can be used to store and later recall values, but are different from RAM in that they are *not* addressed by the bus. Registers are on the CPU chip itself, and only the CPU can address them.

We discussed the Program Counter (PC) as being 16 bits long, allowing it to address 65536 bytes of memory, and noticed that the *ones* bit is not represented on the address bus. This was so that the PC could increment two bytes and hook them together making a single 16–bit word, or "gobbyl," while the address bus would appear to increment only once. Other than this, the TMS9900 PC doesn't differ from those of other CPU's; it still tells the computer where to get its next byte or word from.

Status Register

Almost every CPU has some kind of flag(s). These are set (high) and reset (low) by actions performed in the manipulations of data. Different instructions affect different flags. Modern CPUs combine several flags into a single Status Register. The TMS9900 is no exception. Its Status Register (ST) is 16 bits long. Bits 7-11 are not used at present. The others are shown in the drawing below, and are explained in the text.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
L >	A >	EQ =	C	OV	P	X XOP									
			CARRY	OVERFLOW	PARITY		UNUSED					INTERRUPT MASK			

Each of these conditions will be discussed in more detail as examples are shown. Until then, these simple descriptions will help.

The four bits labeled 12 - 15 can select up to 16 interupt levels. All levels equal to or above the level indicated are enabled.

Bit 0 is set after any operation where the destination value (answer) is greater than the source value (the first operand used and which remains unchanged). All 16 bit are used for the comparison.

Bit 1 is similar to bit 0 except that the values are compared as signed integers. The MSB (most significant bit designates the sign of the integer, with a 1 meaning negative and a 0 meaning positive. The range is +32,767 to –32,768 Negative numbers are represented in a twos complement fashion.

Computer math is cyclic. This means that if you add 1 to the highest possible 16-bit number (FFFF hex), you g back to 0000 hex with a *carry* bit that is set. If yo subtract 1 from 0000 hex without the carry, you get a *overflow*; but if the carry is set, you get FFFF hex. Therefore, –1 is FFFF hex in twos complement. To see its usefulness, let's add –1 and 1: FFFF hex plus 0001 hex equa 0000, the carry is set, and the answer is zero. In a nutshell this whole business of twos complements and carry bits i simply a way to subtract by adding.

Bit 2 is set if the two operators are equal.

Bit 3 is set if a 1 is shifted out of an operator, or if carry occurs in a math operation.

Bit 4 is set if the math requested cannot be done.

Bit 5 is set if the parity is odd, and reset if it is even Odd parity means that there is an odd number of 1s in th binary representation of an operator.

Bit 6 is set after an extended operation has been completed. This is done because an interrupt is not checked fo after completion of an extended operation. (You therefor may wish to have the software check for one if this flag i set.)

The ALU

Most CPUs have an Arithmetic/Logic Unit (ALU) where the simple math is performed. An accumulator, a special register used by the ALU, usually contains the answers to the math. In the TMS9900 there is no accumulator because the destination address is almost always the equivalent. This means, in effect, that any memory location *can* be the accumulator. There *is* an ALU on the TMS9900 chip, but its operation is intrinsic to the instructions.

Other Registers

Most CPUs have a few extra registers where quickly needed values can be stored, as well as a register called **Stack Pointer** which points to a section of memory wher more data can be "piled" and then quickly accessed. These two concepts have been combined on the TMS9900 into a single **Workspace Pointer Register** (WP). The WP points to a block of 32 bytes of memory arranged as 16 **workspaces** (WS), each 16 bits long. The workspaces are synonomous with registers, and are used the same way. We can change the WP in several ways, and can save the old WP when a new one is used. This allows us to return to the old one if we need to. This set-up, in effect, acts like an elaborate stack.

There are five different ways to use these WS registers to indicate an operator for an instruction. These **addressing modes** are as follows.

1. Workspace Register Mode code 00	—the data in the indicated register is the data used.
2. Workspace Register Indirect code 01	—the data in the register is treated as the address of the real data.
3. WS Register Indirect w/Auto-Increment code 11	—same as above, but the register is incremented upon completion.
4. Symbolic or Direct code 10 destination 0	—the address of the data follows the instruction in memory.
5. Indexed code 10 destination 1-15	—address is found by adding the symbol amount to an indirect register address.

There are three other addressing modes not dealing with registers, per se: (1) The **immediate mode** has the data immediately following the instruction code. In other words, the address of the data is the address immediatly following the PC. (2) The **CRU mode** has the address of an external input/output (I/O) device determined by bytes 3-12 of register 12. (3) The **JMP instruction** (and all variations thereof) use the last 8 bits of the instruction to determine where on a 256 byte page to jump. The PC indicates the center of the page, so the jump can be from PC-128 to PC+127. One byte is taken up by the jump instruction itself. The 8 bits store the relative jump in twos complement form.

Programming and the Need for Assemblers

If your CPU is the TMS9900, the simplest computer you could construct would be composed of a clock, a CPU, some memory, a few control switches, 16 data switches, 16 lights for read out, and 15 address switches. It would be crude and slow to program, but once programmed, it would operate as well as any other computer. But how could we program it?

Suppose we wanted to load register 1 with zero, and then increment it until it was equal to either 1024 (decimal) or register 2. The first step can be done several ways. Immediately loading register 1 with 0 comes to mind first. A little investigation of the instructions for the chip show that we could save a word of memory by using the *Clear* command. Figure 1 shows the register format for the various commands, and Figure 2 shows the **op codes** for the instructions.

FORMAT	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	OP CODE			B	T_d		D				T_s		S			
2	OP CODE								RELATIVE JUMP							
3	OP CODE						D				T_s		S			
4	OP CODE						C				T_s		S			
5	OP CODE								C				W			
6	OP CODE										T_s		S			
7	OP CODE											N				
8	OP CODE											N	W			
	IMMEDIATE VALUE															
9	OP CODE						D				T_s		S			

KEY

- B 1=byte 0=word
- T_d destination address mode
- D destination address
- T_s source address mode
- S source address
- C counter
- W register number
- N unused
- RELATIVE JUMP from +127 to −128

T_d/T_s FIELD CODES

- 00 Register
- 01 Indirect
- 10 with R0, symbolic
- 10 with R1—R15, indexed
- 11 Indirect with increment

Figure 1.

Using this information, we can now determine the binary values of each word. **Load immediate** uses the first 10 bits as the op code; the 11th bit is not used, and bits 12-15 select the register. This means the first byte is

00000010000X0001 , where X can be 1 or 0.

The second byte is the value to load, and in this case would be all zeros.

Now, just flip each switch on if there is a 1 at the corresponding bit, off if there is a zero. Press the *Input* control switch (it also might be called *Dump*, or *Load*, or . . .) and the instruction is stored in whatever address the address switches are set to. Then add 1 to the address switches (which adds 2 to the PC) and set all the data switches to zero. Press *Input* again and our complete instruction is ready.

If instead, we use the *Clear* command, we would use the single-operand general format with the first 10 bits being the op code. The next two bits indicate address mode, and the last 4 bits select the register. Since we want to clear the register itself (not the word it points to), the code is 00, and the whole instruction is 0000010011000001.

Even with a hex keypad and a small monitor program, it would be a very time-consuming process to piece together the binary words, and then convert to hex and type them in. Typing in 04C1 is easier than setting swiches to

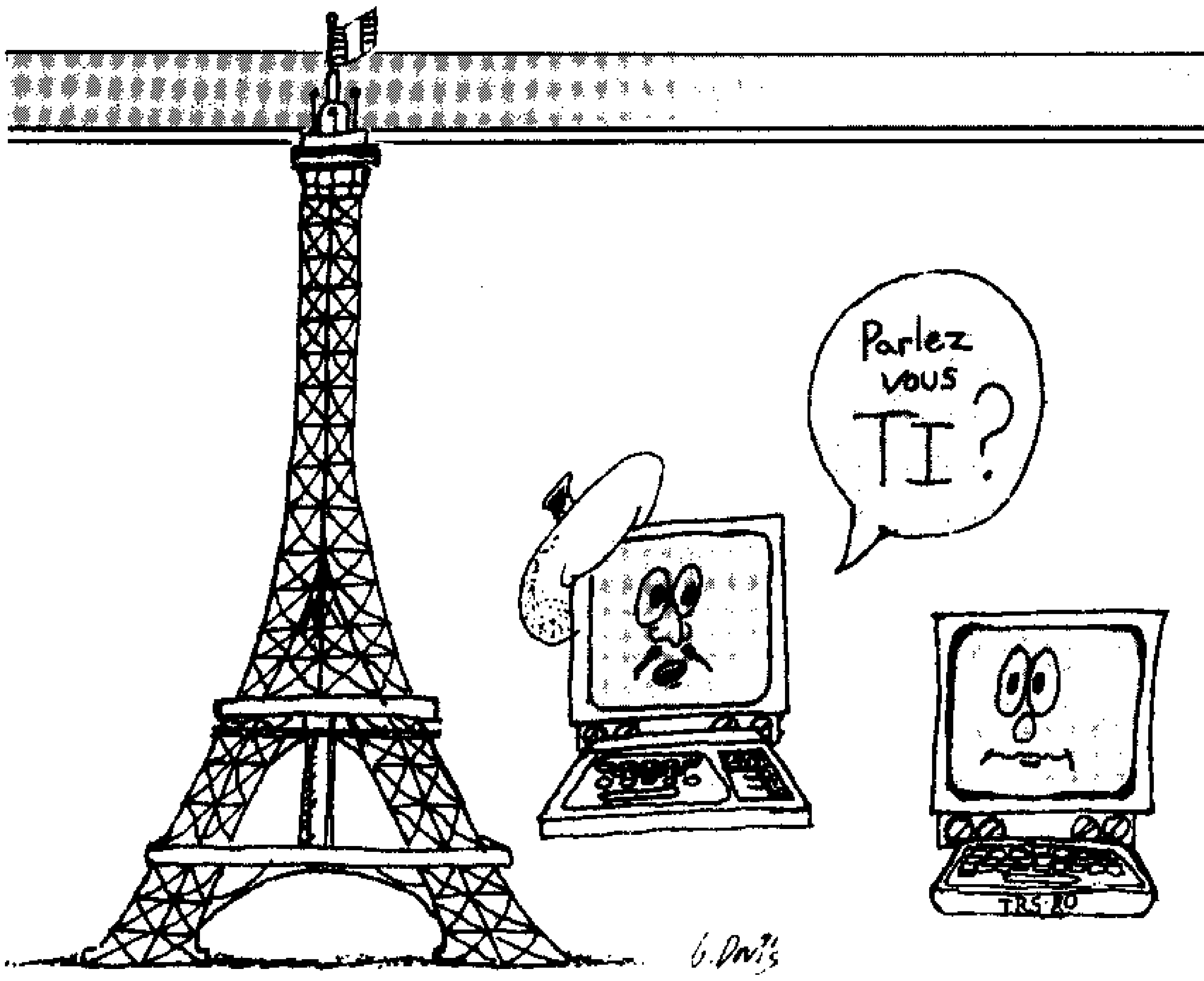
0000010011000001 ,

but putting together those op codes is just the tedious, boring kind of work that computers are supposed to free us of. So why not use them for that?

Why not, indeed . . . That's exactly what we'll do next issue when we look at a TMS9900 assembler.

Figure 2.

Mnemonic	Op Code	Format	Status	Bits Affected	Meaning
A	1010	1	0–4		Add words
AB	1011	1	0–5		Add bytes
ABS	0000011101	6	0–4		Absolute Value
AI	00000010001	8	0–4		Add immediate
ANDI	00000010010	8	0–2		And immediate
B	0000010001	6	– – –		Branch
BL	0000011010	6	– – –		Branch and Link (R11)
BLWP	0000010000	6	– – –		Branch, load WP
C	1000	1	0–2		Compare words
CB	1001	1	0–2, 5		Compare byte
CI	00000010100	8	0–2		Compare immediate
CKOF	0000001111000000	7	– – –		External Control
CKON	0000001110100000	7	– – –		External Control
CLR	0000010011	6	– – –		Clear
COC	001000	3	2		Compare Ones Corresp. (OR)
CZC	001001	3	2		Compare Zero Corresp. (AND)
DEC	0000011000	6	0–4		Decrement by one
DECT	0000011001	6	0–4		Decrement by two
DIV	001111	9	4		Divide
IDLE	0000001101000000	7	– – –		Computer idles
INC	0000010110	6	0–4		Increment by one
INCT	0000010111	6	0–4		Increment by two
INV	0000010101	6	0–2		Invert (complement)
JEQ	00010011	2	– – –	(ST2=1)	Jump if equal
JGT	00010101	2	– – –	(ST1=1)	Jump greater than
JH	00011011	2	– – –	(ST0 and ST2=1)	Jump high
JHE	00010100	2	– – –	(ST0 or ST2=1)	Jump high or equal
JL	00011010	2	– – –	(ST0 and ST2=0)	Jump low
JLE	00010010	2	– – –	(ST0=0 or ST2=1	Jump low or equal
JLT	00010001	2	– – –	(ST1 and ST2=0)	Jump less then
JMP	00010000	2	– – –	(none checked)	Jump unconditionally
JNC	00010111	2	– – –	(ST3=0)	Jump no carry
JNE	00010110	2	– – –	(ST2=0)	Jump not equal
JNO	00011001	2	– – –	(ST4=0)	Jump no overflow
JOC	00011000	2	– – –	(ST3=1)	Jump on carry
JOP	00011100	2	– – –	(ST5=1)	Jump odd parity
LDCR	001100	4	0–2, 5		Load CRU
LI	00000010000	8	0–2		Load immediate
LIMI	00000011000	8	12–15		Load immed. INT mask
LREX	0000001111100000	7	12–15		External control
LWPI	00000010111	8	– – –		Load immed. WP
MOV	1100	1	0–2		Move word
MOVB	1101	1	0–2, 5		Move byte
MPY	001110	9	– – –		Multiply
NEG	0000010100	6	0–4		Negate (2's comp.)
ORI	00000010011	8	0–2		OR immediate
RSET	0000001101100000	7	12–15		External control
RTWP	0000001110000000	7	0–6, 12–15		Return with WP
S	0110	1	0–4		Subtract word
SB	0111	1	0–5		Subtract byte
SBO	00011101	2	– – –		Set CRU bit to one
SBZ	00011110	2	– – –		Set CRU bit to zero
SETO	0000011100	6	– – –		Set ones
SLA	00001010	5	0–4		Shift left (0 fill)
SOC	1110	1	0–2	Words (OR)	Set ones corresp.
SOCB	1111	1	0–2, 5	Bytes (OR)	Set ones corresp.
SRA	00001000	5	0–3		Shift right (MSB fill)
SRC	00001011	5	0–3		Shift right circular
SRL	00001001	5	0–3		Shift right zero fill
STCR	001101	4	0–2, 5		Store from CRU
STST	00000010110	8	– – –		Store ST
STWP	00000010101	8	– – –		Store WP
SWPB	0000011011	6	– – –		Swap bytes
SZC	0100	1	0–2	Words (AND)	Set zero corresp.
SZCB	0101	1	0–2, 5	Byte (AND)	Set zero corresp.
TB	00011111	2	2		Test CRU bit
X	0000010010	6	– – –		Execute
XOP	001011	9	6		Extended operation
XOR	001010	3	0–2		Exclusive OR



TRS-80 BASIC to TI BASIC

By Fred Forster
395 S. W. 6th
Gresham, OR 97030

Tucked away in my basement, I have both a Radio Shack TRS-80 and a Texas Instruments TI-99/4. The half dozen personal computer magazines I read each month provide coding and ideas for many new programs for my TRS-80. I now have a large collection of these programs and have grown to really appreciate the help and enjoyment this software "library" provides. Unfortunately, it just hasn't been that easy to acquire software for the TI machine. [But now, with the birth of *99'er Magazine* this situation will be rapidly remedied—Ed.] The solution for me was obvious. I'd convert my TRS-80 programs to TI BASIC.

At the suggestion of *99'er Magazine*'s editor, I read an article by Harley M. Templeton appearing in the November 1980 issue of *Personal Computing* magazine. Although the article highlighted the major differences between the versions of BASIC used on the two systems, it didn't point out which differences *matter*, and which are merely interesting but of little practical importance. As you might expect, the only way to find out is to actually convert a program and learn from the problems that are encountered.

To set up a fair test, I selected TRS-80 programs from opposite ends of the spectrum: The first was a "number cruncher" which I had written to convert the number correct on a test to a scaled value on a continuum of learning. (My nine-to-five job involves the management of the standardized testing programs for the Portland School District.) The other program was an adaptation of the ideas behind a slot machine game in David Ahl's *Basic Computer Games*—a program with extensive use of graphics.

The first trouble I encountered was in converting the PRINT AT command available on the TRS-80. The procedure suggested by Templeton was to set a loop as follows:

```
100  A$="SOMETHING TO BE
     PRINTED STARTING AT 10,12"
200  FOR I=1 TO LEN(A$)
300  N1=ASC(SEG$(A$,I,1))
400  CALL HCHAR(10,12,N1)
500  NEXT I
```

In theory this works fine, but it is slow if the string length is long; single characters don't walk across the screen—they *crawl!* Since the program requires a prompt printed in the middle of the screen to cue the operator to enter the next five values for the scaling procedure, my final solution was to use the following coding:

```
100  PRINT "MESSAGE AT THE
     MIDDLE OF THE SCREEN"
200  PRINT : : : : : : : : :
```

This procedure causes the text prompt to scroll up from the bottom to the middle of the screen. It is not especially speedy, but it is fast enough for data entry in cases where you don't need lines at the top of the screen that disappear as a result of this scrolling action.

The ease with which the "number crunching" code converted was a pleasant surprise. It was important to keep track of the differences in the line numbers for GOTO's and other branches, but that, in fact, presented little problem. What was more difficult was converting the logic of IF . . . THEN . . . ELSE clauses. TRS-80 (Microsoft) BASIC allows multiple statements per line—coding that is messy to convert, but not too difficult. It does, however, also allow multiple statements following the THEN and ELSE—coding that is difficult to keep straight and re-code. The multiple line conditionals can be converted, but the conversion requires a clear head and a basic understanding of how the program works.

Since I had written the TRS-80 program myself (it had more lines of documentation than coding) and naturally understood its operation, the conversion was fairly straight forward. After changing nearly all the PRINT and PRINT AT statements, the program worked the first time (surprise). To check it out, I made a comparison run on the TI-99/4 and the TRS-80. Surprisingly, they ran the same job in almost the same time (three minutes for a fourty item test). Finally, I spruced up the program a little with CLEAR and CALL SCREEN commands to take advantage of the color options available on the TI machine.

The second program was a challenge. It had essentially four main parts: (1) an introductory message, (2) the set-up graphics of the "slot machine," (3) the rotation of the wheels in the slot machine, and (4) the determination of the winnings and losses. The first and easiest part of the program to set up was the section which printed the introductory messages. I couldn't resist adding the CALL SCREEN command and sprucing up the comments to make it more attractive (at least to me). In this instance, the lack of speed for the HCHAR command was a benefit since it painted the screen at a leisurely-yet-pleasing pace. Before I was through, I had changed all the code in this section for aesthetic reasons.

My real conversion problems began in the second section. There, I came face-to-face with the significant differences in the way graphics are handled by the two systems. In moving from a screen of 16x64 to one of 24x28, I had to stop and develop a new outline shape for the slot machine—one that would fit the TI screen. Deciding the colors to be used in defining the outline of the machine and the shapes to be matched (cherry, bar, bell, orange, lemon) took extra time. After some experimentation, dark blue against a white background, the lemon became a lime (dark green). To develop a new set of four characters for the orange, I experimented with CALL CHAR until the figure finally looked like a circle instead of one of Dali's exploded watches. Since there isn't an orange color available, the orange became a plum (magenta). I was still a character short, so I used the heart from the back of the user's manual.

En route to coding this part of the program, I had to define the shapes—assigning them to one of the sixteen character sets. I, however, twice made the mistake of trying to conserve memory by using one of the character sets with pre-defined codes. This caused errors in the print statements using these codes. The moral of that experience: *Whenever possible, stay away from the first eight character sets when defining new characters.* It took a while to work the kinks out of this section, but the addition of color made a tremendous difference, and I became hooked on TI graphics. (I'll probably never run the TRS-80 version of this program again.)

At this point, I realized that virtually every line of the original program had been rewritten in the move to the TI machine. Since this was to be an article on program conversion, not programming, I called the editor at *99'er Magazine* to make sure I hadn't missed the point of the article. Gary, however, wasn't surprised at all, and encouraged me to include suggestions on rewriting as well as conversion.

The third section of this program was probably the toughest to convert. I have been responsible for programming and systems analysis for over ten years on a variety of large computer systems. This has required establishing "structured" programming standards for every program with which I work. Even though I had *personally* keyed in the slot machine program, I had forgotten how poorly it was documented. This is not a criticism of Ahl's book, but rather, a realistic comment on what you are likely to encounter when converting a program. After an hour of tracing through a maze of GOSUBS without the benefit of a single comment, I decided on a total rewrite.

The TRS-80 version had the program determine the coordinates of one of the nine open spots on the slot machine, and then perform a PRINT AT at that location. Using FOR . . . NEXT loops, it was possible to overprint the nine spots to give the illusion of a rotating machine wheel. By converting the PRINT AT commands to HCHAR calls and storing the four codes for each shape in an array, it was possible to simulate this action on the TI-99/4. The graphics were fantastic (an unbiased estimate),

Want to Get Published?

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

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

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

Send your manuscripts and correspondence to:

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

SUMMARY OF COMMANDS

TRS-80 Commands Not Requiring Conversion

ABS	GOSUB	REM
ASC	INPUT	RESTORE
ATN	INT	RETURN
CHR$	LEN	SGN
COS	LET	SIN
DATA	LOG	SQR
DIM	ON/GOSUB	STR$
END	ON/GOTO	TAN
EXP	PRINT	VAL
GOTO	READ	

TRS-80 Commands Easily Converted

TRS-80	TI BASIC
CLS	CALL CLEAR
FIX	INT
INKEY$	CALL KEY
INPUT#–1	INPUT#1
LEFT$(A$,N)	SEG$(A$,1,N)
MID$(A$,N1,N2)	SEG$(A$,N1,N2)
RANDOM	RANDOMIZE
RIGHT$(A$,N)	M=LEN(A$)–N+1 SEG$(A$,M,N)
RND(N)	INT(N*RND+1)
STOP	BREAK
TAB	TAB,(with comma)
?	PRINT
,	REM

TRS-80 Commands That Can Be Ignored

CLEAR	CSNG	DEFSNG
CDBL	DEFDBL	DEFSTR
CINT	DEFINT	FRE

Commands Difficult to Convert to TI BASIC

TRS-80	TI BASIC
IF . . THEN . . ELSE	IF . . THEN . . ELSE* refer to line numbers
POINT	CALL CHAR CALL GCHAR
POKE (graphics)	CALL CHAR CALL HCHAR
PRINT AT	FOR . . ASC . . CALL HCHAR . . NEXT PRINT . . FOR . . PRINT " " . . NEXT
RESET	CALL CHAR CALL HCHAR
SET	CALL CHAR CALL HCHAR

* Improved capability with multi-line statements in Extended BASIC.

Commands Not Available In TI BASIC**

ERL	PEEK	STRING$
ERR	POKE	USR
ERROR	POS	VARPTR
ON ERROR	RESUME	PRINT USING

**Most of the equivalent commands are available in TI Extended BASIC.

but the speed was disappointing. In the TRS-80 version it was necessary to insert "dummy" FOR . . . NEXT loops to slow down the rotation of the wheels; the TI version, on the other hand, was too slow right from start.

The single enhancement I had made to the TRS-80 version was to have the wheels stop one at a time, to prevent giving away the final result of the pull during rotation. To keep the wheels moving at a constant speed on the TI-99/4, I included dummy counting loops as each wheel was stopped. In spite of its lack of speed, the richness of the TI-99/4 graphics made the TI BASIC program a more appealing simulation of real slot machine action than the TRS-80 version.

To summarize, if the program you want to convert is a number cruncher with few graphics, the conversion should go smoothly and result in a TI BASIC program which runs with speed roughly comparable to its TRS-80 cousin. But if the program involves the heavy use of graphics, expect to rewrite it. And if the program is poorly documented to boot, keep a bottle of aspirin handy. Furthermore, because of the limitations of the TI BASIC IF . . . THEN . . . ELSE, and the lack of a PRINT AT command, you can expect nearly every converted program to increase in length. On the plus side, however, the extended variable names available in TI BASIC make it possible to enhance the quality of the documentation and structure of the rewritten program.

One final note: TI's new *Extended BASIC* command module adds the PRINT AT and PRINT USING statements, the capability of controlling up to 28 moving objects simultaneously, improved IF . . . THEN . . . ELSE capability, and supports true subroutine definition (a significant aid in structuring programs). Although Extended BASIC probably won't alter the need for rewriting graphics programs, it should make the job a lot easier.

An Example of Code Translation From TRS-80 BASIC to TI BASIC

```
120 FOR I1=1 TO NI
130 IF IZ(I1)<>0 THEN PRINT "THIS ITEM
    DROPPED";ID$: GOTO 160
140 IF K$=K1$ THEN IF C1(I1)=0 GOTO 160
    ELSE C2=10*C1(I1)+200
150 DX=C3–C2
160 NEXT
```

Translates to:

```
120 FOR I1=1 TO NI
130 IF IZ(I1)=0 THEN 140
132 PRINT "THIS ITEM DROPPED";ID$
134 GOTO 160
140 IF K$<>K1$ THEN 150
142 IF C1(I1)=0 THEN 160
144 C2=10*C1(I1)+200
150 DX=C3–C2
160 NEXT I1
```

PLEASE DON'T FORGET TO RETURN THE QUESTIONNAIRE ON THE FRONT BIND-IN CARD.

Battle At Sea . . . from p. 26

```
1290 OS=22
1300 RETURN
1310 PR$=S5$
1320 LE=2
1330 S=5
1340 OS=28
1350 RETURN
1360 NEXT S
1370 CALL HCHAR(22,1,32,64)
1380 GOTO 2230
1390 L=LEN(PR$)
1400 SUS$="ENTER ROW,COL. FOR "&STR$(LE)&" SPACES"
1410 FOR X=1 TO LEN(SUS$)
1420 SU1$=SEG$(SUS$,X,1)
1430 CALL VCHAR(22,X+2,ASC(SU1$))
1440 NEXT X
1450 PR$=PR$&" SPACE"
1460 CALL HCHAR(23,2,32,30)
1470 FOR X=1 TO LEN(PR$)
1480 SU1$=SEG$(PR$,X,1)
1490 CALL VCHAR(23,X+2,ASC(SU1$))
1500 NEXT X
1510 FOR X=1 TO LE
1520 CALL HCHAR(23,20,35)
1530 CALL VCHAR(23,21,LE-X+49)
1540 CALL KEY(0,K1,ST)
1550 IF ST=0 THEN 1540
1560 IF K1<65 THEN 1590
1570 IF K1>74 THEN 1590
1580 GOTO 1610
1590 CALL SOUND(100,-2,2)
1600 GOTO 1540
1610 CALL VCHAR(23,23,K1)
1620 CALL KEY(0,KE,ST)
1630 IF ST=-1 THEN 1620
1640 CALL KEY(0,K2,ST)
1650 IF ST=0 THEN 1640
1660 IF K2<48 THEN 1690
1670 IF K2>57 THEN 1690
1680 GOTO 1710
1690 CALL SOUND(100,-2,2)
1700 GOTO 1640
1710 CALL VCHAR(23,24,K2)
1720 SH(S,X,1)=K1-64
1730 SH(S,X,2)=K2-47
*1735 IF P(K1-64,K2-47)>0 THEN 1590
1740 P(K1-64,K2-47)=S
1750 NEXT X
1760 GOSUB 5350
1770 IF SH(S,1,1)=SH(S,2,1)THEN 1800
1780 X2=1
1790 GOTO 1810
1800 X2=2
1810 FOR X3=1 TO LE
1820 F=0
1830 FOR X1=1 TO LE-X3
1840 IF SH(S,X1,X2)=0 THEN 1900
1850 IF SH(S,X1,X2)<SH(S,X1+1,X2)THEN 1900
1860 SW=SH(S,X1,X2)
1870 SH(S,X1,X2)=SH(S,X1+1,X2)
1880 SH(S,X1+1,X2)=SW
1890 F=1
1900 NEXT X1
1910 IF F=0 THEN 1930
1920 NEXT X3
1930 FOR X=1 TO LE-1
1940 IF SH(S,X,1)<>SH(S,X+1,1)-1 THEN 1970
1950 NEXT X
1960 GOTO 2060
1970 FOR X=1 TO LE-1
1980 IF SH(S,X,2)<>SH(S,X+1,2)-1 THEN 2010
1990 NEXT X
2000 GOTO 2060
2010 CALL SOUND(100,-2,2)
2020 FOR X=1 TO LE
2030 P(SH(S,X,1),SH(S,X,2))=0
2040 NEXT X
2050 GOTO 1460
2060 X=S
2070 FOR X1=1 TO 5
2080 IF SH(X,X1,1)=0 THEN 2180
2090 IF SH(X,1,1)=SH(X,2,1)THEN 2120
2100 OSA=1
2110 GOTO 2130
2120 OSA=0
2130 P(SH(X,X1,1),SH(X,X1,2))=X
2140 IF X>1 THEN 2170
2150 CALL VCHAR(SH(X,X1,1)+4,SH(X,X1,2)+17,
     95+X1+OS+((LE-1)*OSA))
2160 GOTO 2180
2170 CALL HCHAR(SH(X,X1,1)+4,SH(X,X1,2)+17,
     95+X1+OS+(LE*OSA))
2180 NEXT X1
2190 IF X>1 THEN 2220
2200 CALL HCHAR(SH(1,4,1)+4,SH(1,4,2)+17,
     97+((LE-1)*OSA))
2210 CALL HCHAR(SH(1,5,1)+4,SH(1,5,2)+17,
     99+((LE-1)*OSA))
2220 RETURN
2230 LE=4
2240 S=1
2250 GOSUB 2390
2260 LE=3
2270 S=2
2280 GOSUB 2390
2290 LE=2
2300 S=3
2310 GOSUB 2390
2320 LE=2
2330 S=4
2340 GOSUB 2390
2350 LE=1
2360 S=5
2370 GOSUB 2390
2380 GOTO 2610
2390 X2=INT(RND*2)+1
2400 IF X2=2 THEN 2440
2410 X=INT(RND*(10-LE))+1
2420 X1=INT(RND*10)+1
2430 GOTO 2460
2440 X=INT(RND*10)+1
2450 X1=INT(RND*(10-LE))+1
2460 ON X2 GOTO 2470,2540
2470 FOR Y=X TO X+LE
2480 IF O(Y,X1)>0 THEN 2390
2490 NEXT Y
2500 FOR Y=X TO X+LE
2510 O(Y,X1)=S
2520 NEXT Y
2530 RETURN
2540 FOR Y=X1 TO X1+LE
2550 IF O(X,Y)>0 THEN 2390
2560 NEXT Y
2570 FOR Y=X1 TO X1+LE
2580 O(X,Y)=S
2590 NEXT Y
2600 RETURN
2610 M1$="MY SHOT"
2620 M2$="YOUR SHOT"
2630 M3$="SCORE"
2640 M4$="COMPUTER"
2650 M5$="USER"
2660 M6$="YOU MISSED"
2670 M7$="I MISSED"
2680 M8$="**HIT**"
2690 GOTO 2780
2700 FOR V=1 TO 7
2710 CALL HCHAR(18,V+4,ASC(SEG$(M1$,V,1)))
2720 NEXT V
2730 RETURN
2740 FOR V=1 TO 9
2750 CALL HCHAR(21,V+4,ASC(SEG$(M2$,V,1)))
2760 NEXT V
2770 RETURN
2780 FOR X=1 TO 5
2790 CALL HCHAR(18,X+22,ASC(SEG$(M3$,X,1)))
2800 NEXT X
2810 FOR X=1 TO 8
2820 CALL HCHAR(19,X+15,ASC(SEG$(M4$,X,1)))
2830 NEXT X
2840 FOR X=1 TO 4
2850 CALL HCHAR(19,X+26,ASC(SEG$(M5$,X,1)))
2860 NEXT X
2870 T=1
2880 IF T=0 THEN 2910
2890 T=0
2900 GOTO 3180
2910 T=1
2920 CALL HCHAR(21,3,32,12)
2930 CALL HCHAR(22,3,32,7)
2940 GOSUB 2700
2950 IF W>0 THEN 3630
2960 RANDOMIZE
2970 X=INT(10*RND)+1
2980 X1=INT(10*RND)+1
2990 H=X
3000 H1=X1
3010 IF P(X,X1)=7 THEN 2960
3020 IF P(X,X1)=6 THEN 2960
3030 CALL HCHAR(19,6,H+64)
```

*** Note**

Line 1735 was inserted as a last-minute enhancement to the program just prior to press time. If you have been using the automatic NUM mode, please exit it (by pressing ENTER) to type in this line. Then go back into it (NUM 1740,10) for convenience in entering the rest of the program.

Battle At Sea . . .

```
3040 CALL HCHAR(19,7,H1+47)
3050 IF P(X,X1)>0 THEN 4460
3060 GOSUB 3100
3070 GOTO 2880
3080 P(X+10,X1)=7
3090 CALL HCHAR(23,1,32,32)
3100 P(X,X1)=6
3110 CALL SOUND(200,-6,2)
3120 CALL HCHAR(23,1,32,32)
3130 CALL VCHAR(X+4,X1+17,144)
3140 FOR Y=1 TO 8
3150 CALL VCHAR(23,12+Y,ASC(SEG$(M7$,Y,1)))
3160 NEXT Y
3170 RETURN
3180 CALL HCHAR(18,3,32,12)
3190 CALL HCHAR(19,3,32,7)
3200 GOSUB 2740
3210 CALL KEY(0,K1,ST)
3220 IF ST=0 THEN 3210
3230 IF K1<65 THEN 3210
3240 IF K1>74 THEN 3210
3250 CALL VCHAR(22,6,K1)
3260 CALL KEY(0,KE,ST)
3270 IF ST=-1 THEN 3260
3280 CALL KEY(0,K2,ST)
3290 IF ST=0 THEN 3280
3300 IF K2<48 THEN 3280
3310 IF K2>57 THEN 3280
3320 CALL VCHAR(22,7,K2)
3330 K3=K1-64
3340 K4=K2-47
3350 IF O(K3,K4)<6 THEN 3390
3360 CALL SOUND(50,110,2)
3370 CALL HCHAR(22,6,32,7)
3380 GOTO 3180
3390 IF O(K3,K4)=0 THEN 3500
3400 CALL SOUND(200,220,2,330,2,440,2,-6,2)
3410 CALL SOUND(400,110,2,220,2,330,2,-8,2)
3420 CALL VCHAR(K3+4,K4+5,136)
3430 SF=O(K3,K4)
3440 O(K3,K4)=7
3450 CALL HCHAR(23,1,32,32)
3460 FOR X2=1 TO 7
3470 CALL HCHAR(23,13+X2,ASC(SEG$(M8$,X2,1)))
3480 NEXT X2
3490 GOTO 4600
3500 CALL SOUND(200,-6,2)
3510 CALL HCHAR(23,1,32,32)
3520 O(K3,K4)=6
3530 FOR X2=1 TO 10
3540 CALL VCHAR(23,13+X2,ASC(SEG$(M6$,X2,1)))
3550 NEXT X2
3560 CALL VCHAR(K3+4,K4+5,144)
3570 GOTO 2880
3580 CH=1
3590 GOTO 4650
3600 CH=0
3610 ON SF GOSUB 1110,1160,1210,1260,1310
3620 IF DS(SF)=LE-1 THEN 3780
3630 IF H=10 THEN 3670
3640 IF P(H+1,H1)<>7 THEN 3660
3650 IF W>1 THEN 4260 ELSE 4060
3660 IF H=1 THEN 3720
3670 IF P(H-1,H1)<>7 THEN 3720
3680 IF W>1 THEN 4260 ELSE 4060
3690 W2=W
3700 W=W1
3710 GOTO 3560
3720 IF H1=10 THEN 3760
3730 IF P(H,H1+1)<>7 THEN 3750
3740 IF W>1 THEN 4060 ELSE 4260
3750 IF H1=1 THEN 3780
3760 IF P(H,H1-1)<>7 THEN 3780
3770 IF W>1 THEN 4060 ELSE 4260
3780 L1=INT(RND*2)+1
3790 ON L1 GOTO 3800,3880
3800 X2=INT(RND*2)+1
3810 ON X2 GOTO 3820,3850
3820 X2=1
3830 X3=0
3840 GOTO 3950
3850 X2=-1
3860 X3=0
3870 GOTO 3950
3880 X3=INT(RND*2)+1
3890 ON X3 GOTO 3900,3930
3900 X3=1
3910 X2=0
3920 GOTO 3950
3930 X3=-1
3940 X2=0
3950 IF H+X2>10 THEN 3780
3960 IF H+X2<1 THEN 3780
3970 IF H1+X3>10 THEN 3780
3980 IF H1+X3<1 THEN 3780
3990 IF P(H+X2,H1+X3)=6 THEN 3780
4000 IF P(H+X2,H1+X3)=7 THEN 3780
4010 X=H+X2
4020 X1=H1+X3
4030 IF P(X,X1)>0 THEN 4460
4040 GOSUB 3100
4050 GOTO 2880
4060 IF H=10 THEN 4160
4070 H=H+1
4080 IF P(H,H1)=7 THEN 4060
4090 IF P(H,H1)=6 THEN 4160
4100 X=H
4110 X1=H1
4120 IF P(X,X1)>0 THEN 4460
4130 GOSUB 3100
4140 H=H-1
4150 GOTO 2880
4160 IF H=1 THEN 4070
4170 H=H-1
4180 IF P(H,H1)=7 THEN 4160
4190 IF P(H,H1)=6 THEN 4060
4200 X=H
4210 X1=H1
4220 IF P(X,X1)>0 THEN 4460
4230 GOSUB 3100
4240 H=H+1
4250 GOTO 2880
4260 IF H1=10 THEN 4360
4270 H1=H1+1
4280 IF P(H,H1)=7 THEN 4260
4290 IF P(H,H1)=6 THEN 4360
4300 X=H
4310 X1=H1
4320 IF P(X,X1)>0 THEN 4460
4330 GOSUB 3100
4340 H1=H1-1
4350 GOTO 2880
4360 IF H1=1 THEN 4260
4370 H1=H1-1
4380 IF P(H,H1)=7 THEN 4360
4390 IF P(H,H1)=6 THEN 4260
4400 X=H
4410 X1=H1
4420 IF P(X,X1)>0 THEN 4460
4430 GOSUB 3100
4440 H1=H1+1
4450 GOTO 2880
4460 CALL VCHAR(4+X,17+X1,136)
4470 CALL HCHAR(23,1,32,32)
4480 GOSUB 2700
4490 FOR Z=1 TO LEN(M8$)
4500 CALL HCHAR(23,14+Z,ASC(SEG$(M8$,Z,1)))
4510 NEXT Z
4520 CALL SOUND(200,220,2,330,2,440,2,-8,2)
4530 CALL SOUND(300,110,0,220,0,330,0,-8,0)
4540 SF=P(X,X1)
4550 CALL VCHAR(19,6,X+64)
4560 CALL VCHAR(19,7,X1+47)
4570 P(X,X1)=7
4580 H=X
4590 H1=X1
4600 FOR X2=1 TO 5
4610 DS(X2)=0
4620 NEXT X2
4630 FOR X2=1 TO 10
4640 FOR X3=1 TO 10
4650 IF CH=1 THEN 4670
4660 IF T=0 THEN 4720
4670 IF P(X2,X3)=0 THEN 4760
4680 IF P(X2,X3)=6 THEN 4760
4690 IF P(X2,X3)=7 THEN 4760
4700 DS(P(X2,X3))=DS(P(X2,X3))+1
4710 GOTO 4760
4720 IF O(X2,X3)=0 THEN 4760
4730 IF O(X2,X3)=6 THEN 4760
4740 IF O(X2,X3)=7 THEN 4760
4750 DS(O(X2,X3))=DS(O(X2,X3))+1
4760 NEXT X3
4770 NEXT X2
4780 IF CH=1 THEN 3600
4790 W=0
4800 SCORE=0
4810 FOR Z4=1 TO 5
4820 ON Z4 GOSUB 1110,1160,1210,1260,1310
```

Continued on p. 87

Harried Housewife . . . from p. 28

```
780 CALL CHAR(157,"090A0C0808")
790 CALL CHAR(158,"A119070101")
800 CALL CHAR(159,"0")
810 GOSUB 3070
820 TIME=0
830 MATCH=0
840 DIM HH(16),WORK(16)
850 CALL CHAR(43,"FFFFFFFFFFFFFFFF")
860 CALL CHAR(44,"0")
870 CALL SOUND(4225,44000,30)
880 CALL SOUND(4,44000,30)
890 CALL CLEAR
900 CALL SCREEN(12)
910 CALL COLOR(2,6,9)
920 GOSUB 2040
930 GOSUB 3370
940 DATA 72,79,85,83,69,87,79,82,75
950 RESTORE 940
960 FOR Y=23 TO 31
970 READ GR
980 CALL HCHAR(2,Y,GR)
990 NEXT Y
1000 DATA 77,65,84,67,72,32,50
1010 RESTORE 1000
1020 FOR Y=23 TO 29
1030 READ GR
1040 CALL HCHAR(5,Y,GR)
1050 NEXT Y
1060 DATA 76,69,84,84,69,82,83
1070 RESTORE 1060
1080 FOR Y=23 TO 29
1090 READ GR
1100 CALL HCHAR(6,Y,GR)
1110 NEXT Y
1120 CALL COLOR(8,7,1)
1130 CALL HCHAR(8,25,95)
1140 CALL HCHAR(8,27,95)
1150 CALL KEY(0,K1,ST)
1160 IF K1=83 THEN 1830
1170 IF K1<65 THEN 1150
1180 IF K1>80 THEN 1150
1190 CALL HCHAR(8,25,K1)
1200 SS=1
1210 ON (K1-64)GOSUB 5350,5390,5430,5470,5510,5550,
     5590,5630,5670,5710,5750,5790,5830,5870,5910,
     5950
1220 IF CH(1)<>0 THEN 1250
1230 GOSUB 4430
1240 GOTO 1260
1250 ON CH(1)GOSUB 2290,2400,2480,2590,2680,2770,
     2870,2970
1260 CALL KEY(0,K2,ST)
1270 IF K2=83 THEN 1830
1280 IF K2<65 THEN 1260
1290 IF K2>80 THEN 1260
1300 IF K2=K1 THEN 1260
1310 CALL HCHAR(8,27,K2)
1320 SS=2
1330 ON (K2-64)GOSUB 5350,5390,5430,5470,5510,5550,
     5590,5630,5670,5710,5750,5790,5830,5870,5910,
     5950
1340 IF CH(2)<>0 THEN 1380
1350 GOSUB 4430
1360 GOTO 1400
1370 IF CH(1)=0 THEN 1400
1380 ON CH(2)GOSUB 2290,2400,2480,2590,2680,2770,
     2870,2970
1390 IF CH(1)=CH(2)THEN 1620
1400 TIME=TIME+1
1410 IF TIME=10 THEN 1550
1420 IF TIME=12 THEN 1440
1430 GOTO 4480
1440 CALL HCHAR(22,2,32,31)
1450 DATA 79,72,32,78,79,33,32,75,73,68,83,32,65,
     82,69,32,72,79,77,69,33
1460 RESTORE 1450
1470 FOR Y=3 TO 23
1480 READ GR
1490 CALL HCHAR(24,Y,GR)
1500 NEXT Y
1510 GOSUB 2040
1520 GOSUB 3370
1530 TIME=0
1540 GOTO 4480
1550 DATA 83,80,69,69,68,59,32,75,73,68,83,32,87,
     73,76,76,32,66,69,32,72,79,77,69,32,83,79,
     79,78,33
1560 RESTORE 1550
1570 FOR Y=2 TO 31
1580 READ GR
1590 CALL HCHAR(22,Y,GR)
1600 NEXT Y
1610 GOTO 4480
1620 CALL SOUND(1000,440,2)
1630 MATCH=MATCH+1
1640 IF MATCH<>1 THEN 1710
1650 DATA 70,73,78,73,83,72,69,68,58
1660 RESTORE 1650
1670 FOR Y=23 TO 31
1680 READ GR
1690 CALL HCHAR(11,Y,GR)
1700 NEXT Y
1710 X=MATCH+9
1720 Y=26
1730 ON CH(1)GOSUB 3610,3700,3780,3870,4030,4110,
     4250,4340
1740 HH(K1-64)=0
1750 HH(K2-64)=0
1760 IF MATCH<>8 THEN 1400
1770 DATA 67,76,69,65,78,32,72,79,85,83,69,33,33
1780 RESTORE 1770
1790 FOR Y=3 TO 27 STEP 2
1800 READ GR
1810 CALL HCHAR(24,Y,GR)
1820 NEXT Y
1830 CALL HCHAR(8,25,83)
1840 GOSUB 3570
```

Harried Housewife . . .

```
1850 FOR S=1 TO 16
1860 SS=3
1870 ON S GOSUB 5350,5390,5430,
     5470,5510,5550,5590,5630,
     5670,5710,5750,5790,5830,
     5870,5910,5950
1880 ON CH(SS)GOSUB 2290,2400,
     2480,2590,2680,2770,2870,
     2970
1890 NEXT S
1900 CALL HCHAR(21,3,32,6)
1910 CALL HCHAR(22,2,32,31)
1920 IF MATCH<>8 THEN 1970
1930 FOR X=2 TO 8
1940 CALL HCHAR(X,23,32,9)
1950 NEXT X
1960 GOTO 2030
1970 DATA 72,79,85,83,69,87,79,
     82,75,32,78,69,86,69,82,32,
     69,78,68,83,33
1980 RESTORE 1970
1990 FOR Y=3 TO 23
2000 READ GR
2010 CALL HCHAR(24,Y,GR)
2020 NEXT Y
2030 GOTO 2030
2040 FOR Z=1 TO 11 STEP 10
2050 FOR X=Z TO Z+4
2060 FOR Y=2 TO 12 STEP 10
2070 CALL SOUND(100,1047,2)
2080 CALL HCHAR(X,Y,43,5)
2090 CALL HCHAR(X,Y+5,44,5)
2100 CALL SOUND(100,523,2)
2110 CALL HCHAR(X+5,Y,44,5)
2120 CALL HCHAR(X+5,Y+5,43,5)
2130 NEXT Y
2140 NEXT X
2150 NEXT Z
2160 DATA 3,4,3,9,3,14,3,19,8,4,
     8,9,8,14,8,19,13,4,13,9,13,
     14,13,19,18,4,18,9,18,14,
     18,19
2170 RESTORE 2160
2180 FOR CC=65 TO 80
2190 READ X,Y
2200 CALL HCHAR(X,Y,CC)
2210 NEXT CC
2220 CALL HCHAR(21,3,83)
2230 CALL HCHAR(21,4,61)
2240 CALL HCHAR(21,5,83)
2250 CALL HCHAR(21,6,84)
2260 CALL HCHAR(21,7,79)
2270 CALL HCHAR(21,8,80)
2280 RETURN
2290 CALL HCHAR(X-1,Y,96)
2300 CALL HCHAR(X-1,Y-1,103)
2310 CALL HCHAR(X-1,Y+1,97)
2320 CALL HCHAR(X,Y-1,98)
2330 CALL HCHAR(X,Y,99)
2340 CALL HCHAR(X,Y+1,100)
2350 CALL HCHAR(X+1,Y-1,101)
2360 CALL HCHAR(X+1,Y,102)
2370 CALL HCHAR(X+1,Y+1,103)
2380 GOSUB 3650
2390 RETURN
2400 CALL HCHAR(X-1,Y-1,104,2)
2410 CALL HCHAR(X,Y,109)
2420 CALL HCHAR(X-1,Y+1,105)
2430 CALL HCHAR(X,Y-1,106)
2440 CALL HCHAR(X,Y+1,107)
2450 CALL HCHAR(X+1,Y-1,108,3)
2460 GOSUB 3740
2470 RETURN
2480 CALL HCHAR(X-1,Y-1,112)
2490 CALL HCHAR(X-1,Y,113)
2500 CALL HCHAR(X-1,Y+1,114)
2510 CALL HCHAR(X,Y-1,115)
2520 CALL HCHAR(X,Y,116)
2530 CALL HCHAR(X,Y+1,117)
2540 CALL HCHAR(X+1,Y-1,119)
2550 CALL HCHAR(X+1,Y+1,119)
2560 CALL HCHAR(X+1,Y,118)
2570 GOSUB 3820
2580 RETURN
2590 CALL HCHAR(X-1,Y-1,120)
2600 CALL HCHAR(X-1,Y,121)
2610 CALL HCHAR(X-1,Y+1,122)
2620 CALL HCHAR(X,Y-1,123)
2630 CALL HCHAR(X,Y,124)
```

```
2640 CALL HCHAR(X,Y+1,125)
2650 CALL HCHAR(X+1,Y-1,126,3)
2660 GOSUB 3980
2670 RETURN
2680 CALL HCHAR(X,Y-1,128)
2690 CALL HCHAR(X-1,Y-1,133,3)
2700 CALL HCHAR(X+1,Y+1,133)
2710 CALL HCHAR(X,Y,129)
2720 CALL HCHAR(X,Y+1,130)
2730 CALL HCHAR(X+1,Y-1,131)
2740 CALL HCHAR(X+1,Y,132)
2750 GOSUB 4070
2760 RETURN
2770 CALL HCHAR(X-1,Y-1,136)
2780 CALL VCHAR(X-1,Y+1,142,2)
2790 CALL HCHAR(X+1,Y-1,142)
2800 CALL HCHAR(X-1,Y,137)
2810 CALL HCHAR(X,Y-1,138)
2820 CALL HCHAR(X,Y,139)
2830 CALL HCHAR(X+1,Y,140)
2840 CALL HCHAR(X+1,Y+1,141)
2850 GOSUB 4210
2860 RETURN
2870 CALL HCHAR(X-1,Y-1,144)
2880 CALL HCHAR(X-1,Y,149,2)
2890 CALL HCHAR(X+1,Y,149)
2900 CALL HCHAR(X,Y-1,145)
2910 CALL HCHAR(X,Y,146)
2920 CALL HCHAR(X,Y+1,147)
2930 CALL HCHAR(X+1,Y-1,148)
2940 CALL HCHAR(X+1,Y+1,148)
2950 GOSUB 4290
2960 RETURN
2970 CALL HCHAR(X-1,Y-1,152)
2980 CALL HCHAR(X-1,Y,153)
2990 CALL HCHAR(X-1,Y+1,154)
3000 CALL HCHAR(X,Y-1,155)
3010 CALL HCHAR(X,Y,156)
3020 CALL HCHAR(X+1,Y-1,157)
3030 CALL HCHAR(X+1,Y,158)
3040 CALL VCHAR(X,Y+1,159,2)
3050 GOSUB 4380
3060 RETURN
3070 X=3
3080 Y=5
3090 GOSUB 2290
3100 X=4
3110 Y=16
3120 GOSUB 2400
3130 Y=27
3140 GOSUB 2480
3150 X=8
3160 Y=7
3170 GOSUB 2590
3180 X=10
3190 Y=26
3200 GOSUB 2680
3210 X=17
3220 GOSUB 2770
3230 X=16
3240 Y=15
3250 GOSUB 2870
3260 X=15
3270 Y=6
3280 GOSUB 2970
3290 CALL SOUND(300,494,2,196,7)
3300 CALL SOUND(200,440,2)
3310 CALL SOUND(200,392,2)
3320 CALL SOUND(300,440,2,185,8)
3330 CALL SOUND(200,392,3)
3340 CALL SOUND(200,370,3)
3350 CALL SOUND(1000,392,3,165,9)
3360 RETURN
3370 FOR Z=1 TO 8
3380 WORK(Z)=Z
3390 WORK(Z+8)=Z
3400 NEXT Z
3410 RANDOMIZE
3420 FOR R=1 TO 16
3430 RR=INT(16*RND)+1
3440 IF WORK(RR)=0 THEN 3430
3450 HH(R)=WORK(RR)
3460 WORK(RR)=0
3470 NEXT R
3480 FOR R=1 TO 16
3490 WORK(R)=HH(R)
3500 NEXT R
3510 MATCH=0
3520 FOR X=11 TO 18
3530 CALL HCHAR(X,23,32,9)
3540 NEXT X
```

```
3550 CALL HCHAR(24,3,32,22)
3560 RETURN
3570 FOR R=1 TO 16
3580 HH(R)=WORK(R)
3590 NEXT R
3600 RETURN
3610 CALL COLOR(9,15,7)
3620 CALL COLOR(9,7,15)
3630 CALL COLOR(9,15,7)
3640 CALL COLOR(9,7,15)
3650 CALL HCHAR(X+2,Y-1,68)
3660 CALL HCHAR(X+2,Y,85)
3670 CALL HCHAR(X+2,Y+1,83)
3680 CALL HCHAR(X+2,Y+2,84)
3690 RETURN
3700 CALL COLOR(10,12,13)
3710 CALL COLOR(10,13,12)
3720 CALL COLOR(10,12,13)
3730 CALL COLOR(10,13,12)
3740 CALL HCHAR(X+2,Y-1,83)
3750 CALL HCHAR(X+2,Y,69)
3760 CALL HCHAR(X+2,Y+1,87)
3770 RETURN
3780 CALL COLOR(11,11,14)
3790 CALL COLOR(11,14,11)
3800 CALL COLOR(11,11,14)
3810 CALL COLOR(11,14,11)
3820 CALL HCHAR(X+2,Y-1,87)
3830 CALL HCHAR(X+2,Y,65)
3840 CALL HCHAR(X+2,Y+1,83)
3850 CALL HCHAR(X+2,Y+2,72)
3860 RETURN
3870 CALL COLOR(12,3,16)
3880 CALL COLOR(12,16,3)
3890 CALL COLOR(12,3,16)
3900 CALL COLOR(12,16,3)
3910 CALL HCHAR(X+2,25,68)
3920 CALL HCHAR(X+2,26,73)
3930 CALL HCHAR(X+2,27,83)
3940 CALL HCHAR(X+2,28,72)
3950 CALL HCHAR(X+2,29,69)
3960 CALL HCHAR(X+2,30,83)
3970 RETURN
3980 CALL HCHAR(X+2,Y-1,68)
3990 CALL HCHAR(X+2,Y,73)
4000 CALL HCHAR(X+2,Y+1,83)
4010 CALL HCHAR(X+2,Y+2,72)
4020 RETURN
4030 CALL COLOR(13,12,7)
4040 CALL COLOR(13,7,12)
4050 CALL COLOR(13,12,7)
4060 CALL COLOR(13,7,12)
4070 CALL HCHAR(X+2,Y-1,67)
4080 CALL HCHAR(X+2,Y,79,2)
4090 CALL HCHAR(X+2,Y+2,75)
4100 RETURN
4110 CALL COLOR(14,8,5)
4120 CALL COLOR(14,5,8)
4130 CALL COLOR(14,8,5)
4140 CALL COLOR(14,5,8)
4150 CALL HCHAR(X+2,25,86)
4160 CALL HCHAR(X+2,26,65)
4170 CALL HCHAR(X+2,27,67)
4180 CALL HCHAR(X+2,28,85,2)
4190 CALL HCHAR(X+2,30,77)
4200 RETURN
4210 CALL HCHAR(X+2,Y-1,86)
4220 CALL HCHAR(X+2,Y,65)
4230 CALL HCHAR(X+2,Y+1,67)
4240 RETURN
4250 CALL COLOR(15,16,15)
4260 CALL COLOR(15,15,16)
4270 CALL COLOR(15,16,15)
4280 CALL COLOR(15,15,16)
4290 CALL HCHAR(X+2,Y-1,83)
4300 CALL HCHAR(X+2,Y,72)
4310 CALL HCHAR(X+2,Y+1,79)
4320 CALL HCHAR(X+2,Y+2,80)
4330 RETURN
4340 CALL COLOR(16,16,3)
4350 CALL COLOR(16,3,16)
4360 CALL COLOR(16,16,3)
4370 CALL COLOR(16,3,16)
4380 CALL HCHAR(X+2,Y-1,73)
4390 CALL HCHAR(X+2,Y,82)
4400 CALL HCHAR(X+2,Y+1,79)
4410 CALL HCHAR(X+2,Y+2,78)
4420 RETURN
4430 CALL HCHAR(X,Y-1,68)
4440 CALL HCHAR(X,Y,79)
```

Continued on p. 87

ON-LINE INFORMATION RETRIEVAL

This article has been excerpted from **Information Brokering : A State-of-the-Art Report,** *By Gary M. Kaplan, Copyright 1980 & 1981, Emerald Valley Publishing Co., Eugene, OR.*

Why Use It?

Computerized information retrieval—a fairly recent development that didn't achieve widespread acceptance until the late 1970's—is undoubtedly the most important factor in the emergence of information brokering as a viable business opportunity for most of its current practitioners (the database search specialists, mentioned earlier). It is a fast, cost-effective method for retrieving bibliographic references, full text documents and files, and numerical data—all of it on an extremely diverse range of topics. Although the process goes by a variety of names—i.e., online searching, computer-assisted literature searching, online retrieval, on-line bibliographic searching, on-line non-bibliographic searching, computer database searching, etc. — the principal facts are that:

- The process takes a fraction of the time of searching by manual methods, is less tiring, and is often cheaper in terms of a researcher's time.
- The electronic databases are usually more comprehensive, more deeply indexed, and more frequently and readily updated than their printed counterparts.
- An on-line interactive search can be re-directed (expanded or narrowed) instantly on the basis of results coming in.
- Certain strategies (combining terms and concepts with Boolean operators, explained later) that are possible with a computer search are difficult or impossible with manual methods.
- Searching can be done from the convenience of your own home or office, or even a client's office (if you have a portable terminal) through a simple hook-up to virtually any telephone.

Types of Databases

1. Bibliographic Databases — are the "machine-readable" (searchable by computer) counterparts of printed abstracts and indexes, but are not as limited in size or flexibility. They contain references to (and sometimes brief summaries of) "the literature" of numerous fields. Armed with the appropriate references (periodical names, issue dates, and article titles/authors/pagination) on specific topics of interest, it's an easy matter to get copies of the actual articles that contain (hopefully) the information wanted. (In fact, some vendors are now offering the option of ordering printed or microfilm documents and articles while on-line, directly from the searcher's terminal—providing quick access to a "library" of literally millions of separate titles in a full spectrum of diverse subject areas. Additionally, there's a substantial time savings: By typing in the order and shipment information yourself, right after identifying the relevant titles on-line, the order gets immediately routed to the appropriate document delivery service. You pay this delivery service a set fee per document (which you can mark up and charge your clients that request this additional service) and typically cut a week from the normal delivery time.

2. Numeric Databases — are the machine-readable counterparts of handbooks and tables, but are not as limited in currency, accuracy, and completeness. These types of files carry information which is useful in and of itself; they are not surrogates which point to other documents (although some do indeed refer a search to other sources). Data may also be manipulated on-line to do analyses, generate reports, and make predictions. Most of these databases are in the areas of business, finance and economics (e.g., value of product shipments, manufacturing data, securities and exchange rates, forecasts, and currency exchange rates). There are some databases in the social sciences and humanities containing demographic information (e.g., census statistics). Presently, not very many numeric databases in science and technology are publicly available.

3. Referral Databases — are the machine-readable counterparts of directories; they provide references to individuals, institutions, and projects with "know-how" availability or need. These files are usually much more current than printed versions since they are easier to update at much shorter intervals.

4. Miscellaneous Databases — that don't fall under the three previous groupings are in areas such as regulatory, legislative, and judicial decisions and status reports, as well as charts and maps.

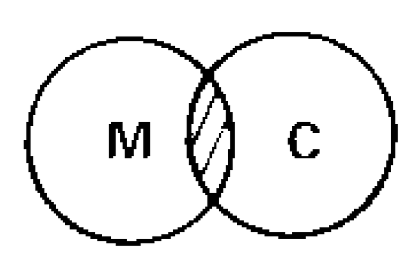
Note:
Several directories containing descriptions of the individual databases of these four groups are available in most libraries that perform on-line searches. Ask to see (1) *Computer-Readable Data Bases : A Directory and Data Sourcebook*, (2) *Directory of On-Line Information Resources*, or (3) *Directory of Online Databases.*

The Power of Boolean Logic

Most on-line search systems use Boolean (named after 19th century mathematician George Boole) operators. These are simply the basic combining operations performed on groups of documents or articles by the three symbols **AND**, **OR**, and **NOT**. The best way to illustrate the power of these search operators is to show their effects on different search terms through simple diagrams (called Venn diagrams). In all of these that follow, the shaded portions represent the items that are retrieved as a result of the combination.

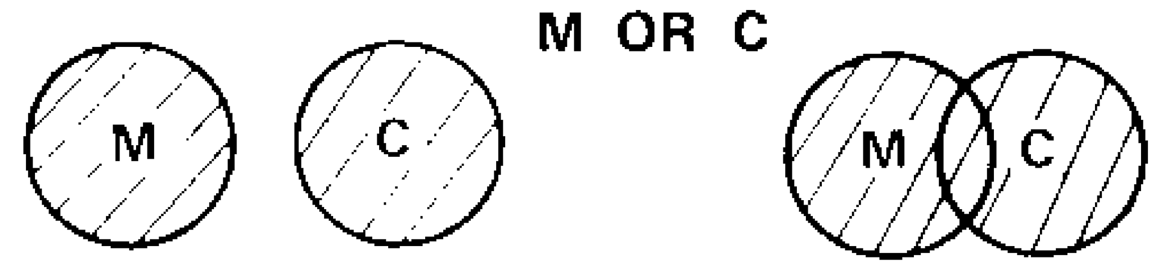
(1) The **AND** operator retrieves the articles that have *both* children(C) and the consumption of milk(M) *in common* (e.g., the title "Consumption of Milk in Pre-school Children").

M AND C



(2) The **OR** operator retrieves the articles that *may or may not* have children(C) and consumption of milk(M) *in common* (e.g., the titles "Milk Sales Are Up 9% From Last Year" and "Public School Enrollment Declines" are both retrieved but have no common elements).

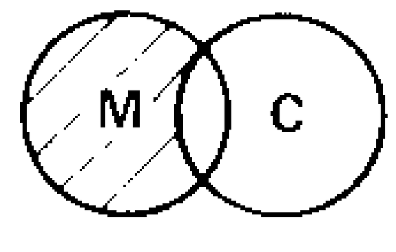
M OR C



No common elements   Some common elements

(3) The **AND NOT** operator retrieves the articles that have to do with consumption of milk(M) but that also *simultaneously exclude* those with specific references to children(C) — e.g., "Adults Drinking 150% More Milk Than A Decade Ago."

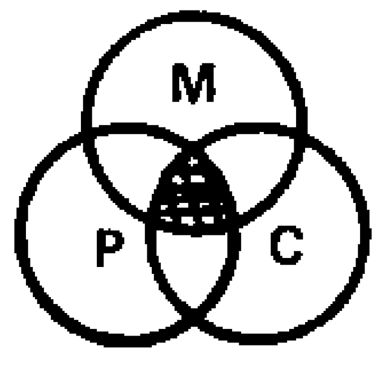
M AND NOT C



Now if we want to make the search more specific, let's say that we need articles concerned with cases of milk poisoning in children (where poisoning is represented by "P") the Boolean operators would be

M AND C AND P

The small shaded area in the center represents the retrieved articles.



In this case, of course, you could do the search manually – i.e., through scanning the appropriate printed indexes – and might be able to retrieve a few articles related to this topic before fatigue set in. But consider the following search in the field of medicine, where a client needs to know the effects of blood coagulation disorders on heart disease in children. Here, you could just check printed indexes for the three subject headings "blood coagulation disorders," "heart disease," and "children," and hopefully you'd find something. But if your client needed (and expected) *all* relevant information (to the extent that "all" is possible), you'd need to expand the search terms to include the other subject headings given in the cross references. Take for example:

Blood coagulation disorders (B1)
- See also:
  - anticoagulants (B2)
  - blood disease (B3)
  - blood coagulation (B4)
  - blood platelets (B5)
  - blood viscosity (B6)
  - hematologic diseases (B7)
  - hemmagglutination (B8)
  - hemostatics (B9)
  - thrombocytes (B10)

heart disease (H1)
- see also:
  - cardiac arrest (H2)
  - cardiology (H3)
  - cardiovascular disease (H4)
  - heart auculation (H5)

children (C1)
- see also:
  - infants (C2)

Here, you'd have to check under 17 subject headings in the printed index (B1 to B10, plus H1 to H5, plus C1 & C2) and find all the journal articles that are related to *100 possible combinations* (10x5x2=100) of the 17 headings.

These are combinations such as

B1 & H1 & C1 (blood coagulation disorders, heart disease, & children)
B2 & H1 & C1 (anticoagulants, heart disease, & children)
B8 & H2 & C1 (hemmagglutination, cardiac arrest, & children)
B10 & H5 & C2 (thrombocytes, heart auculation, & infants)
plus 96 other combinations to search for!

I think you can start to see how a computer search with Boolean operations can handle this awesome task much more effectively. After entering all the subject headings, you would command the computer to combine and search for the 100 expanded terms according to the following logic : (B1 OR B2 OR B3 OR B4 OR B5 OR B6 OR B7 OR B8 OR B9 OR B10) **AND** (H1 OR H2 OR H3 OR H4 OR H5) **AND** (C1 OR C2). Then, in a flash, the computer at the other end tells you how many "hits" (articles) it found in the database. You can command it to list these titles and then decide for which ones you'd like the complete bibliographic citations (and abstracts, if available) printed out—either on your terminal/printer, or on their high-speed printer and mailed to you (a much cheaper option if the list is lengthy and the delivery time isn't critical). In effect, the computer has produced a *very specialized and comprehensive* reading list that a human working with manual methods alone, couldn't likely duplicate.

> **Note :**
> **Interested readers who would like to know more about the field of information brokering may purchase the complete spiral-bound monograph from which this article has been excerpted *(Information Brokering: A State-of-the-Art Report)* for the special reduced price of $7. (Regular price is $10.) For postage information, see the bound-in order card in the back of this magazine.**

A Sample On–Line Search

● **QUESTION:**

What articles are available on attitudes toward hiring the handicapped?

```
FILE INFORM

SS   1: ALL HANDICAP:/IT (118)
SS   2: ALL HIR: OR ALL EMPLOY: OR ALL PLAC: (20704)

SS   3: ALL ATTITUD: (3087)
SS   4: 1 AND 2 AND 3 (4)

SS   5/C?
USER:
PRT TI 4

-1-
TI - EMPLOYEE SERVICES - A STUDENT PERSPECTIVE

-2-
TI - IBM CALLED EXCEPTION - DP INDUSTRY HIT FOR ATTITUDE
     ON HANDICAPPED

-3-
TI - NOT ALL ILLNESSES ARE HANDICAPS

-4-
TI - TOTAL PERSON TRAINING

SS   6 /C?
USER:
PRT FULL 1 SKIP 1
```

Name of database

Number of article titles found within the database that are indexed under each of the subject terms (or synonyms) given in the search statements

Number of articles found (4) within the database that are indexed under all the terms corresponding to search statements 1–3

Command to print the titles of the 4 articles

The article titles are printed

Command to print full information about the most likely title

```
PROG:

-2-
AN - 77-05588
TI - IBM CALLED EXCEPTION - DP INDUSTRY HIT FOR ATTITUDE
     ON HANDICAPPED
AU - WISEMAN, TONI
SO - COMPUTERWORLD (CMPWAB, COW), V11 N14, P15, ISSN 0010-4841,
     APR. 4, 1977
DT - J (JOURNAL)
LA - ENGLISH

IT - ADDITIONS; AREAS; CASES; COMPANIES; COMPUTER PROGRAM
     MING; COMPUTERS; COSTS; DEAFNESS; DISABILITIES; EMPLOYERS;
     EXCEPTIONS; FIRMS; GOVERNMENT; HANDICAPPED; IBM-ARMONK
     NY, INDUSTRIES; MANAGEMENT; MANUFACTURERS; NEGATIVE;
     NOISE; PROGRAMMERS; SOLUTIONS; USERS

AB - WITH ONE EXCEPTION, THE DP INDUSTRY - BOTH USERS AND
     MANUFACTURERS - HAS DONE VERY LITTLE IN THE AREA OF
     EMPLOYING THE HANDICAPPED. IBM ALONE HAS BEEN 'IN THE
     FOREFRONT' ALMOST FROM ITS INCEPTION AS AN EMPLOYER
     OF THE HANDICAPPED. IN THE CASE OF SOME COMPUTER COM-
     PANIES THERE HAS BEEN ALMOST A NEGATIVE REACTION, A
     FEELING OF 'I'M NOT GOING TO DO ANYTHING UNTIL THE
     GOVERNMENT FORCES ME TO.' IN MANY CASES, HOWEVER, IT
     IS THE DISABILITY WHICH MAKES THE HANDICAPPED PERSON
     SUITED TO A JOB. A DEAF PERSON, FOR INSTANCE, WAS THE
     IDEAL SOLUTION TO ONE COMPANY'S NOISE PROBLEM IN ITS
     BURSTING ROOM. THE OPERATOR WAS TRAINED TO DO THE
     JOB AND THE FIRM SAVED THE COST OF INSTALLING SOUND-
     PROOFING. DEAF PEOPLE ARE ALSO LESS DISTRACTED BY
     OTHER NOISES AND HAVE BEEN SHOWN TO HAVE A HIGHER
     PROGRAMMING OUTPUT THAN PROGRAMMERS WHOSE HEARING
     IS UNIMPAIRED. IN ADDITION, ACCORDING TO THE LAW, THE
     HANDICAPPED PERSON MUST ALSO BE HIRED IN THE MANAGE-
     MENT AREA.
```

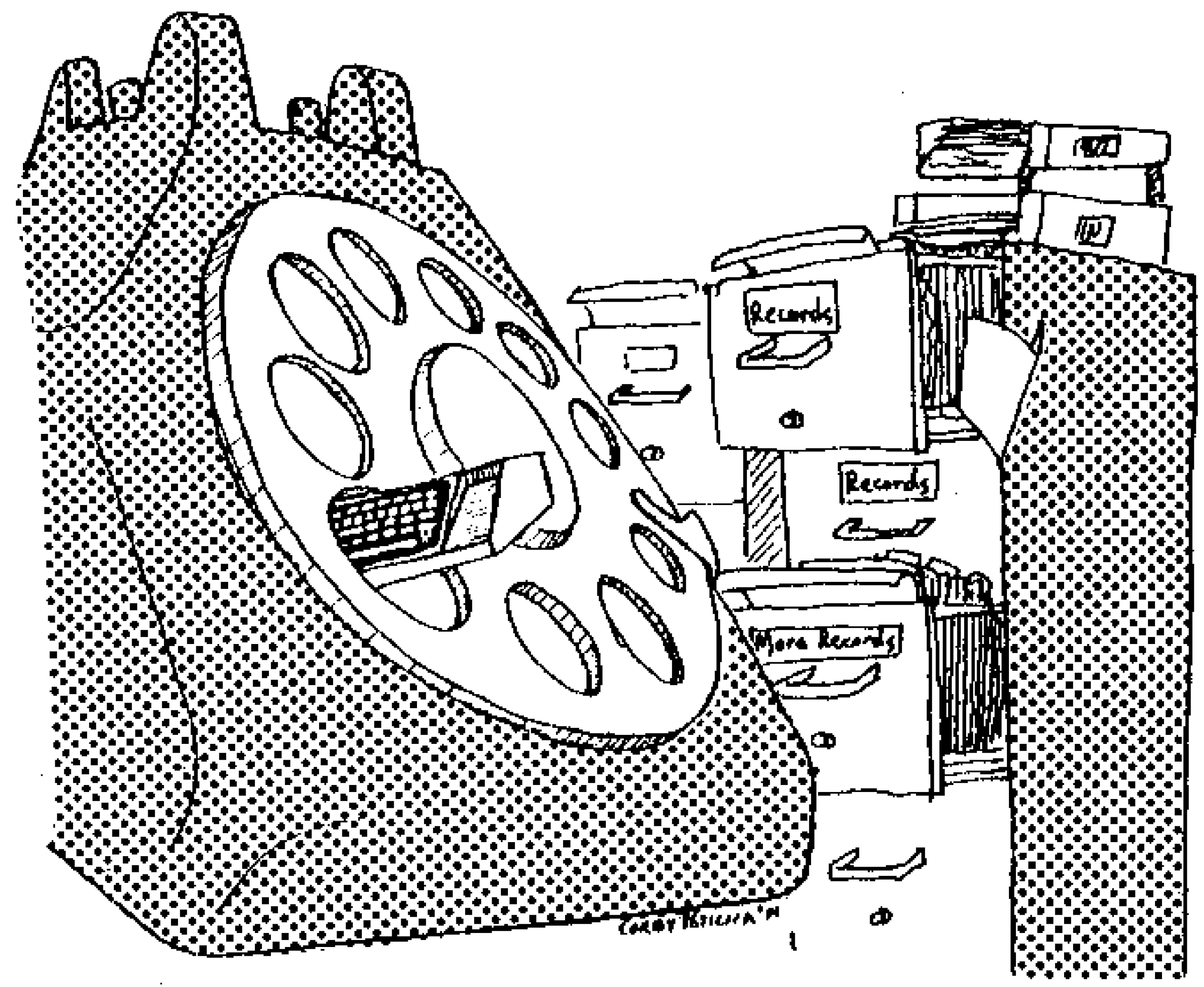
Full bibliographic information about requested article is printed

Terms under which the article was indexed

Full abstract of article

The Electronic Home Secretary

Computerized Filing and Telephone Dialing



By Malladi V. Subbaiah

Post-Doctoral Research Fellow
Department of Mechanical Engineering
301-46 CALTECH, Pasadena, CA 91125

Now that you have a personal computer, you've probably been looking for ways to use it around the house. When writing software for home applications, it's often possible to create a *general* program that functions in a variety of household situations. The program accompanying this article follows this design philosophy. With it, you can create a personal phone and address directory, time events (such as elapsed telephone connect time), have your computer dial or redial any number in your directory, and set up an inventory of household possessions for insurance and maintenance purposes. All this in standard 16K TI BASIC —with some room to spare for customizing the program according to your preference.

GENERAL DESCRIPTION OF THE PROGRAM

Data Entry

When the program is first RUN, the screen options give a user the choice of updating or using a previous data file saved on cassette or disk, or creating an entirely new data file for one of two options: (1) the phone and address directory, or (2) the houshold inventory. Both of these options also provide sub-options: For example, the program can draw on the data files to dial (by the dual-tone method) an appropriate phone number, sum the total cost in the inventory, and print hardcopy listings of each. The category names for the file organization are provided in the data statements 220 and 230.

The input data is stored in the arrays A1$, A2$, A3$, A4$ and A5$. A dimension of 60 is assigned to each of the arrays, and a maximum string length of 190 characters is allowed for each complete entry. Line 710 checks the validity of each data set. At this stage, the program also checks for dimension overflow and memory overflow (lines 480 and 810), and appropriate warning messages are displayed. These protection features prevent accidentally keying in excess data—a situation that would result in an error and program termination. Additionally, the cost category (A2$) in option 2 is designed to accept only numerical input so that you can conveniently carry out numerical operations on the data—for example, the total cost of possessions. And keep in mind that you can, of course, change the categories by altering the data in lines 220 and 230.

Sort Routine:

An efficient sort subroutine is presented in the program at line 2410. The routine employs a tree sort procedure which needs approximately 2*N*(Log_2N−1) comparisons to sort N entries. Since various versions of sorting routines have been previously published and are readily available [see for example, reference no. 2 at the end of this article, or any elementary book on numerical analysis], I won't discuss the mathematical details of the sorting procedure. Here, the sorting is based on the entries in the arrays A1$ (i.e., names or items in the default categories). The remaining arrays are appropriately rearranged to be consistent with the original data. The procedure is carried out without the use of any intermediate arrays, thereby saving on the core usage. Complete sorting and rearranging of 50 entries takes about 4 minutes.

Data Deletion and Alteration

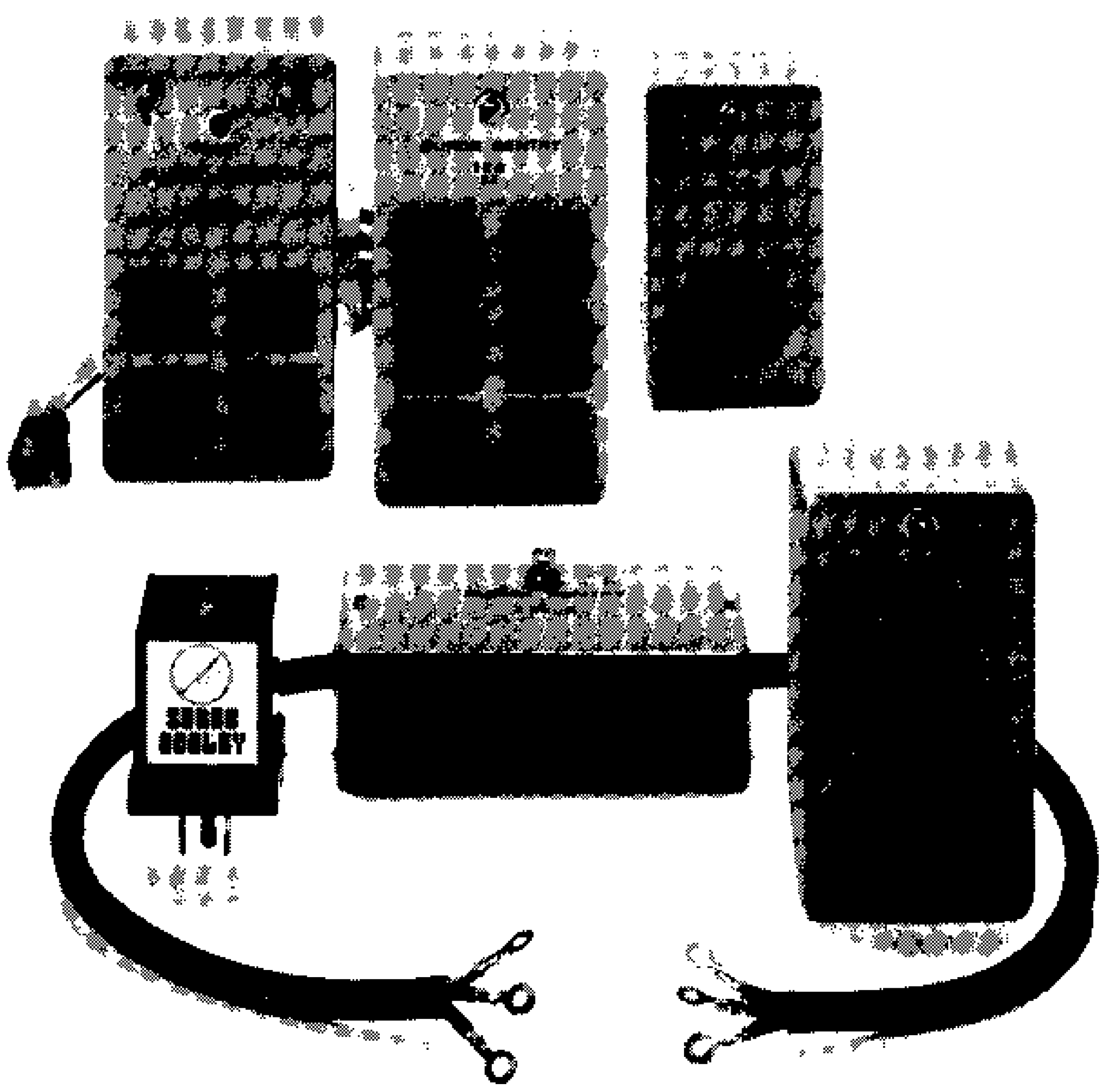
The subroutine at line 1010 updates any existing data set. You can access any particular entry by its serial number or by its name (or a segment of it). A search routine (line 1790) retrieves the data set with the specified name, or the next higher one if the name match is not exact. As previously described, validation of the altered data for allowable string length and memory overflow are carried out. At this stage, you have the option to move up or down in the list, search for a different entry, or finish the editing session. Any alteration of the entry title (i.e., A1$) causes the variable FLAG2 to be set equal to unity. This causes resorting of the data set before the directory can be displayed.

Display of the Directory

The program allows you to display the data directory in two formats. In the first format (at line 1420), a concise, quick-reference listing of the complete directory is provided. This includes name and phone number for the phonebook option, and item and cost for the inventory option.

In the second format, you can display all the data contained in any single entry. Access to individual entries is either by its serial number in the directory, or by a string search as discussed in the previous section.

Additionally, you can get a hardcopy listing of the entire directory (line 4280) through an RS232-compatible printer, or the TI thermal printer. The screen printing routine at line 4150 was used to get a hardcopy print-out of screen



displays for this article. This portion (lines 4150-4260) can be deleted without affecting the operation of the program.

Computerized Phone Dialing

Now let's look at how we can get the TI-99/4 to do Touch Tone dialing. Since the telephone company prohibits direct connection of any unapproved (by the FCC) user equipment to the phone line, the method we will have to use involves proximity: Dialing of the phone is accomplished simply by placing the microphone of the phone handset in front of the monitor speaker, without any direct connection to the phone lines.

Briefly, the Touch Tone system of telephone dialing operates by sending a specific pair of audio frequency tones over the voice channel of the phone line for each digit. The switching circuits at the telephone facility decode the tones and operate the appropriate circuits to make the connection. The tone pairs consist of a low frequency group (697 - 941 Hz) and a high frequency group (1209 - 1477 Hz) as shown in Figure 1. For example, to dial the number 5, we have to send the audio tones at 770 Hz and 1336 Hz simultaneously for a sufficiently long time to be recognized by the switching circuits. Also, there should be a sufficient gap between digits to have each digit registered individually. Although a 40 millisecond signal duration followed by a 40 millisecond silence should theoretically be adequate, I observed that a 150 - 200 millisecond signal duration and a gap of about 100 - 150 milliseconds is required for reliable operation with this system.

With the CALL SOUND (duration, frequency 1, volume 1, frequency 2, volume 2) command of TI BASIC, the TI-99/4 can generate the dual tones of Figure 1. In doing this, however, an interesting problem arises: If we examine the monitor's output on an oscilloscope, we can observe that the so called "pure tone" from the computer is, in fact, a *square wave* and is not a *sine wave*. By Fourier analysis, the square wave can be decomposed into its constituent sine waves. (Interested readers can refer to any elementary book of calculus for the details of the analysis.) To be specific, the output from CALL SOUND (100, 500, 1) is a square wave of 500 Hz for a 100 millisecond duration at the volume level 1. This is a combination of sine waves at 500 Hz, 1500 Hz, 2500 Hz and so on. This can pose a problem when we try to dial the first two members (i.e., 697 Hz and 770 Hz) of the low frequency group. The third harmonics of these frequencies, namely, 2091 Hz and 2310 Hz, are also recognized by the switching circuits, resulting in the rejection of the signal. The third harmonics of 852 Hz and 941 Hz seem to be outside the frequency response of the switching circuits and pose no problem.

There are several ways we can overcome this problem of dialing the digits 1 thru 6. One very simple and inexpensive way is to use a passive low-pass filter with a cut-off frequency of about 1.5 KHz in the audio line to the monitor, thereby attenuating the higher frequencies. Figure 2 shows a block diagram for the installation. The circuit for the filter which I built for under five dollars is shown in Figure 3.

HOW TO USE THE PROGRAM

Initial Set-Up

With a choice of **N** (for NO) for the **LOAD DATA?** option in Display 1, the program has you select either the Phone Directory or Household Inventory option. (If your choice was **Y**, and you loaded a file, one of the data elements on the file tells the program which option to branch to.) The data file is then keyed in, guided by the input prompts. The phone number can be entered with spaces and parentheses, if desired. The most recent entry can be re-entered by pressing **R** for the name (or item). You can terminate by pressing **E** or **END**; this causes the data to be sorted and returns you to the master selection list (Display 3).

Low Frequency Group \ High Frequency Group	1209 Hz	1336 Hz	1477 Hz
697 Hz	1	2	3
770 Hz	4	5	6
852 Hz	7	8	9
941 Hz	*	0	#

Figure 1. Basic Frequencies for the Two-Tone System of Telephone Dialing

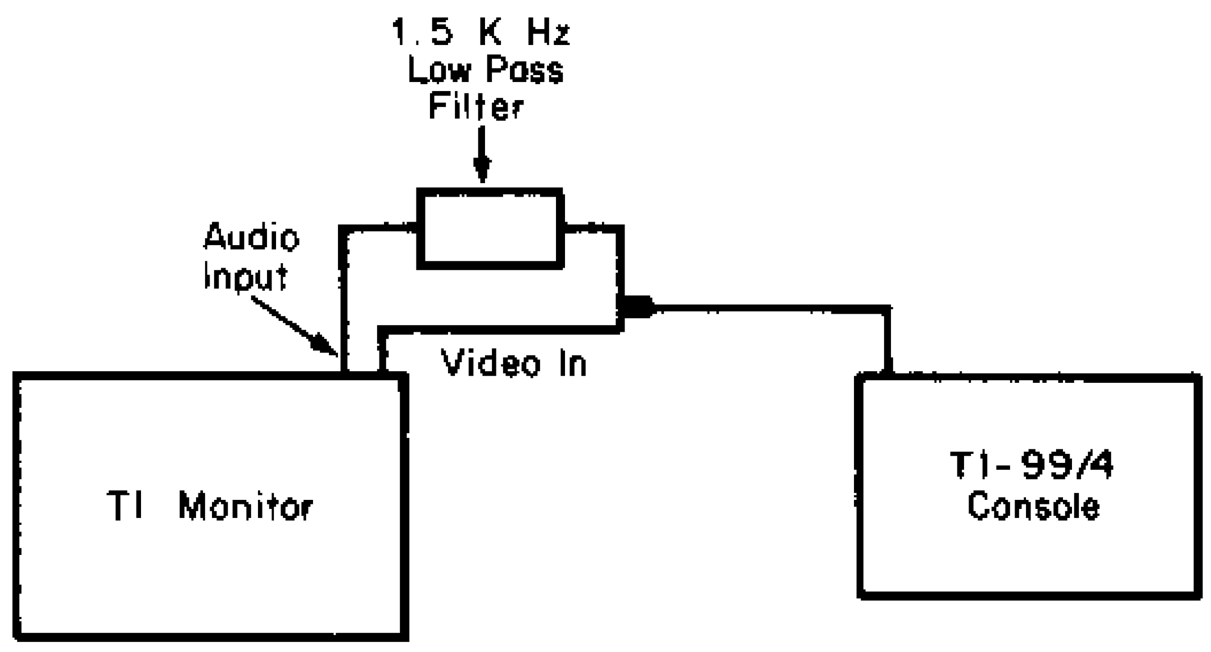


Figure 2. Schematic Layout of Filter Location

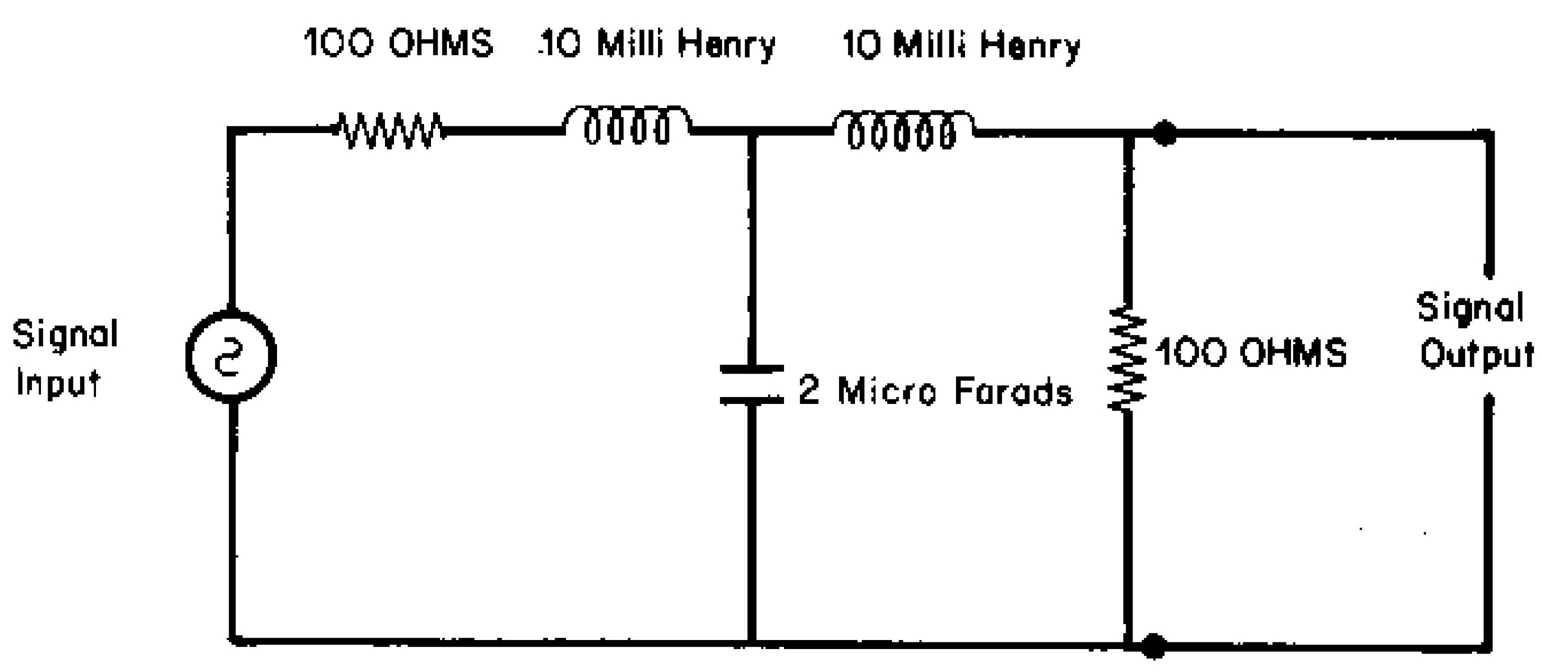


Figure 3. Circuit Diagram of the Filter

Note: On many touch-tone phone systems this filter will not be needed for correct dialing. We suggest you first try without it—Ed.

Load Previous Data File

To load a previously stored data file, we select Y for the LOAD DATA? option and follow the screen displays to operate the cassette player or disk. When loaded, the name of the data file, its size and the date of the previous revision will be displayed (Display 2); the program will then return you to the master selection list (Display 3).

Master Selection List and Its Functions

The master selection list (Display 3) provides access to the program's various options. A banner ***UPDATE DIRECTORY*** will be displayed if there has been any alteration of the data file since the last update. This should act as a constant reminder to save the revised version of the data on a cassette or disk. The different options of the master selection list are as follows:

Press 1: Select this to add any new entry to the data file. This leads to the data entry of Display 1.

Press 2: This leads to Display 4. You can access any individual entry by its serial number in the directory (from Display 5) or by a string search. Here, entering a null string (i.e., just pressing the ENTER key) for any category will leave it unaltered.

Press 3: This displays a short form of the directory as in Display 5. The display stops when the screen is filled. Pressing any key causes the remaining data to be displayed, or

```
LOAD DATA? (Y/N)          N
PHONE BOOK? (Y/N)         Y
ENTER
   E TO EXIT
   R TO REENTER
NAME:DOE ↵
PHONE: 987 6543 ↵
STREET:4321 NORTH SOUTH ST ↵
CITY & ZIP:HOLLYWOOD; CA99888 ↵
MISC:JOHN; DATE OF BIRTH JAN
  1 1921; WIFE MARY; CHILDREN
  JOE ; SUSAN; WEDD ANNIV FEB
  25; ↵
```

DISPLAY 1 INITIAL SET-UP

Note:

↵ = pressing **ENTER**, after the user's response

```
LOAD DATA (Y/N)      Y
ENTER
1. CS1
2. DISK 1
3. OTHER

*  REWIND CASSETTE TAPE     CS1
   THEN PRESS ENTER          ↵

*  PRESS CASSETTE PLAY      CS1
   THEN PRESS ENTER          ↵
INVENTORY - 1
LSIZE(3800)=1628
LAST UPDATE: MARCH 26 81
```

DISPLAY 2 LOAD PREVIOUS DATA FILE*

(FOR OPTION 1)

* *OPTION 2: ENTER FILE NAME*
OPTION 3: ENTER DEVICE NAME:

```
PRESS
1 - TO ADD MORE DATA
2 - TO ALTER THE DATA
3 - TO DISPLAY THE DIRECTORY
4 - TO DISPLAY ONE ENTRY
5 - TO USE THE DATA
6 - TO STORE DATA FILE
7 - FOR PRINTER LISTING
8 - TO END PROGRAM

*** UPDATE DIRECTORY ***
```

DISPLAY 3 MASTER SELECTION LIST

```
WHICH ONE; DOE ↵
ENTER
   NEW DATA AT CURSOR
   'D' TO DELETE THE ITEM
   'ENTER' FOR NO CHANGES

DOE?
987 6543;  (424) 987 6543 ↵
4321 NORTH SOUTH ST? ↵
HOLLYWOOD; CA99888? ↵
JOHN; DATE OF BIRTH JAN 1 19
21; WIFE MARY; CHILDREN JOE
$ SUSAN: WEDD ANNIV FEB 25;? ↵
```

DISPLAY 4 DATA ALTERATION

```
1. ARPACI JOE                321 1234
2. DOE               (424)   987 6543
3. DOE MARY          (424)   789 3456
4. MOORE N.            578   657 8901
5. NORTON P.                 356 4473
6.  OHSHIMA                  368 8714
7. SASTRY M.                 765 2345
8. SHIELD B.           654-789 4532
9. SHYAMALA                  206 6808
0. SUBBAIAH          (213)   356 4473
1. WONG V.                   256 3902
PRESS ANY KEY TO CONTINUE
```

DISPLAY 5 SHORT FORM DIRECTORY

```
  WHICH ONE?   DOE ↵
DOE
(424) 987 6543
4321 NORTH SOUTH ST
HOLLYWOOD; CA99888
JOHN; DATE OF BIRTH JAN 1  19
21; WIFE MARY; CHILDREN JOE
& SUSAN; WEDD ANNIV FEB 25;

PRESS
   E  TO LIST UP
   X  TO LIST DOWN
   S  TO SEARCH MORE

PRESS ANY KEY TO CONTINUE
```

DISPLAY 6 SINGLE ITEM DISPLAY

```
 DOE
(424) 987 6543
4321 NORTH SOUTH ST
HOLLYWOOD; CA99888'
JOHN; DATE OF BIRTH JAN 1  19
21; WIFE MARY; CHILDREN JOE
$ SUSAN; WEDD ANNIV FEB 25;
          1(424) 987 6543
PRESS
   R  TO REDIAL
   S  TO START STOPWATCH
   N  FOR NEW NUMBER
PRESS ANY KEY TO CONTINUE ↵
PRESS
   R  TO REDIAL
   S  TO START STOPWATCH
   N  FOR NEW NUMBER
PRESS ANY KEY TO CONTINUE    S
HOLD DOWN
R  TO DIAL AGAIN
ANY KEY TO CONTINUE
                        00:55
```

DISPLAY 7 PHONE DIALING AND STOPWATCH

```
TOTAL COST OF ALL THE ITEMS

     $$$$$$$$$$$$$$$$$$$
     $                 $
     $                 $
     $                 $
     $     7450.6      $
     $                 $
     $                 $
     $                 $
     $$$$$$$$$$$$$$$$$$$

PRESS ANY KEY TO CONTINUE
```

DISPLAY 8 TOTAL COST OF INVENTORY

```
ENTER   1. CS1
        2. DSK1
        3. OTHER

           YOUR CHOICE? ↵
TODAY'S DATE:  MARCH 7  1981 ↵
DIR. NAME: PHONE BOOK - 1 ↵
 *  REWIND CASSETTE TAPE     CS1
    THEN PRESS ENTER          ↵
 *  PRESS CASSETTE RECORD    CS1
    THEN PRESS ENTER          ↵
 *  PRESS CASSETTE STOP      CS1
    THEN PRESS ENTER          ↵
```

DISPLAY 9 SAVE DATA FILE

returns you to the master selection list if no more data is to be displayed.

Press 4: This produces a complete listing of a single entry (Display 6)—selectable by its serial number in the directory, or by a string search as in Display 4.

Press 5: This allows the program to use the data files when dialing/redialing in the phonebook option, or to obtain the total purchase cost of the inventory in the household inventory option. If you are in the phonebook option, the program will advance to Display 6. If the display is approved by pressing any key other than E, X and S, the computer dials the displayed phone number. In the beginning, you may have to adjust the volume control of your TV set or monitor for proper operation. The digits will be displayed one-by-one as they are dialed. If the total number of characters in the phone number is greater than or equal to 10, the routine recognizes it as a long distance call, and dials 1 at the beginning (Display 7). After getting familiar with the operation, you may want to reduce the time periods assigned in the CALL SOUND statements on lines 3540, 3580, 3590. You can redial the number by pressing **R**, start the stopwatch by pressing **S** (and quickly releasing the key), or select a new number using the choice **N**. Any other key (including a prolonged pressing of **S**) terminates the dialing session and the master selection screen will be displayed.

With the selection **S**, the stopwatch routine on line 3700 is activated. The elapsed time is displayed at the lower right-hand corner (Display 7). You can control the accuracy of the stopwatch by adjusting the time delay constants of the DATA statement in line 3320. Here, holding down **R** starts the dialing procedure all over again; pressing any other key returns you to the master selection list (Display 3).

If you are in the household inventory option, choice 5 of the master selection list will cause the program to calculate the total purchase cost (Display 8) of all the items in the data file. There's no adjustment here for inflation. This, however, could easily be done. For example, you could key in the consumer price index into the data file at the time of an item's purchase and scale the purchase cost with the current price index when evaluating the SUM (in the routine on line 3150). I felt, however, that this procedure would be rather involved for day-to-day use.

Press 6: This permits storing the data file on either cassette or disk. The computer asks (Display 9) for the title of the data file and the date of revision for future reference. This information will be displayed when you re-load the data for another session.

Press 7: This produces a hardcopy listing (with nine complete entries per page) on either the TI thermal printer, or a printer connected to the RS232 interface. The computer first asks you to verify that either the thermal printer or the RS232 interface is connected, to avoid the File-Error termination. As a precaution, alway SAVE the updated file on cassette or disk (option 6) *prior* to printing.

SUMMARY AND FINAL REMARKS

This program is capable of performing a wide variety of functions. I have shown you how to use it to maintain a computerized phone directory and do automatic dialing, as well as to maintain very flexible data files for day-to-day use in the home. Typical applications include an inventory of household valuables, a record of credit cards and bank accounts, lists of author/subject references for research, recipe card files, etc. Some of the individual subroutines (in particular, the sorting routine and the stopwatch routine) should also be useful in many other applications. The program, as presented here, is contained within the standard 16K TI BASIC. A version in Extended BASIC to access the additional 32K RAM should give the program an even broader scope.

References

1. Blinchikofff H. J. and Zverev A. I., *Filtering in the Time and Frequency Domain*, John Wiley and Sons, 1976, Chapter 4.
2. Floyd R. W., *Algorithm 245, TREESORT 3*, Communications of the A. C. M., Volume 7, December 1964, pp 701.
3. Luff P. P., *The Electronic Telephone*, Scientific American, Volume 238, Number 3, March 1978, pp 58-64.
4. Renbarger J., *A Telephone-Dialing Microcomputer*, BYTE, June 1980, pp 140-170.


```
100 REM   ***********************
110 REM   * THE HOME SECRETARY *
120 REM   ***********************
130 REM   99'ER VERSION 7.81.1
140 REM   BY MALLADI SUBBAIAH
150 REM
160 REM
170 DIM A1$(60),A2$(60),A3$(60),A4$(60),A5$(60)
180 DIM CAT$(6)
190 DIM P1(3),P2(3)
200 DATA 697,770,852,1210,1340,1481
210 READ P1(1),P1(2),P1(3),P2(1),P2(2),P2(3)
220 DATA NAME:,PHONE:,STREET:,CITY&ZIP:,MISC:
230 DATA "ITEM:      ",COST:,SHOP:,WHEN:,MISC:
240 CALL CLEAR
250 LSIZE=0
260 OPT=1
270 READ CAT$(1),CAT$(2),CAT$(3),CAT$(4),CAT$(5)
280 PRINT "LOAD DATA? (Y/N)"
290 GOSUB 3120
300 IF KEY<>89 THEN 330
310 GOSUB 1900
320 GOTO 410
330 REM  NEW SET UP
340 PRINT "PHONE BOOK? (Y/N)"
350 GOSUB 3120
360 IF KEY=89 THEN 390
370 OPT=2
380 READ CAT$(1),CAT$(2),CAT$(3),CAT$(4),CAT$(5)
390 N=0
400 GOSUB 430
410 GOSUB 850
420 GOTO 410
430 REM  KEY INPUT FOR DATA SET UP
440 PRINT :"  ENTER":"          E TO
    EXIT":"          R TO REENTER"
450 FLAG1=1
460 FLAG2=1
470 I=N+1
480 IF I<=60 THEN 520
490 PRINT ::"  *** ARRAY FULL  (N=60)***"::
500 GOSUB 3100
510 RETURN
520 PRINT
530 INPUT CAT$(1):A1$(I)
540 IF A1$(I)="E"THEN 750
550 IF A1$(I)="END"THEN 750
560 IF A1$(I)=""THEN 530
570 IF A1$(I)<>"R"THEN 620
580 I=I-1
590 N=I-1
600 PRINT :"  *** REENTER LAST SET ***"
610 GOTO 520
620 IF OPT<>1 THEN 650
630 INPUT CAT$(2):A2$(I)
640 GOTO 670
650 INPUT CAT$(2):T
660 A2$(I)=STR$(T)
670 INPUT CAT$(3):A3$(I)
680 INPUT CAT$(4):A4$(I)
690 INPUT CAT$(5):A5$(I)
700 GOSUB 770
710 IF T>190 THEN 600
720 GOSUB 800
730 N=I
740 GOTO 470
750 GOSUB 2410
760 RETURN
770 REM  MEMORY CHECK
780 T=LEN(A1$(I)&A2$(I)&A3$(I)&A4$(I)&A5$(I))
790 RETURN
800 LSIZE=LSIZE+T
810 IF LSIZE<3300 THEN 840
```

Continued on p. 70

at the CONSUMER ELECTRONICS SHOW

By Gary M. Kaplan

THE STAR OF THE SHOW: Texas Instruments' new console — the TI-99/4A with typewriter-like keyboard and new low price. Other show highlights included demonstrations of the TI LOGO language, TEXNET database access with text-to-speech, and a VCR controller (shown above, to left of disk drive).

at the CONSUMER

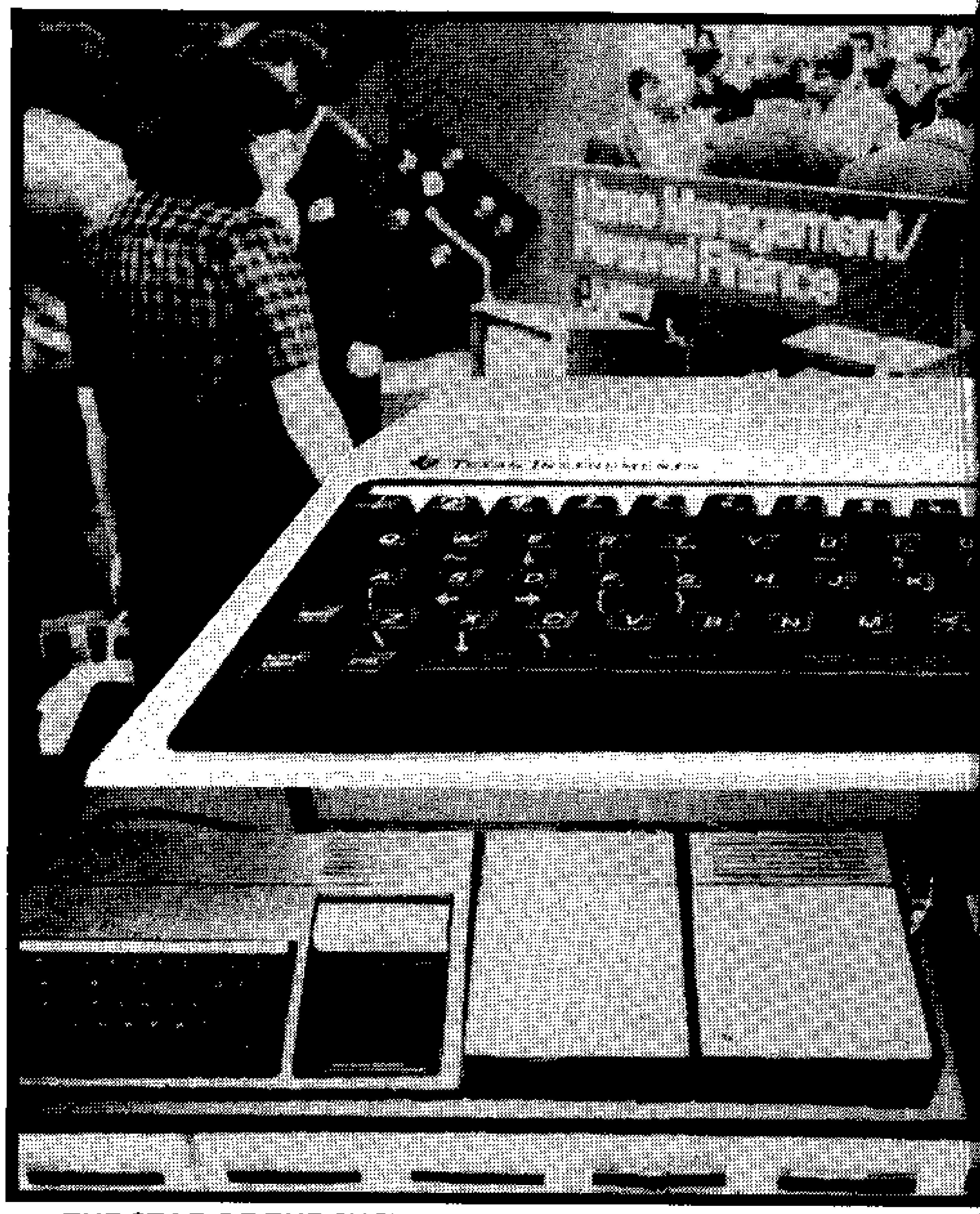
I went to Chicago not knowing quite what I'd find . . . Speculation in the microcomputer community had been running rampant: Would Texas Instruments introduce a new personal computer at the summer Consumer Electronics Show? What would it be like? What would it cost?

During the month prior to the May 31st-June 3rd event, rumors rippled throughout the trade press, computer dealer network, and users-group grapevine. According to a host of "reliable sources," TI would both release and not release a new machine at this time. If one believed the *former* group of reliable sources, the new machine would bridge both the *lower gap* between TI's programmable calculators and the TI-99/4, as well as the *upper gap* between TI-99/4 and the 990-series of business computers. That seemed like quite a bridge to me, although *where* the bridge was actually located depended on *which* group of reliable sources one believed was more reliable . . . The cost of this phantom machine would, I was reliably informed, be both significantly less and significantly more than the price of a TI-99/4. I guessed that was because, according to more reliable sources, it would be "bare-boned" as well as "fully loaded." Quite versatile this existing/non-existent machine!

The stage was set for its—if it existed—Chicago debut. Well, it does indeed exist: Enter the TI-99/4A personal computer—a new console that retains the same profile, speech capability, color graphics and software compatibility of its predecessor, but replaces the old 40-key calculator-style keyboard with a standard-configuration, 48-key typewriter keyboard. This, in itself, would be quite a welcome improvement; but take a closer look at the new keyboard's specifications and you'll soon discover that it really is a *new* keyboard and *not* just the recipient of some superficial cosmetic changes. Witness its impressive array of features: such niceties as both upper and lowercase; an ALPHA LOCK key (for conveniently keeping the keyboard in uppercase mode); auto repeat (with a two second time delay) on all keys; function (FCT'N) and control (CRTL) keys (for secondary functions and telecommunications use); and finally, a BASIC that will accept both upper and lowercase characters but displays all reserved words, variable names, and subprogram names in capitals for easy identification. How's that for versatility?

The keyboard action has an excellent "feel." Although it is still the same small size as its predecessor, touch typing on this new keyboard gives you the illusion of typing on a big office machine. It seemed quite capable of keeping up with my occasional blinding bursts of speed—something the old keyboard (and quite a few other microcomputer keyboards as well) is incapable of doing. Incidentally, I have been informed by Texas Instruments that since the circuit board has been re-designed, there's no practical way of retrofitting existing TI-99/4s with the new keyboard.

In addition to the features I've mentioned above, there have been some other changes. As you can see from the photograph, don't expect the familiar keyboard overlays to fit this new machine. Instead, it uses a two-level strip overlay mounted above the top row of keys (similar to super-



THE STAR OF THE SHOW: Texas Instruments' new console — the TI-99 demonstrations of the TI LOGO language, TEXNET database access with

market shelf pricing strips and holders). Also, due to some ROM/GR swaps on the new circuit board, the console now has a built-in "tr circuit (that has to be installed in the old console through a *separ* factory modification) for TMS9900 assembly language debugging the UCSD Pascal Development System (see *99'er Magazine*, May/J 1981, p. 56); in the process, however, the console lost its Equation C culator mode.

Even with all these changes described so far, I've yet to mention most exciting feature of the new console: its remarkably low price suggested retail price of $525 sure buys a lot of features in this packa The bottom line is that TI's new machine is *less than half* the price anything from another manufacturer that even comes close to its pow and versatility. (There presently isn't any other micro on the mar

ELECTRONICS SHOW

By Gary M. Kaplan

ith typewriter-like keyboard and new low price. Other show highlights included o-speech, and a VCR controller (shown above, to left of disk drive).

that offers the capability of combining color graphics, sound, speech, sprites, and music in the *same* program!) I strongly suspect that very shortly we will be welcoming a large number of new users into our 99'er community . . .

TI LOGO & TEXNET[SM]

The new console wasn't the only TI product to attract crowds of show-goers. TI LOGO, a new programming language, and TEXNET, an on-line computer information and communication service for the TI Home Computer [see related articles in this and the previous issue] captured the interest and imagination of all who passed the exhibit and took the time to look at the colorful sprite animation on the video monitors and listen to the latest UPI presswire news "voiced" through the speech synthesizer. A prototype of TI's *Terminal Emulator II* provided the text-to-speech translation of the UPI database material that was being sent over the telephone from The Source[SM] in McLean, Virginia, and also provided access to the TEXNET portion (a subset of The Source) for viewing pictures from the *TI Graphics Library* and hearing scores from the *TI Music & Sound Library*. Subscribers to TEXNET will be able to download any items in these libraries for use in their own programs.

TI Video Controller

Besides the numerous TI-99/4A consoles that were hard at work demonstrating new software at various locations throughout the exhibit area, Texas Instruments also had another piece of new hardware earning its keep—although, this one, the *TI Video Controller*, was *not* being presented as a consumer item. Rather, it was part of an interactive point-of-sales display in which a videotaped Bill Cosby joked with the passers-by as he asked and answered questions about the TI product line according to typed in keyboard responses. TI will be positioning the Video Controller system for sale in the institutional market—it's a particularly attractive package for schools and developers of corporate training programs.

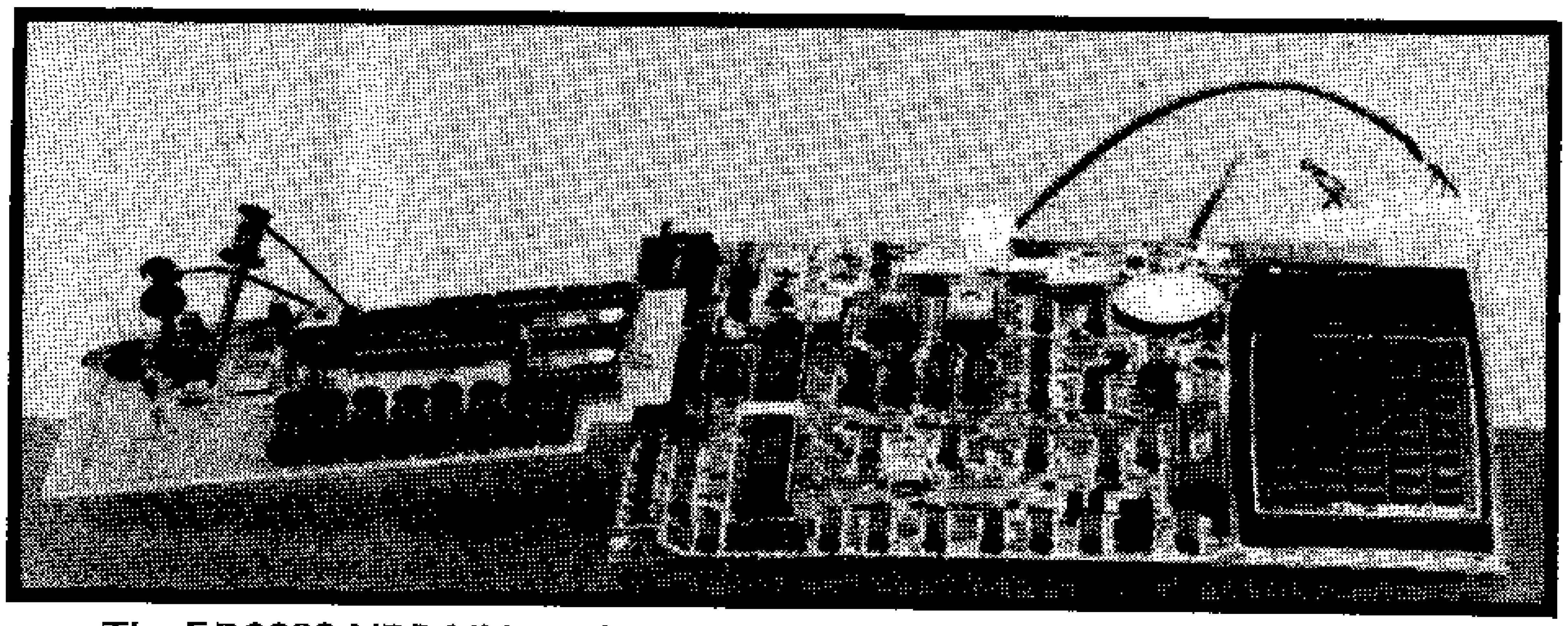
The Video Controller (packaged in the familiar silver and black peripheral box of the TI-99/4 product line) is designed to work with industrial-grade video recorders having the ability to accept a remote control unit; it may be used with both ½-inch and ¾-inch video cassette recorders, as well as a video disk player. With it, the TI computer can automatically direct a tape or disk to predetermined segments of the video program (under Extended BASIC control). The result is a video system for developing and delivering customized teaching or training programs—a perfect marriage between the interactive and immediate feedback capabilities of the computer and the realism of video.

With the Video Controller (about $700) and the *Authoring Software* package (about $200) hooked into a TI-99/4A disk system with Extended BASIC, an author can design or update courses using existing videotapes (without altering the original tape), or design new video courses that take maximum advantage of the computer's branching capabilities. The *Author* program is designed to *eliminate* the need for computer programming in developing a course, by guiding the development with step-by-step instructions on the display. Adding the text-to-speech capabilities of the Terminal Emulator II and Solid State Speech Synthesizer, will allow a developer to utilize virtually unlimited vocabulary to instruct a student taking the course. And since it's possible to store the resulting interactive AV course on the second audio track of the videotape, there is no need for

Continued on p. 56

BOMBS AWAY! ! ! !

(UNIVERSITY BASIC Style)



The ER3389 VDP Video Display Processor Board shown connected to the Texas Instruments TM990/189 University Module.

By Paul R. Roper & Richard T. Adams

Eyring Research Institute, Inc.
1455 West 820 North
Provo, Utah 84601

BOMBS AWAY is an interactive game for the Texas Instrument's TM990/189 University Board with a TMS9918 Color Video Display Processor (VDP) board attached. The game is written in University BASIC and takes advantage of all the color graphic commands included in the language. This is especially true of the MOVE command which moves and reflects SPRITE images at different speeds, whether or not a BASIC program is running.

Interactive inputs from either a joystick or the user keyboard control the altitude and speed of the airplane. Bombs are dropped from the plane on three moving targets: a fighter plane, a boat, and a submarine. When bombs are not falling, the fighter plane is trying to shoot down the bomber plane. Also, the boat is bobbing up and down on the whitecapped water, and the submarine is diving and surfacing continually.

Points are scored for each hit depending on the plane's altitude and the target hit. Points are deducted for each bomb dropped and each hit inflicted by the fighter plane. A running high score is maintained by the program and displayed at the end of the game.

MOVE Command

The key to the simulation is the MOVE command of University BASIC. This versatile command allows sprite movement to be done by the University BASIC system clock without program control. The MOVE command generates specific BASIC variables for monitoring and altering the sprite positions in real time. The X and Y delta velocities and automatic reflection limits can also be set for each sprite image.

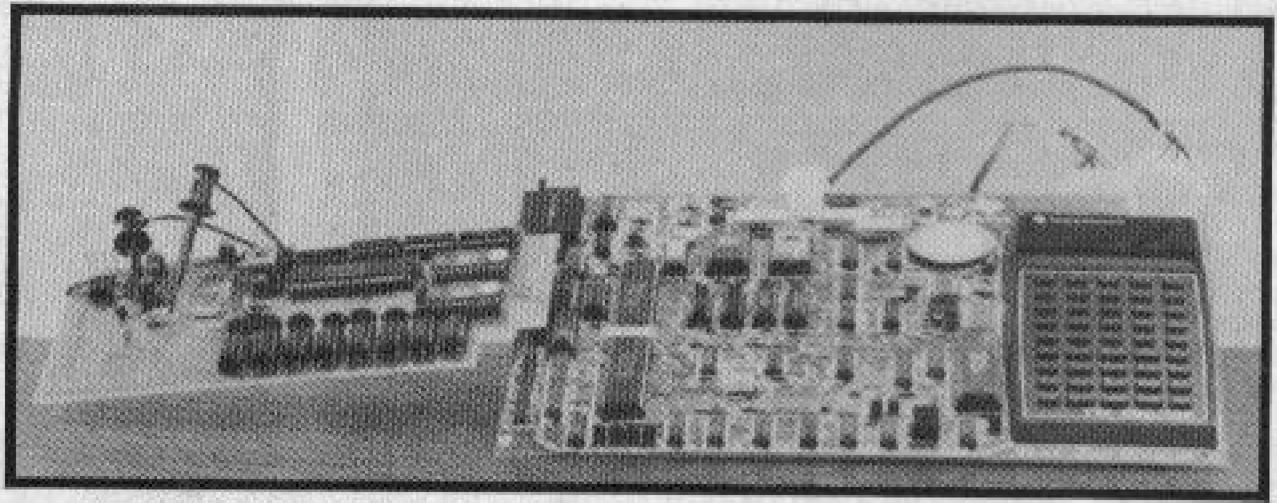
In this program, five sprite images are set in motion by the MOVE command. (Up to 32 sprites can be simultaneously set in motion.) The main loop of the program monitors the sprite positions, initiates new sprites, and watches for sprite coincidence. If the BASIC program had to move each sprite individually, the smooth and fast responses would be impossible.

If the MOVE command has only one argument, a table is generated to hold sprite positions, increments, and limits. Corresponding variables are also added to the symbol table of the BASIC interpreter. The variables begin with the letters X and Y followed by the sprite number. For example, MOVE 5 creates ten BASIC variables: X1, Y1, X2, Y2, X3, Y3, X4, Y4, X5, and Y5. These are continuously updated by University BASIC to the current position of the corresponding sprite—even while BASIC is in keyboard mode. The variables can be read or changed at any time by a BASIC program, and the system will use the new values. *Bombs Away* uses these variables to check when to change the sprite patterns as the planes fly back and forth. They are also used to determine which sprites caused the coincidence flag (variable CF) to be set.

Sprites are set in motion by the MOVE command when the sprite number is followed by the delta X and Y velocities. These velocities are the amount added to a sprite sum or position by each clock increment. The sprite is moved according to the sum divided by 256. Hence, if the delta movement were 2, then 128 clocks would be required to move the sprite 1 pixel or screen position. For example, MOVE 1, 256,512 would move sprite #1 one pixel in the X axis and two in the Y axis for every clock increment. As was stated, the clock runs independent of any BASIC program. The boat and submarine are given both X and Y velocities while the planes have only X movements.

If the sprite is already moving and a new positive delta velocity is given, then the sprite will continue to move in the same direction with the new delta velocity. If the new delta velocity is negative, then the sprite will begin to move in the opposite direction with the new absolute delta velocity, reflecting the sprite. Hence, in order to direct a moving sprite in a specific direction, the sprite must first be stopped, and then given the signed delta velocity. For example, MOVE 1,0,DY;1,–DX,DYwill set sprite #1 in a right to left motion from the current position and leave Y unaffected.

The reflection limits are specified with a negative first argument followed by the X and Y limits. These limits are generated by multiplying the lower limit by 256 and adding the higher limit. The lower limit should be less than the upper limit. The X limit ranges from 0 to 255 while the Y limit ranges from 0 to 192. (Since University BASIC is an integer BASIC and numbers range from –32768 to 32767, these limits may be negative.) For example, MOVE –3,230, –30580 will reflect on the X axis between 0 and 230 (0*256+230=230). The Y axis reflects between 136 and 140 (136*256+140=34956). However, this number is greater than 32768 and hence 65536 must be subtracted from it



The ER3389 VDP Video Display Processor Board shown connected to the Texas Instruments TM990/189 University Module.

giving −30580. The effect of the boat bobbing up and down on the water and the submarine diving and surfacing is achieved by setting small Y reflection limits.

Hardware

The game is played interactively with the keyboard, or it can be made more enjoyable by attaching joysticks having a "fire" button. Line 5 of the program selects a joystick. It must be deleted when only the keyboard is to be used.

The color VDP board is manufactured by Eyring Research Institute Inc., Provo, Utah. It attaches directly to address bus connector P4 of the University Board and has composite video output. A joystick interface is also provided on the board. The JOY function makes an assembly language call to hex address >1002. A listing of the routine is given in Figure 2.

The Game

Now, a little about the game. You have 25 bombs and start with 500 points. Ten points are deducted for each bomb dropped and fifty points for each time you're hit by the fighter plane. A submarine hit by a high altitude scores the maximum number of points.

Sprite patterns are defined as follows:

- 0 = Plane moving right
- 1 = Plane moving left
- 2 = Boat
- 3 = Submarine
- 4 = Bomb
- 5 = Explosion
- 6 = Fighter shell

Two fixed patterns are also used to set up the water and the whitecaps. These are defined as follows:

- 7 = Whitecaps on water
- 8 = Water

The actual sprites are assigned patterns and colors according to their position on the screen. The sprite object and color assignments are:

1 = Plane	White
2 = Fighter plane	Light Green
3 = Boat	Dark Blue
4 = Submarine	Red
5 = Bomb or Fighter Shells	Green

If only the keyboard is used, then line 5 of the program should be deleted. (Type 5 followed by a carriage return.) Then the following keyboard characters control the speed and altitude of the plane:

- I = Drop bomb
- J = Slow plane's velocity
- H = Increase plane's velocity
- N = Increase plane's altitude
- D = Lower plane's altitude

Figure 1. Program Listing

```
5 SP=1
10 MODE -1;-2;1,-1: N=500: NB=25: PS=100: GOSUB 200
12 SPRITE 0,"0010838FFE031020000811FD7F100804001 3183CDB7E3C"
14 SPRITE 3,"1808083C7EFF7E3C0028103838281 0007E31BDA5A5BD817E30"
16 PATTERN 7,"00000000001028C7": COLOR -240: VDP(960)=240: VDP(961)=5
20 FOR I=576 TO 767
22 VDP(I)=7+I/608
24 NEXT I
30 MOVE 5;-1,230,0;-2,230,0;-3,230,-30580;-4,230,-25666;-5,230,191
40 SPRITE -2,0,2;-3,2,4;-4,3,6
42 MOVE 2,130,0;3,80,30;4,120,50: Y2=110: GOSUB 250
44 PRINT @99;NB" "@119;N" ": IF CF: GOTO 44
46 IF NB<0: GOTO 300
50 IF X1<10: BD=1
52 IF X1>220: BD=-1
54 SPRITE -1,(1-BD)/2,15: IF X2<10: SPRITE -2,0
56 IF X2>220: SPRITE -2,1
60 GOSUB 200
62 IF A<5: A=5
64 IF A>115: A=115
66 Y1=A: MOVE 1,0,0;1,PS*BD,0: IF B: GOTO 90
68 IF Y5<Y1: X5=X2+10: Y5=Y2-4: SPRITE -5,6,15: MOVE 5,0,0;5,RND 500-250,-500
70 IF CF: P=1: N=N-50: GOTO 100
72 IF J2=0: GOTO 50
80 B=50: BX=PS*BD: X5=X1: Y5=Y1+16: MOVE 5,0,0;5,BX,B
82 Y=Y5: SPRITE -5,4,2: N=N-10: NB=NB-1: GOTO 44
90 IF Y>Y5: GOTO 42
92 IF Y5>130: B=100: IF J2: GOTO 80
94 Y=Y5: B=B+75: BX=BX-BX/5: MOVE 5,ABS BX,B: IF CF=0: GOTO 50
96 P=1: IF Y5>95: P=Y5/32-1
98 N=N+P*30-A/10
100 MOVE P,0,0: GOSUB 250
102 FOR I=1 TO 16
104 FOR J=1 TO RND 4
106 COLOR I: TONE J,I
108 NEXT J
110 SPRITE -P,5,I
112 NEXT I
114 GOTO 40
200 IF SP: PS=JOY 0: A=JOY 1/4-20: J2=JOY 2/200: RETURN
202 J2=0: I=KEY 0: IF I=73: J2=1
204 IF I=74: PS=PS-25
206 IF I=72: PS=PS+50
208 IF I=78: A=A-5
210 IF I=68: A=A+5
212 RETURN
250 B=0: MOVE 5,0,0: SPRITE 5,99,240;-5,4: Y5=0: RETURN
300 IF N>HI: HI=N
302 PRINT @99"GAME OVER"@298"HI ="HI: IF KEY 0: GOTO 10
304 GOTO 302
```

Program Description

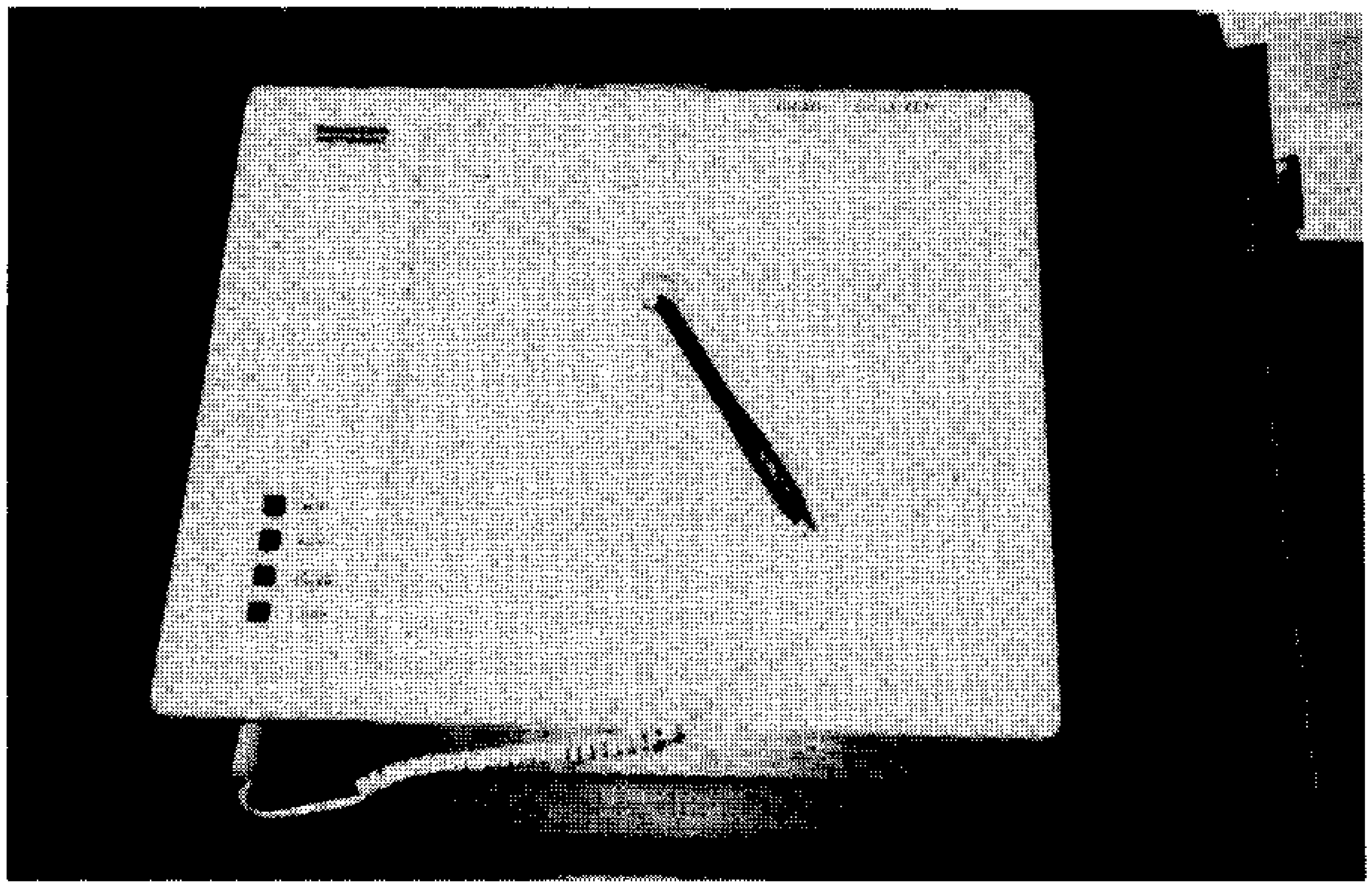
Line Nos.	
5	Keyboard or Joystick variable.
10-24	SPRITEs, PATTERNs, and screen initialization.
30	SPRITE MOVE limits.
40	SPRITE assignments.
42	SPRITE velocities and coordinates.
44	Print score & wait for non-coincidence.
46	End of game check.
300-304	End of game.
50-66	Set plane position & sprite assignment.
68	Fighter plane gun fire.
70	Plane hit by bomb.
72-82	Bomb drop.
90-98	Accelerate bomb and check for hit.
100-114	Explosion!
200-212	Plane speed and altitude inputs.
250	Disable bomb.

Note for Figure 2.
[This software routine must either be burned into EPROM and plugged into one of the vacant sockets on the Eyring board, or loaded into external, off-board RAM. Watch next issue for a discussion of adding this additional RAM to the University Board—Ed.]

Figure 2. Joystick Software

```
        0000'1000                  AORG >1000
 1000:  1000                       NOP
 1002:  1004                       JMP JOYFF
U1004:  0000                       DATA PRGM0      ;CONTROL
U1006:  0000                       DATA PRGM1      ;SLOT
U1008:  0000                       DATA PRGM2      ;BREAKOUT
U100A:  0000                       DATA PRGM3      ;BOMBS
                           *
                           ****************************************
                           *       JOY FUNCTION
                           *
 100C:  04C0               JOYFF   CLR R0          ;CLEAR RESULT
 100E:  D061 103C                  MOVB @JOYTB(1),R1
 1012:  110E                         JLT JOYF6     ;SWITCH
 1014:  0300 0000                  LIMI 0          ;DISABLE CLOCK
 1018:  04E0 1801                  CLR @JOYW       ;START CONVERSION
 101C:  6020 003A                  S @JOYB,R0      ;SUBTRACT BIAS
 1020:  0202 2305                  LI R2,JOYR      ;GET JOYSTICK ADDRESS
                           *
 1024:  0580               JOYF2   INC R0          ;COUNT LENGTH OF 1 SHOT
 1026:  5052                       SZCB *R2,R1     ;WAIT TO GO 1
 1028:  16FD                         JNE JOYF2
                           *
 102A:  0300 0001          JOYF4   LIMI IMSK       ;ENABLE INTERRUPTS AGAIN
 102E:  045B                       RT              ;RETURN
                           *
 1030:  D020 0082          JOYF6   MOVB @JOYD,R0   ;GET SWITCH READ
 1034:  5001                       SZCB R1,R0      ;MASK
 1036:  7800 0082                  SB R0,@JOYD     ;REMOVE
 103A:  045B                       RT
                           *
 103C:  0102 FB00          JOYTB   BYTE >01,>02,>FB
 103F:  0810 DF00                  BYTE >08,>10,>DF
```

HI PAD™ DIGITIZER



By W.K. Balthrop & G.R. Michaels

"Simplify your life . . ." An extremely pleasant thought, isn't it? At least that's what one of America's large insurance companies is betting on. Their multi-million dollar TV ad campaign boldly asserts that its policy coverage will, in fact, simplify your life,—make the financial and protection aspects of life seem easier, less complicated . . .

These series of ads started us thinking: A personal computer is *supposed to* simplify our lives. But does it really? Many times, the new computerized methods that replace the old manual methods are still cumbersome; we've somehow traded in slow and tedius methods—but usable, nevertheless, because they're exactly the way *we* feel comfortable doing things—for faster, more organized, but noticeably awkward mechanized methods. Most of the time this is because of the constraints imposed on the software through its dependence upon an overly complicated "man-machine" interaction via a keyboard. In a nutshell, we are forced to conform to an unnatural way of entering data.

Think how much easier our "life with computers" would be if we could simplify this interaction . . . Readers of *99'er Magazine* are already familiar with the giant strides taken by Texas Instruments in the area of information *output*—specifically voice synthesis and text-to-speech. An efficient, moderately priced system of voice recognition for data *input*, however, still seems somewhat far away at present.

Even though we have to forego voice input at present, there are still other methods for communicating with a computer that are more natural than through a keyboard. Take for example, the data tablet/digitizer—an input device that converts graphic, tabular, or menu-type information into digital values usable by a computer. By merely touching a pen-like stylus to any position on a map, diagram, chart, menu, or other graphic presentation resting on the tablet's surface, the coordinates of that position are transformed into their digital equivalents, and are sent back to your computer where the appropriate software converts the data into meaningful information input. For more accurate positioning on the tablet's surface, digitizers use a cursor with a cross-hair sight and "fire" button.

What can you use a digitizer for? Applications are as unlimited as your imagination. For starters, how about order entry, opinion sampling, or menu selection? Checking an appropriate box on a pre-printed form inputs the data quickly and with less chance for errors.

In the field of education—especially computer-assisted instruction—a digitizer can eliminate the tedium of typing and enhance the excitement of learning through more natural interaction.

In design and drafting work a digitizer is a natural: Structural elements, floor plans, piping diagrams, printed circuit board layouts, mechanical parts—virtually every type of graphic information you can trace—can be more efficiently input with this device.

Typical uses in business and industry include forecasting and planning applications, trend and comparative financial analysis, sales performance, inventory control, and stock/commodity charting.

The medical field, too, can benefit from simplified data entry with a digitizer. Whether in lab or clinic you'll find plenty of possible uses—applications such as x-ray and ultrasound planning and analysis, and lab report data entry.

And by all means, let's not forget games and computer graphics. Since a digitizer is a more precise alternative to joysticks and light pens, think of the possibilities for fast, interractive simulations. The list of possible applications goes on and on . . .

The potential uses for a digitizer are indeed impressive, but we were wondering how easy it would be to interface one to a personal computer such as the TI-99/4, and make it "do its tricks." For our testing and software development we selected the Houston Instrument HI PAD—one of the popular digitizers designed for personal computers. In a series of articles we will report our progress in integrating this peripheral into our 9900-based family.

The HI PAD comes completely assembled with its own power supply and with built-in RS-232C and parallel interfaces. It has a digitizing area of 11x11 inches, a resolution of 0.005 inches or .01 inches (with scaling in English or metric units), and has an accuracy of ± 0.015 inches (at high resolution) in relationship to a user-defined origin. The coordinate system is absolute Cartesian with a choice of relocatable or fixed origin. In addition to RESET, there are three operating modes: (1) POINT, (2) SWITCH STREAM, and (3) STREAM.

Three user formats are available: (1) parallel binary, (2) parallel BCD (binary coded decimal), and (3) serial ASCII (with selectable BAUD rates of 300, 1200, 2400, and 4800). The digitizing rate, defined as the number of coordinate pairs per second (with a maximum of 100 in the HI PAD), is controlled and/or limited by the selection of out-

Figure 1.

Cable Connection Diagram for 4800 BAUD in POINT Mode

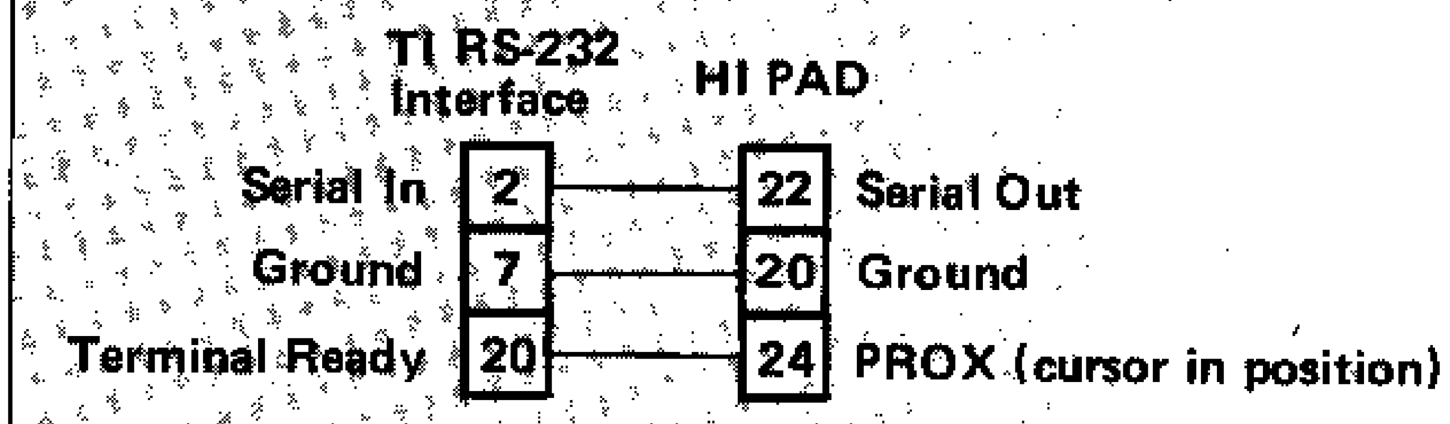


Figure 2.

C_B	±	X	X	X	X	X	±	Y	Y	Y	Y	Y	C_R	L_F

Where C_B
- = **0, indicates the first coordinate of a switched stream.**
- = **1, indicates successive coordinates of a switched stream.**
- = **2, indicates a coordinate of point mode.**
- = **3, indicates a coordinate of stream mode with the cursor button released.**
- = **4, indicates a coordinate of stream mode with the cursor button depressed.**

Where ± = **an ASCII coded + or −.**
Where X or Y = **ASCII coded digits 0 - 9.**
Where C_R = **ASCII coded carriage return.**
Where L_F = **ASCII coded line feed.**

put formats. The serial format happens to be the slowest, but for reasons of simplicity in interfacing, we chose it for our initial tests. To try out the POINT mode, all that was necessary was to wire up a cable (not supplied) with a male DB-25 connector going to the Texas Instruments RS-232 Interface, and a female DB-25 going to the HI PAD as indicated in Figure 1. In Figure 2 you'll find the serial format for the 15 ASCII coded characters.

The Programs

HI PAD DECODER is a short TI Extended BASIC routine to display the 15-character strings of serial data on the monitor or TV screen. Line 170 OPENs the RS-232 on port 2 (for 4800 BAUD and 8 data bits) to receive the 15-character ASCII string. The −128 at the end of line 210 is necessary for proper translation of the 8-bit codes to 7-bit ASCII codes for correct alphanumeric screen display (see *99'er Magazine*, May/June 1981 issue, p. 22). This translation does, however, slow down both this program and the one that follows.

HI PAD DEMO, with the interface connections as described above, is an Extended BASIC program that will allow you to calibrate the digitizer, and then calculate distance and total enclosed area for the POINT mode of operation. In forthcoming articles we'll explore other modes and applications.

```
100 REM ******************
110 REM * HI PAD DECODER *
120 REM ******************
130 REM 99'ER VERSION 7.81.1XB
140 REM BY W.K. BALTHROP
150 REM
160 REM
170 OPEN #1:"RS232/2.BA=4800.DA=8",
    FIXED 15
180 INPUT #1:C1$
190 DISPLAY AT(5,3):"DIGITIZED
    CODE:"
200 FOR X=1 TO 15
210 DISPLAY AT(6,5+X)SIZE(1):
    CHR$(ASC(SEG$(C1$,X,1))-128)
220 NEXT X
230 GOTO 180
```

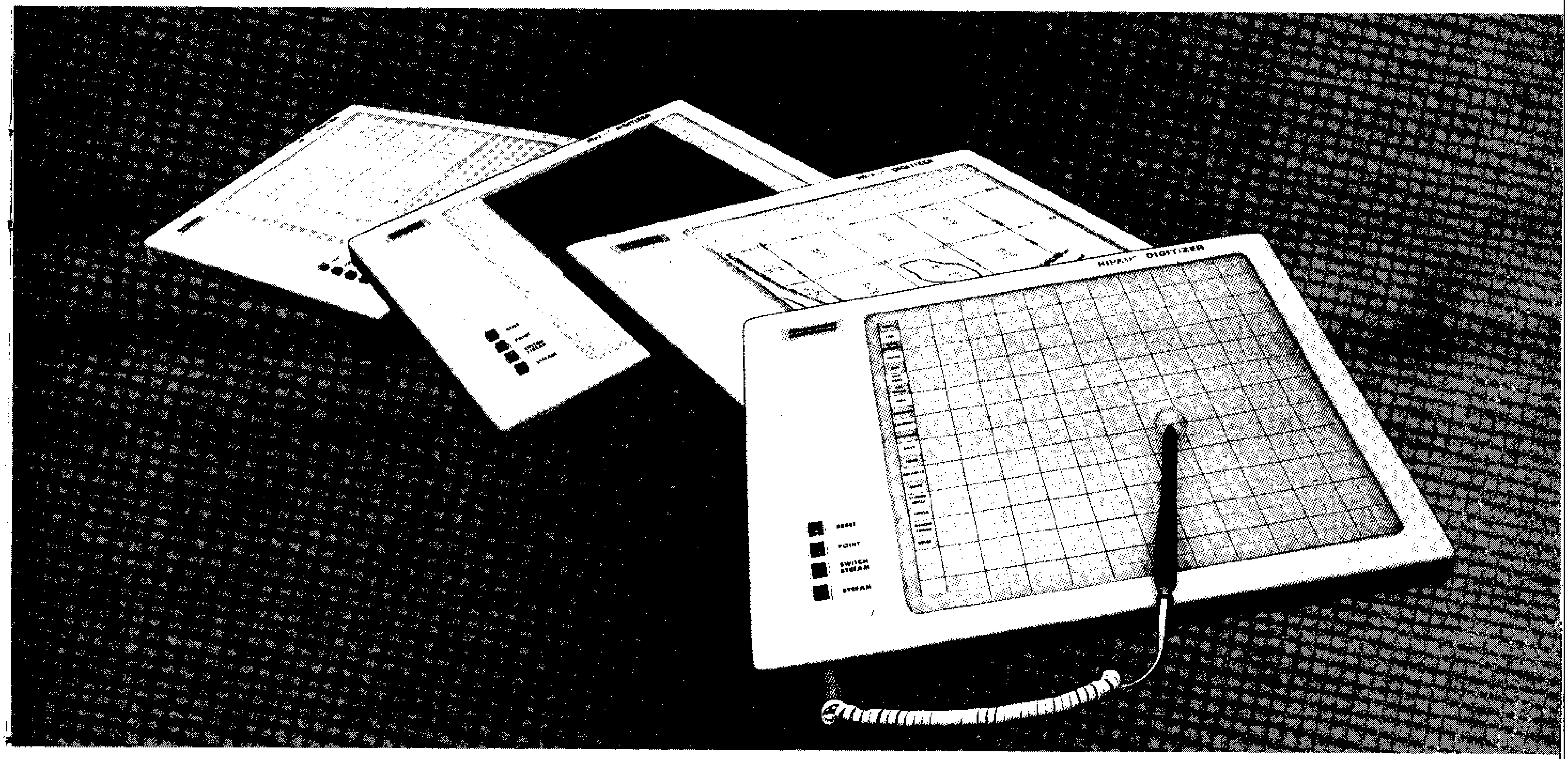
EXPLANATION OF THE PROGRAM
HI PAD DEMO

Line Nos.	
170	OPEN RS-232 port.
180	Set up variables.
190-250	Display option page & INPUT choice.
260-280	Branch to routine specified.
290-310	Instruction and control.
320-360	Instruction and control for digitizing end of line.
370	What type of linear unit will be used?
380	How many of the units are there in the calibration line you plotted?
390-400	Calculate units per inch; GOTO title page.
410-420	Instruction & control for digitizing beginning of line.
430-510	Print instructions to digitize the line. Update line length and print on screen.
520-590	Instruction and control to calculate total area.
600-640	Calculate total area and display.
650-670	Calculate distance between two points.
680-710	Define X, Y coordinates.
720-790	Input data from digitizer pad.
800-830	Assemble data from pad into proper ASCII characters. (subtract 128 ASCII).

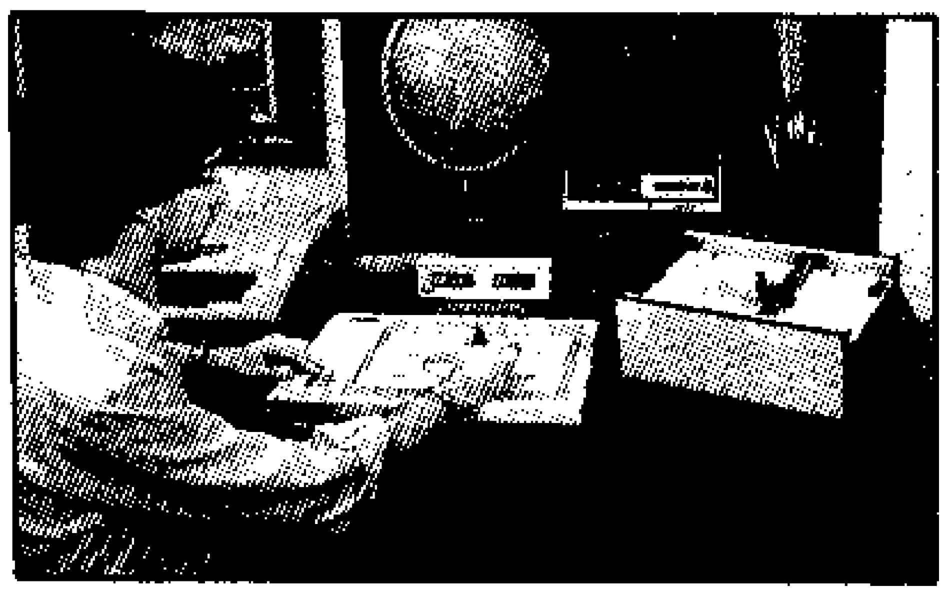
```
100 REM ***************
110 REM * HI PAD DEMO *
120 REM ***************
130 REM 99'ER VERSION 7.81.1XB
140 REM BY HOUSTON INSTRUMENT
150 REM
160 REM
170 OPEN #1:"RS232/2.TW.BA=4800.
    DA=8",FIXED 15
180 UPI=1 :: V$="INCHES"
190 CALL CLEAR
200 DISPLAY AT(3,3):"MENU: SELECT
    ONE OF THE          PROGRAMS."
210 DISPLAY AT(7,3):"1. CALIBRATE."
220 DISPLAY AT(9,3):"2. DISTANCE."
230 DISPLAY AT(11,3):"3. TOTAL
    AREA."
240 DISPLAY AT(24,6):"YOUR CHOICE?"
250 ACCEPT AT(24,18)BEEP:C :: IF
    C=0 OR C>3 THEN 250
260 CALL CLEAR
270 ON C GOTO 290,410,520
280 GOTO 250
290 DISPLAY AT(3,3):"PLACE
    DIGITIZER AT BEGINNING OF
    CALIBRATION LINE
    AND      DIGITIZE."
300 GOSUB 730
310 GOSUB 680 :: X2=X :: Y2=Y
320 DISPLAY AT(8,3):"NOW PLACE THE
    CURSOR AT THE END OF THE LINE
    AND DIGITIZE"
330 GOSUB 730
340 GOSUB 680
350 X1=X2 :: Y1=Y2 :: GOSUB 650
360 IPU=D
370 DISPLAY AT(12,3)BEEP:"WHAT TYPE
    OF UNIT IS THIS   CALIBRATION
    BASED UPON?" :: ACCEPT
    AT(15,3):V$
380 DISPLAY AT(17,3)BEEP:
    "HOW MANY ";V$;" DOES THE":"
    CALIBRATION LINE REPRESENT?" :
    : ACCEPT AT(20,3):A
390 UPI=A/D
400 GOTO 190
410 DISPLAY AT(3,3):"NOW DIGITIZE
    THE BEGINNING OF THE LINE."
420 GOSUB 730 :: GOSUB 680 :
    : X1=X :: Y1=Y
430 DISPLAY AT(5,3):"BEGIN USING
    THE CURSOR TO   DIGITIZE
    THE LINE."
440 DISPLAY AT(8,3):"TOTAL
    DISTANCE:"
450 DISPLAY AT(12,3):"WHEN
    FINISHED SELECT ""M"" TO
    RETURN TO MENU."
460 GOSUB 730 :: GOSUB 680 :
    : GOSUB 650
470 CALL KEY(0,K,S):
    : IF S=ASC("M")THEN 190
480 X1=X :: Y1=Y
490 TD=TD+D*UPI
500 DISPLAY AT(8,18):ABS(TD);V$
510 GOTO 460
520 TA=0 :: DISPLAY AT(3,3):
    "DIGITIZE ALONG OUTER LINE   OF
    AREA TO BE COMPUTED."
530 DISPLAY AT(8,3):"HIT THE ""A""
    KEY WHEN        FINISHED."
540 GOSUB 730 :: GOSUB 680 :
    : XP=X :: YP=Y :: YS=Y :: XS=X
550 GOSUB 730 :: GOSUB 680 :
    : X=X :: Y=Y
560 A=((YP+Y)/2)*(XS-XP)
570 TA=TA+A
580 YP=Y :: XP=X
590 GOTO 550
600 TA=TA+((YP-YS)/2)*(XS-XP)
610 DISPLAY AT(6,3):"TOTAL AREA:";
    ABS(TA*UPI^2);"SQUARE";V$
620 DISPLAY AT(20,3):"HIT THE ""M""
    KEY TO SELECT   THE MENU."
630 ACCEPT AT(22,12):I$ :: IF
    I$<>"M" THEN 630 ELSE CALL
    CLEAR :: GOTO 200
640 END
650 REM SUBROUTINE TO COMPUTE
    DISTANCE BETWEEN POINTS
    (X1,Y1)&(X,Y)
660 D=SQR(((X1-X)^2)+((Y1-Y)^2))
670 RETURN
680 REM SUBROUTINE TO DEFINE X,Y
    COORDINATES
690 X=VAL(SEG$(D$,3,5))/1000
700 Y=VAL(SEG$(D$,9,5))/1000
710 RETURN
720 REM SUBROUTINE TO ASSEMBLE
    STRING
730 D$=""
740 INPUT #1:C1$
750 CALL KEY(0,K,S)
760 IF K=ASC("M")THEN 190
770 IF K=ASC("A")THEN 600
780 GOSUB 800
790 RETURN
800 FOR Z=1 TO 15
810 D$=D$&CHR$(ASC(SEG$(C1$,Z,
    1))-128)
820 NEXT Z
830 RETURN
```

In this age of runaway inflation...

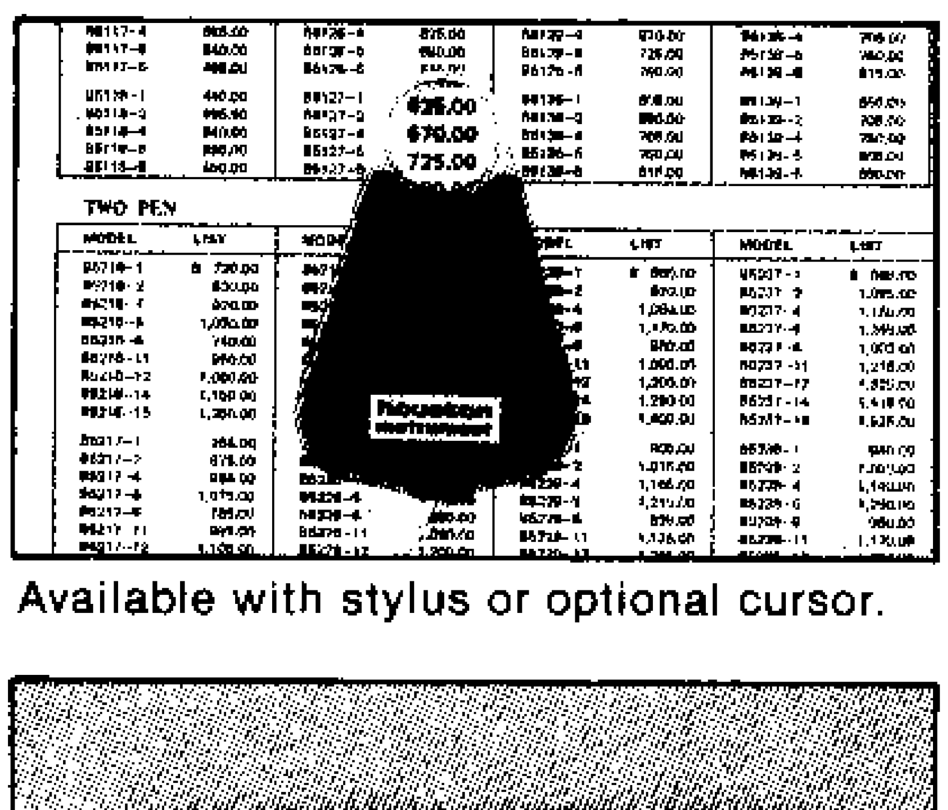
Look what $825* will buy



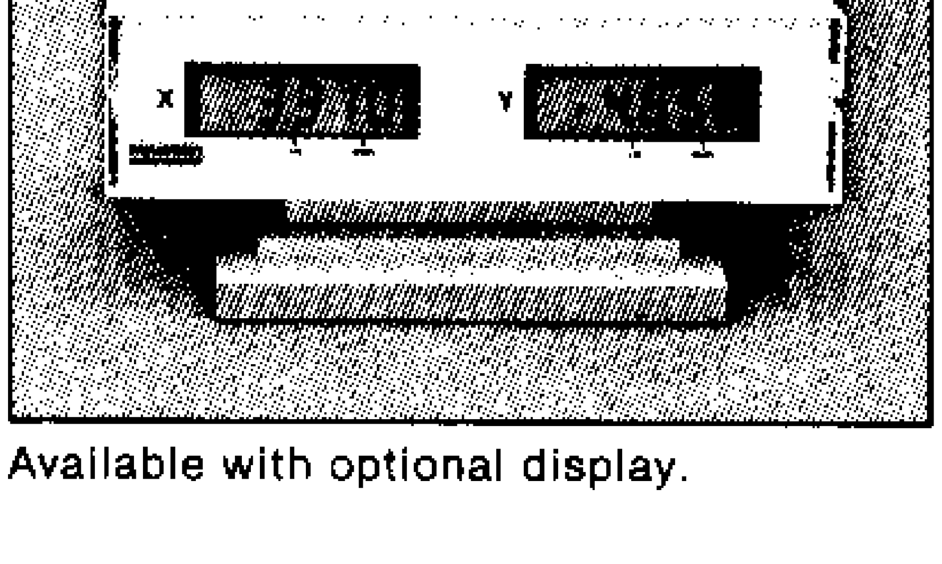
The HIPAD™ digitizer



The ideal input device for the small system user.



Available with stylus or optional cursor.



Available with optional display.

*U.S. Suggested retail price

Inexpensive input to your computer

The HIPAD™ digitizer can be used for both converting graphic information into digital values and as a menu. Utilizing either the stylus or the optional cursor, the operator can input graphic data into the computer by locating individual points on the digitizers 11" x 11" (28cm x 28cm) active area. In the "stream mode" a continuance of placements of coordinate pairs may be input.

Not a kit, the HIPAD™ comes complete with both RS-232-C and parallel interfaces and has its own built-in power source. The origin is completely relocatable so coordinates may be positive or minus for a true reference value and oversized material may by input by simply resetting the origin.

Accurate positional information, free form sketches, even keyboard simulation

All can be entered using the multi-faceted HIPAD™ digitizer. Its capabilities and low price make the UL listed HIPAD™ a natural selection over keyboard entry, inaccurate joysticks, or expensive approximating light pens. It's perfect for inputting isometric drawings, schematics, X-rays, architectural drawings, business graphs, and many other forms of graphic information, as well as creating your own graphics.

Use it with Apple II™, TRS-80 Level II™, PET™ or other popular computers

The HIPAD's™ built-in **RS-232-C** and parallel 8 bit interfaces make it all possible. (For Apple II order DT-11A, for TRS-80 or PET order DT-11). Furthermore, you get English or metric scaling, data format (Binary/BCD/ASCII), selectable baud rates, and resolution of either .005" or .01".

For complete information, contact Houston Instrument, One Houston Square, Austin, Texas 78753, (512) 837-2820. 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.*

TM HIPAD is a trademark of Houston Instrument
TRS-80 is a trademark of Tandy Corporation
APPLE is a trademark of Apple Computer Inc.
PET is a trademark of Commodore Business Machines, Inc.

houston instrument
GRAPHICS DIVISION OF
BAUSCH & LOMB

"WRITE YOUR OWN PROGRAMS"

Solution for Problem May/June Issue

```
100 REM   ****************
110 REM   *              *
120 REM   * NUMBER MATCH *
130 REM   *   EXPANDED   *
140 REM   *              *
150 REM   ****************
160 REM
170 REM  BY JAMES DUGAN
180 REM  99'ER VERSION 7.81.1
190 RANDOMIZE
200 GOSUB 620
210 FOR I=1 TO 25
220 LET A=INT(RND*10)
230 LET MSG$=MSG$&STR$(A)
240 CALL CLEAR
250 PRINT "HERE IS THE NUMBER"::
260 PRINT "YOU HAVE";D;"SECONDS TO STUDY IT"::::
270 FOR DELAY=1 TO 100
280 NEXT DELAY
290 PRINT TAB(5);MSG$:
300 GOSUB 530
310 CALL CLEAR
320 GOSUB 360
330 LET T=T+1
340 NEXT I
350 GOTO 720
360 PRINT "TYPE THE NUMBER"
370 INPUT RES$
380 IF RES$<>MSG$ THEN 450
390 CALL CLEAR
400 PRINT TAB(11);"GOT IT!"::::::::::
410 CALL SOUND(1000,262,1,330,1,392,1)
420 FOR DELAY=1 TO 400
430 NEXT DELAY
440 RETURN
450 PRINT "SORRY! THE NUMBER WAS: ";MSG$
460 CALL SOUND(1000,-3,1)
470 FOR DELAY=1 TO 500
480 NEXT DELAY
490 PRINT ::::
500 INPUT "DO YOU WANT TO CONTINUE?    ""Y"",
    OR ""N"".":ANS$
510 IF ANS$<>"Y"THEN 560
520 RETURN
530 FOR DELAY=1 TO D*333
540 NEXT DELAY
550 RETURN
560 PRINT :::::
570 PRINT "THE LONGEST NUMBER YOU"
580 PRINT "REPEATED CORRECTLY WAS ";T
590 PRINT "DIGITS LONG"::::
600 STOP
610 REM  **INITIAL SCREEN**
620 CALL CLEAR
630 PRINT TAB(9);"NUMBER MATCH"::::
640 PRINT TAB(5);"I CHALLENGE YOU TO REPEAT A"
650 PRINT TAB(5);"SERIES OF NUMBERS.":
660 PRINT TAB(5);"EACH TIME YOU GET IT RIGHT"
670 PRINT TAB(5);"I WILL ADD ANOTHER DIGIT"::
680 PRINT "HOW MANY SECONDS DO YOU"
690 PRINT "WANT TO STUDY THE NUMBER?"
700 INPUT "(.1-10)":D
710 RETURN
720 END
```

CORRECTION:

"How To Write Your Own Programs" May/June 1981, p. 11 [Ex. 3, Step 4, Coding]

Due to a typographical error, line 140 was incorrectly listed. The correct statement should have been:

```
140 If N=99 THEN 150 ELSE 110
```

or, reduced to an even simpler expression:

```
140 IF N<>99 THEN 110
```

Consumer Electronic Show . . . from p. 49

additional storage devices at each student station. [Watch for an in-depth product review in a forthcoming issue of *99'er Magazine.*]

Additional Software

TI's floor display was subtly "engineered" to drive home the point that its rapidly growing software arsenal is *already* quite substantial. All pathways led into a central "software monolith"—a huge display case covered with all of the existing Command Module packages. Of all the software that was actually available for "hands-on" testing, I was most impressed with the Command Modules produced for TI by Scott Foresman and Company—a leading publisher of school text books. *Addition & Subtraction I* is the perfect example of an interactive program that utilizes *all* the special features (and utilizes them *well*) of the TI machine—including speech. Between the unique creative programming environment of TI LOGO and the polished, animated, Sessame Street-style drill and practice software of Scott Foresman, it appears that the TI microcomputer is destined to assume a leadership position in the field of computer-based education. [Watch the September/October issue for a review of the new Scott Foresman software.]

The most significant piece of software at the show, however, was kept confined to a "back room." I say "most significant" because with it a serious programmer can pare away the layers of "friendliness" on TI's mild-mannered consumer computer to expose the heart of the extremely powerful 16-bit machine within. This stand-alone (*not* part of the UCSD Pascal Development System) TMS9900 assembler will be released on Command Module and will come with a thorough reference guide. TI obviously has a lot of experience in writing and documenting TMS9900 assembler packages, so we can expect this one to be a quality product. Incidentally, the assembler and full-featured text editor were largely lifted from TI's 990-series of business computers.

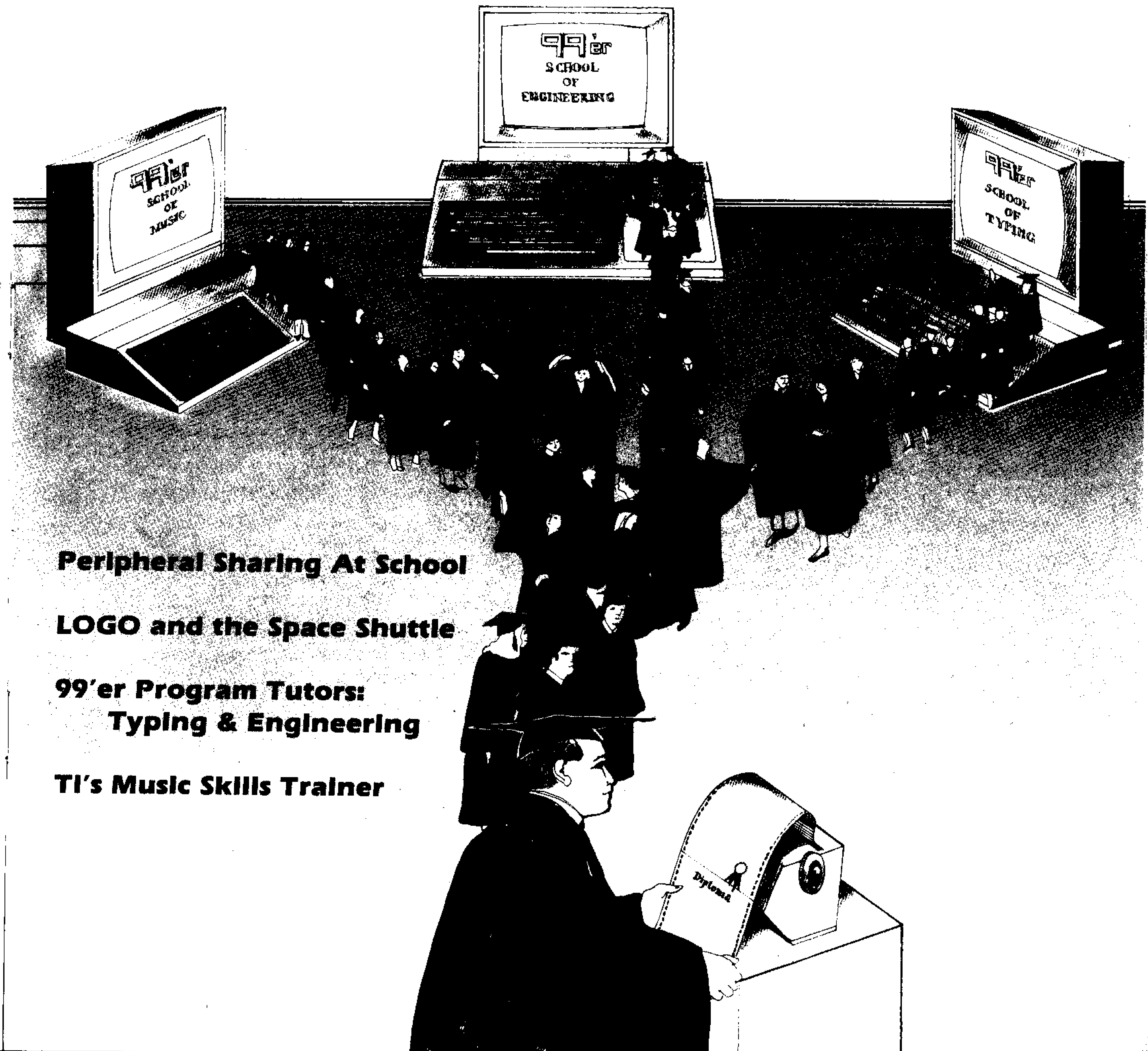
With typical Command Module simplicity and efficiency, at the touch of a number key you can (1) ASSEMBLE, (2) EDIT, (3) LOAD & RUN, or (4) RUN. The text editor provides you with a horizontally scrolling (in 20 column tabs) 40-column "window" into the full 80-column screen, and executes at "blinding speed"—significantly faster than the one written in pseudocode that comes as a part of the UCSD Pascal Development System (see *99'er Magazine*, May/June 1981, p. 55). Soon, programmers with a TI-99/4 or 99/4A disk system (one disk drive required) will be able to snap in this Command Module and be off and running. The prospects, therefore, look very good for the development of a host of serious, high-caliber new applications and simulations software for end users (LOADable into the 32K Expansion RAM and CALLable through Extended BASIC). Software of this type has not yet appeared because of the limitations of speed and high memory overhead imposed by a BASIC programming environment.

That was the summer CES as I saw it—all in all, an impressive performance from Texas Instruments. There was a world of difference between the TI at the winter CES, five months earlier in Las Vegas, and the TI exhibiting this summer in Chicago: The feeling this time—readily observable in the spirit and excitement of attending TI sales and management personnel—seemed to be one of "We've come a long way, Baby . . . " It makes one anxious to tap some more "reliable sources" to find out what TI is planning for the next CES this winter . . .

July/August 1981 Vol. 1, No. 2 $2.00 in USA

ON LoCAItion™

THE INTERNATIONAL JOURNAL OF COMPUTER ASSISTED INSTRUCTION.

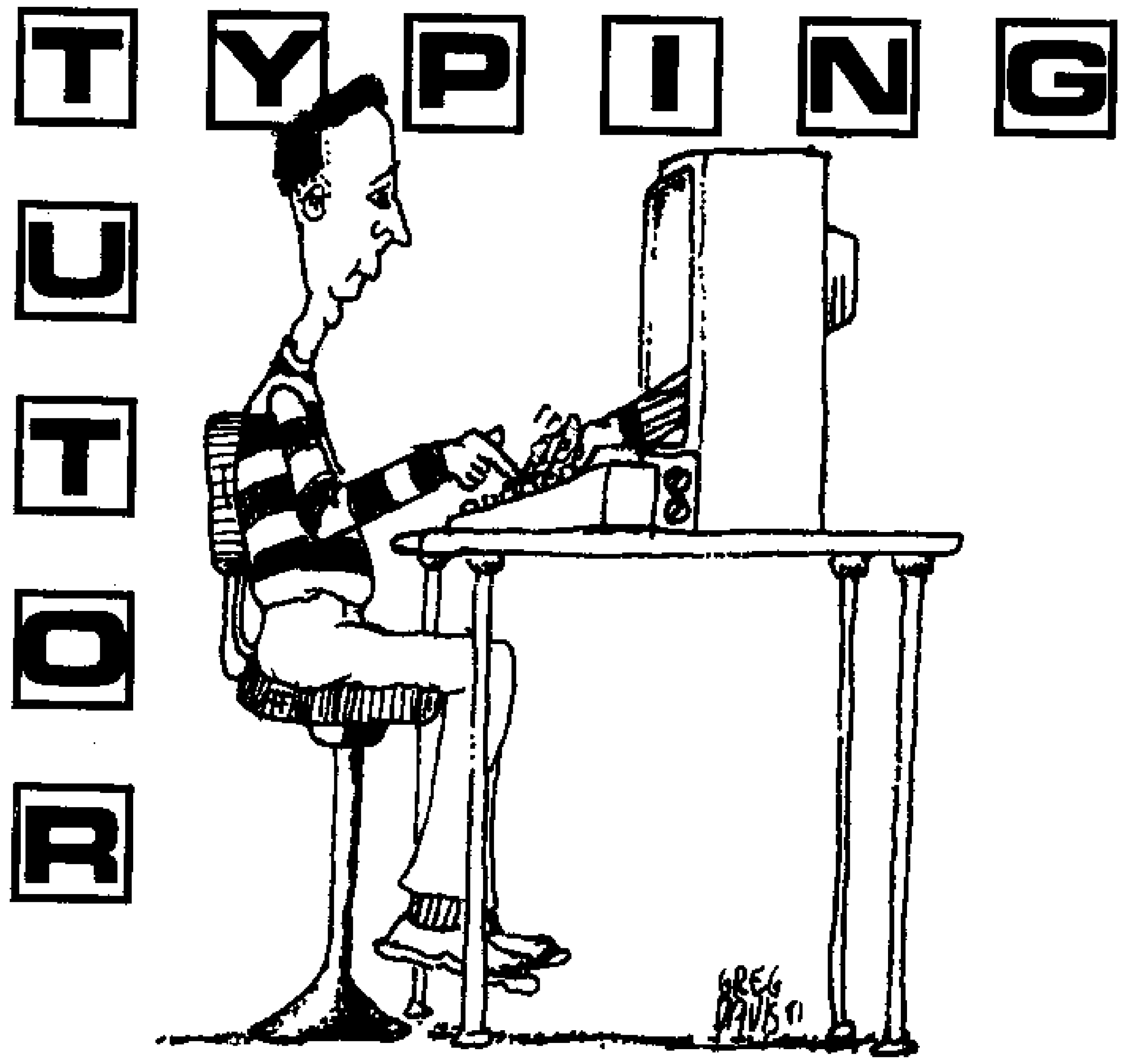


Peripheral Sharing At School

LOGO and the Space Shuttle

99'er Program Tutors: Typing & Engineering

TI's Music Skills Trainer



Part 1: Learning the Keyboard Symbols

By Regena

Welcome to QWERTY Academy—the school where everyone can either learn typing, or improve their existing typing skills. The Academy was named after the first six letters (left to right) on the second row of keys found on a standard typewriter. The instructors at QWERTY are probably already familiar to you. In fact, most of you are sharing your home with one right now. That's right—a Texas Instruments TI-99/4 personal computer, whose color graphics and sound capabilities make it one teacher you'll actually have *fun* learning from . . .

There are several differences between the TI-99/4 keyboard and a standard typewriter keyboard. Beginners won't have to worry about this until switching to a regular typewriter. At that point, the necessary adjustments are easy to make. Those of you who already "touch type" should learn the special fingering procedures needed for efficient typing on the TI keyboard. This will permit your letter writing to go faster, with fewer mistakes, and allow you to enter code more efficiently with fewer syntax errors when programming.

One main difference in the keyboards is that there is not a "home" key for the right little finger ("pinky") to rest upon. A "dummy" key is available (on a overlay from Texas Instruments) to help solve this problem. Another difference is the placement of the ENTER key: The TI-99/4 computer has the ENTER key where the typewriter has a SHIFT key (or where the period would be if you're using a dummy key), so a typist may tend to ENTER more than he or she wishes. And when, for example, a comma is needed, a typist must use a left-hand SHIFT to have it print correctly. Also, since there isn't a SHIFT key for the right hand, the symbols on the left side of the keyboard require pressing the SHIFT key with the left little finger, and pressing the symbol with either the left index or middle finger.

The instructional program that follows assumes that its users are already familiar with touch-typing techniques and just need to learn and practice the symbols on the TI-99/4 keyboard. A series of programs for teaching touch typing to beginners will commence in the next issue. In the meantime, beginners can practice with the symbols if they wish, or can save the program until after mastering the alphabetic keys. There will be versions (with only slight differences) for both the TI-99/4, and the new, soon-to-be-available TI-99/4A with its standard typewriter keyboard. This magazine will also provide programs for experienced touch typists who wish to improve their proficiency.

The Program

The program starts with the period (decimal), the only symbol that does not require the SHIFT key, and progresses through the other symbols. Suggested fingering for each of the symbols is presented. After each symbol or set of symbols is presented, there is a practice drill which includes the new symbol and previously learned symbols.

Each drill consists of actual phrases and statements that a programmer would use. You must type five phrases correctly to complete the drill and move on to the next lesson. A phrase is chosen randomly from nine phrases entered as data, and it is printed on the screen. After the prompter "beep," you must copy the phrase. If you type the phrase correctly, that phrase will not be used again; but if you type it incorrectly, it may be used again until a total of five phrases are correctly typed.

Because only a left-hand SHIFT key is available, the left little finger presses the SHIFT key and the middle finger presses the !, @, and # keys. You may prefer to use the index finger on %, $, and # and the third finger on ! and @, and also use the right index finger for ? rather than the left. After most of the symbols have been presented, six of the rarely used symbols are listed with their suggested fingering. There is not a drill for the final six symbols because they do not appear in programming statements other than perhaps a PRINT statement or a variable name. You press any symbol to continue the program.

After all the symbols have been introduced, you have a choice of starting the instruction over, having a final review, or ending the program. The final review consists of ten phrases or statements chosen randomly from the fifteen possibilities. You type the ten statements, and are given a score of number right and number wrong.

The program uses color graphics and sound to enhance the instruction. Musical phrases from Chopin's "Fantasie Impromptu" are played at the title screens and after each drill.

Programming Techniques

This is a full-memory program so some conservation techniques were necessary. PRINT statements are "stacked" by using colons to indicate new lines. Also, DATA statements contain over three lines of data (less than 112 characters). There are no REM statements (except for the title header). The CALL SOUND statements contain only the melody note; there wasn't enough memory for accompaniment.

In lines 430-760, graphics statements are interspersed among the CALL SOUND statements to define the special characters and colors while the music is playing.

Statements 810-880 and 4370-4450 print the keyboard a line at a time rather than using HCHARS, which are slower.

The drill phrases and typed responses are on a yellow background. This makes it easier to type in responses underneath the prompt. For example, you can tell if you have pressed the space or not.

Nine phrases are read in as string variables in data. The phrases in the DATA statements must be enclosed in quotes because they contain commas and other symbols that would otherwise be unacceptable in a DATA statement. For phrases that contain quote marks, three quotes must be used if the quote is at the beginning or end of the phrase, and two quotes if it is within. An example is: 500 DATA "PRINT ""HI"""

In a FOR-NEXT loop (statements 1530-1740, 3590-3790, 3880-3960), one of the phrases is chosen randomly. It is printed on the screen using HCHAR. A user's response is received by a CALL KEY statement and printed on the screen a letter at a time as it is typed, and combined in another string variable, B$. This method is slower than using PRINT and INPUT statements, but has the advantage of not scrolling the screen, not messing up graphics, and avoiding user input error. Only 28 strokes are allowed for a user to type. No backspaces are permitted. This is an instructional program for learning the symbols, therefore users should not be fast enough typists to notice the relatively slow response time.

Typed responses are compared with the prompts. If they are the same, an arpeggio is played and the phrase P$(J) is set to "" (null) so it cannot be used again. If the typed phrase is incorrect, a low noise is sounded, and 1 is subtracted from I in the loop counter. You must type 5 correct phrases to get out of the loop and complete the drill. That phrase also remains in the batch of available ones for possible use in the drill. Each time a new phrase is picked randomly, it is checked to see if it is null. If so, another phrase must be chosen.

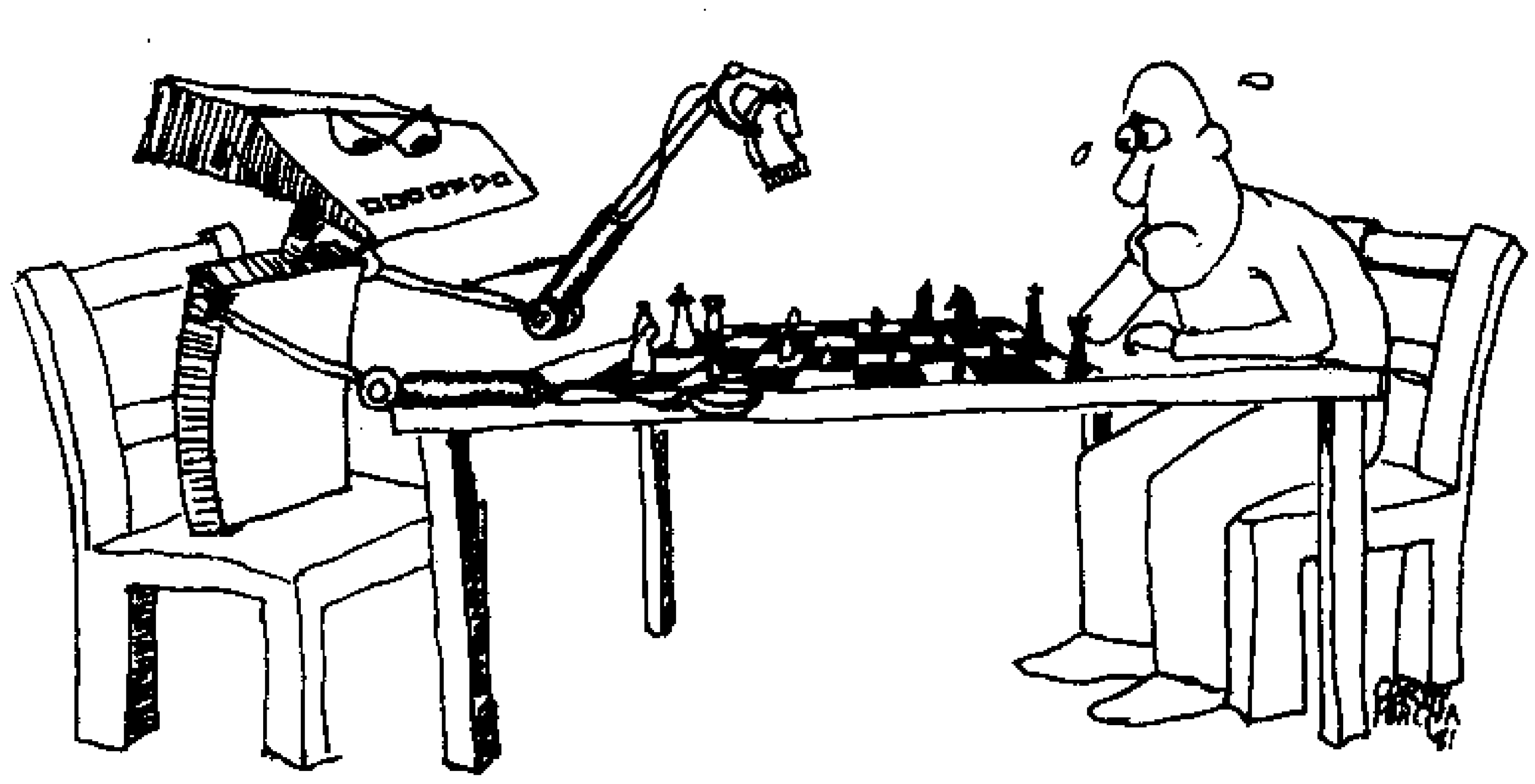
EXPLANATION OF THE PROGRAM
Learning the Keyboard Symbols

Line Nos.	
170	Dimensions P$ to allow storage of 15 phrases for drill.
180-260	Clears the screen and prints the title; defines graphics characters.
270-410	Draws TI-99/4 console.
420-860	Plays music and defines special graphics characters and colors.
870-1280	Prints subtitle screen and plays music while symbols are placed on the keyboard.
1290-1390	Instructions for period or decimal.
1400-1500	Prints the keyboard with the numbers and decimal visible.
1510-1780	Drill for decimal.
1790-1870	Instructions for comma.
1880-1950	Drill for comma.
1960-2110	Instructions and drill for – and +
2120-2370	Instructions and drill for (and)
2380-2560	Instructions and drill for * / ^
2570-2700	Instructions and drill for = <>
2710-2850	Instructions and drill for " and $
2860-3060	Instructions and drill for : ; &
3070-3260	Instructions and drill for #
3270-3360	Instructions for all other symbols.
3370-3450	Prints menu screen for choice to start over, have final review, or end program.
3460-3810	Final review of ten phrases; includes scores.
3820	Returns to menu screen.
3840-4100	Subroutine for performing the drills.
4110-4170	Subroutine for correctly typed phrase.
4180-4310	Subroutine for printing the given and typed phrase.
4320-4360	Subroutine for incorrectly typed phrase.
4370-4450	Subroutine for printing the keyboard.
4460-4600	Subroutine for drawing the right hand fingers.
4610-4660	Subroutine for blinking the middle finger red.
4670	END.

```
100 REM  ******************
110 REM  * TYPING SYMBOLS *
120 REM  ******************
130 REM  99'ER VERSION 7.81.1
140 REM  BY REGENA
150 REM
160 REM
170 DIM P$(14)
180 CALL CLEAR
190 CALL CHAR(152,"0")
200 CALL CHAR(153,"FFFFFFFFFFFFFFFF")
210 CALL CHAR(154,"00FFFF0000FFFF")
220 PRINT TAB(9);"T Y P I N G"::TAB(11);
    "ON  THE"::TAB(11);"TI 99/4":::::::
    ::::::::
230 CALL CHAR(155,"000000FFFF")
240 CALL CHAR(156,"FF8F8F8FFFF8F8F8")
250 CALL CHAR(157,"FFFFFFFF")
260 CALL CHAR(158,"F0F0F0F0F0F0F0F0")
270 CALL COLOR(16,2,12)
280 FOR X=13 TO 23
290 CALL HCHAR(X,7,152,18)
300 NEXT X
310 CALL HCHAR(15,7,155,18)
320 CALL HCHAR(13,20,154,4)
330 CALL HCHAR(14,20,154,4)
340 FOR Y=20 TO 23
350 CALL VCHAR(16,Y,153,7)
360 CALL HCHAR(Y-2,8,153,11)
370 NEXT Y
380 CALL HCHAR(22,8,153,11)
390 CALL HCHAR(19,9,156,9)
400 CALL HCHAR(20,9,156,9)
410 CALL HCHAR(21,10,157,7)
420 T=200
430 CALL SOUND(6*T,440,5)
440 CALL CHAR(96,"FFFFFFFFFFFFFFFF")
450 CALL CHAR(97,"00000000003C7EFF")
460 CALL CHAR(104,"0F0F0F0F0F0F0F0F")
470 CALL CHAR(105,"03070F0F0F0F0F0F")
480 CALL CHAR(106,"C0E0F0F0F0F0F0F0")
490 CALL CHAR(107,"F0F0F0F0F0F0F0F0")
500 CALL SOUND(40,494,5)
510 CALL SOUND(40,554,5)
520 CALL SOUND(2*T,494,4)
530 CALL CHAR(124,"00000000003C7EFF")
540 CALL SOUND(T,440,4)
550 CALL CHAR(120,"0F0F0F0F0F0F0F0F")
560 CALL SOUND(2*T,587,3)
570 CALL CHAR(121,"03070F0F0F0F0F0F")
580 CALL CHAR(122,"C0E0F0F0F0F0F0F0")
590 CALL SOUND(T,659,3)
600 CALL CHAR(123,"F0F0F0F0F0F0F0F0")
610 CALL SOUND(6*T,740,2)
620 CALL CHAR(128,"007446654474")
630 CALL CHAR(129,"00BE88888888")
640 CALL CHAR(130,"00EE8ACC8AEA")
650 CALL CHAR(131,"007A427B0A7A")
660 CALL SOUND(6*T,880,1)
670 CALL CHAR(132,"005754D75454")
680 CALL CHAR(133,"00BE08080808")
690 CALL CHAR(134,"0")
700 CALL CHAR(135,"FFFFFFFFFFFFFFFF")
710 CALL SOUND(3*T,784,2)
720 CALL COLOR(13,2,7)
730 CALL COLOR(9,12,1)
740 CALL COLOR(10,12,1)
750 CALL SOUND(3*T,740,2)
760 CALL COLOR(12,12,1)
770 CALL SOUND(3*T,659,2)
780 CALL SOUND(2*T,740,2)
790 CALL SOUND(T,587,3)
800 CALL SOUND(6*T,440,4)
810 R$=CHR$(152)
820 FOR I=1 TO 9
830 R$=R$&"  "&CHR$(152)
840 NEXT I
```

Continued on p. 76

COMPUTER CHESS CORNER



By Jerry Wolfe

Ever wondered where your computer got the "intelligence" to beat you in a game of chess? It's all in the program, you say? But then where did chess-playing computer programs come from? You might suppose that the impetus for the development of these programs came from chess players themselves. But in fact, this was not the case at all. It was researchers in the field of artificial intelligence (psychologists and computer scientists) who we have to thank for those embarrassing checkmates . . .

The goal of these researchers was to determine the nature of intelligence itself: what precisely it was, and consequently, what it was not. This was no easy task. They hoped to shed some light on this problem by getting computers to do things that if performed by a human would require "intelligence." It didn't take long to figure out that chess was a natural: It presumably required highly intelligent behavior, and yet, it was "contained" enough so that initial programs designed just to play "legal" games would not be prohibitively large. As these programs were developed, it soon became obvious that to progress from legal games to good, or even just reasonable play, required close attention to basic theory and concepts as understood by humans. For example, the number of possible positions after only the first ten moves in a game is a number having over a *hundred* zeros in it! Hence, looking at all possible positions is clearly impossible.

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.

As a consequence of this recognized need for a higher level of understanding of the game, strong chessplayers had to be consulted. One of these was international master David Levy of Scotland. Levy is perhaps best known for his $10,000 bet (made in August 1968) that even within a decade, there still wouldn't be a computer program that could defeat him in a match. In the years since his bet (which he won easily), Levy has been a frequent visitor at computer conferences, where he lectures and plays simultaneous exhibitions against several of the current programs. Incidentally, he also acted as a consultant to Texas Instruments in the development of the *Video Chess* program.

"Even though they play relatively strong chess, chessplaying programs have certain characteristic weaknesses which can often be exploited."

Levy has therefore provided a valuable link between the artificial intelligence community and the large community of chessplayers. He, perhaps more than anyone else, has been in the best position to measure the rate of computer chess progress. In his view (and mine as well) the rather recent advent of microprocessor chessplaying machines will make chess popular and accessible as never before. The revolution has just begun!

As indicated above, chessplaying programs do not attempt to find a move by searching all possible combinations of moves. Rather, chess programs combine chess theory and concepts together with brute force searching techniques to choose a move, Therefore, they are limited by how well the program "understands" chess theory and can "think" like a human player, and by speed and memory considerations. The speed and available memory determine how far ahead the program can look and how many positions can be examined and evaluated in a given amount of time. The number of moves the program can look ahead in a given position is called its "search horizon" (Levy's term).

For these reasons, even though they play relatively strong chess, chessplaying programs have certain characteristic weaknesses which can often be exploited. For example, a program may sacrifice a bishop or a knight on one side of the board to win a rook (with a knight usually) in a corner on the other side, and leave the knight trapped after it captures the rook. To any human chessplayer, it would be evident that the knight was permanently trapped and would eventually be lost—leaving the player with only a rook (5 units) to show for the loss of two minor pieces (a total of 6 units). However, the computer would merely consider the situation a gain of two units (lose a bishop or knight and gain a rook) as long as the stranded knight could not be captured within the number of moves in its search horizon. The limited search horizon leads to other situations where short term expedients are followed to the detriment of position.

Future improvements in speed will extend the search horizon of chess programs and thereby increase their playing strength even further. In my opinion, without considerable improvement in the longer range strategic capabilities of these programs, they will not be able to reach the level of world-class human players. However, we players in the other 99.99% had better watch out!

As an experiment, I recently pitted my *Video Chess* (a TI Command Module) program against the Boris machine with the Morphy module. Boris-Morphy is reputedly the strongest commercially available microprocessor chessplaying machine. The match consisted of playing the Video Chess program at its highest level (Intermediate, 200 seconds per move) against the Boris-Morphy machine at three different levels from high to low. Although the Boris-Morphy program won all three games, the Video Chess program obtained a winning position against the two lower levels, but could not find the knock-out punch. The top level of Boris-Morphy seems clearly stronger than Video Chess. All in all, the results were not bad and since the top current level of Video Chess is called "Intermediate," we may look forward to further strengthening of the program. I hope to report on similar experiments with other machines in future articles.

The two problems I'll leave you with this time are both from games by famous chessplayers. The first position is from a game of "speed" chess played in 1912 between American Edward Lasker (who died recently at age 96!) and former English champion Sir George Thomas. The rules were, I believe, that neither player could allow his own clock to get more than five minutes ahead of his opponent's clock. To find such a pretty mating combination at that speed is impressive. The second position was played by the great American champion Harry Nelson Pillsbury near the turn of the century in an exhibition where he played blindfolded against 22 different opponents simultaneously! Blindfold play is not as difficult as you might think—try it against your Video Chess program sometime—but to (successfully) play 22 such games is phenomenal. In recent times George Koltanowski has played blindfolded against more than 50 opponents simultaneously. But Pillsbury's achievement is magnified by the fact that he could perform well in blind simultaneous play against *masters!*

SOLUTIONS TO THE PROBLEMS IN THE LAST ISSUE:

Problem No. 1:

1. D3 - D8 check E8 - D8
2. D2 - G5 double check.
(a) 2 . . . D8 - E8
3. D1 - D8 checkmate
(b) 2 . . . D8 - C7
3. G5 - D8 Checkmate.

Problem No. 2: **1 . . . C3 - G3 !!**

Black appeared to be in trouble since after the apparently forced retreat of his queen out of danger, white could capture the rook on H3 and be decisively ahead in material. Black had forseen all this, however, and replied with the crushing move above. White has three ways to capture the black queen (which must be captured else mate on H2 is inevitable)—all unsatisfactory.

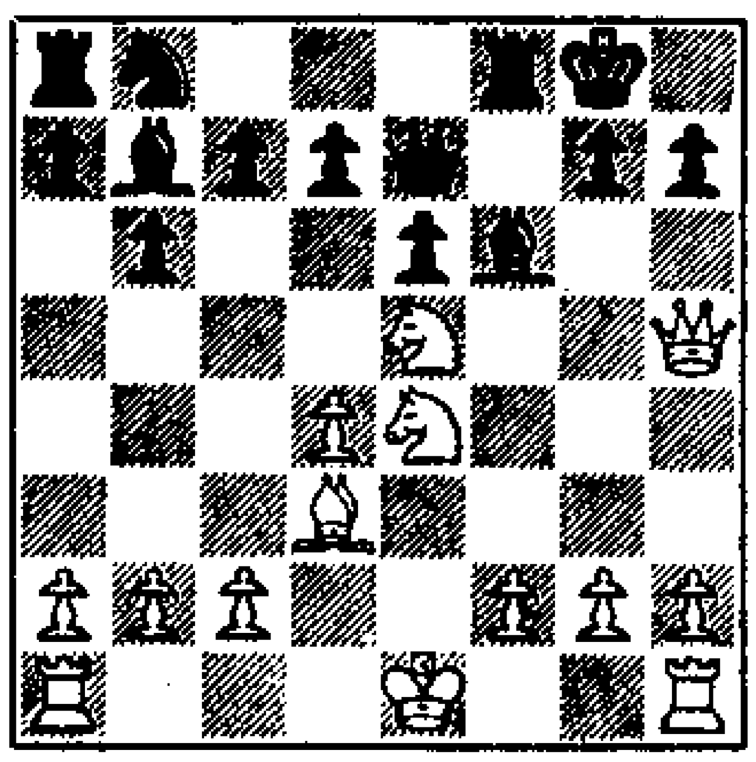
(a) 2.	H2 - G3	D4 - E2	checkmate.
(b) 2.	F2 - G3	D4 - E2	check.
3.	G1 - H1	F8 - F1	checkmate.
(c) 2.	G5 - G3	D4 - E2	check.
3.	G1 - H1	E2 - G3	check.
4.	H1 - G1	G3 - E2	check.
5.	G1 - H1	H3 - C3	

and black is a full piece ahead with an easy win.
In the actual game, white resigned after 1 . . . C3 - G3.

New Problems (The solutions with appear in our next issue.)

Problem No. 1

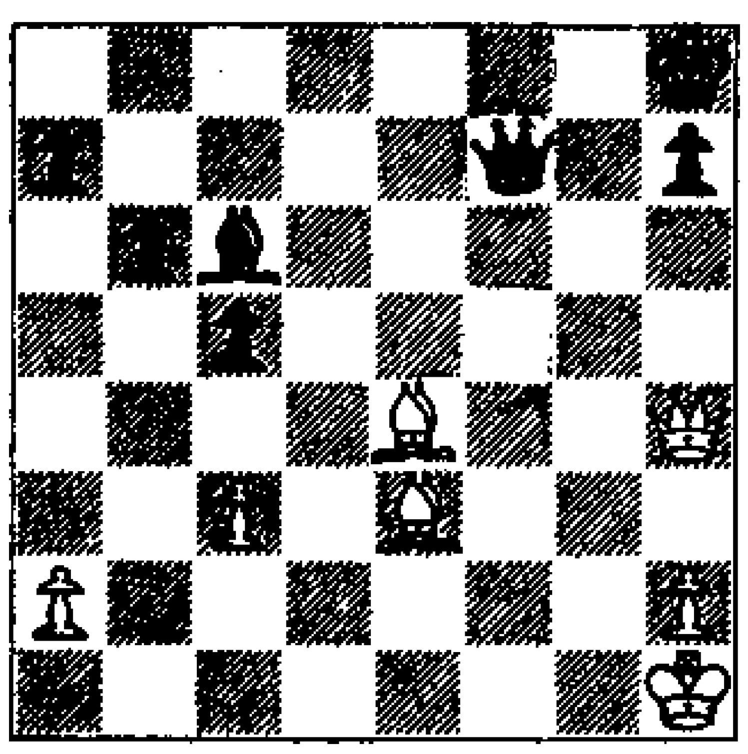
White:	Pawns:	A2, B2, C2, D4, F2, G2, H2
	Knights:	E4, E5
	Bishops:	D3
	Rooks:	A1, H1
	Queen:	H5
	King:	E1
Black:	Pawns:	A7, B6, C7, D7, E6, G7, H7
	Knights:	B8
	Bishops:	B7, F6
	Rooks:	A8, F8
	Queen:	E7
	King:	G8



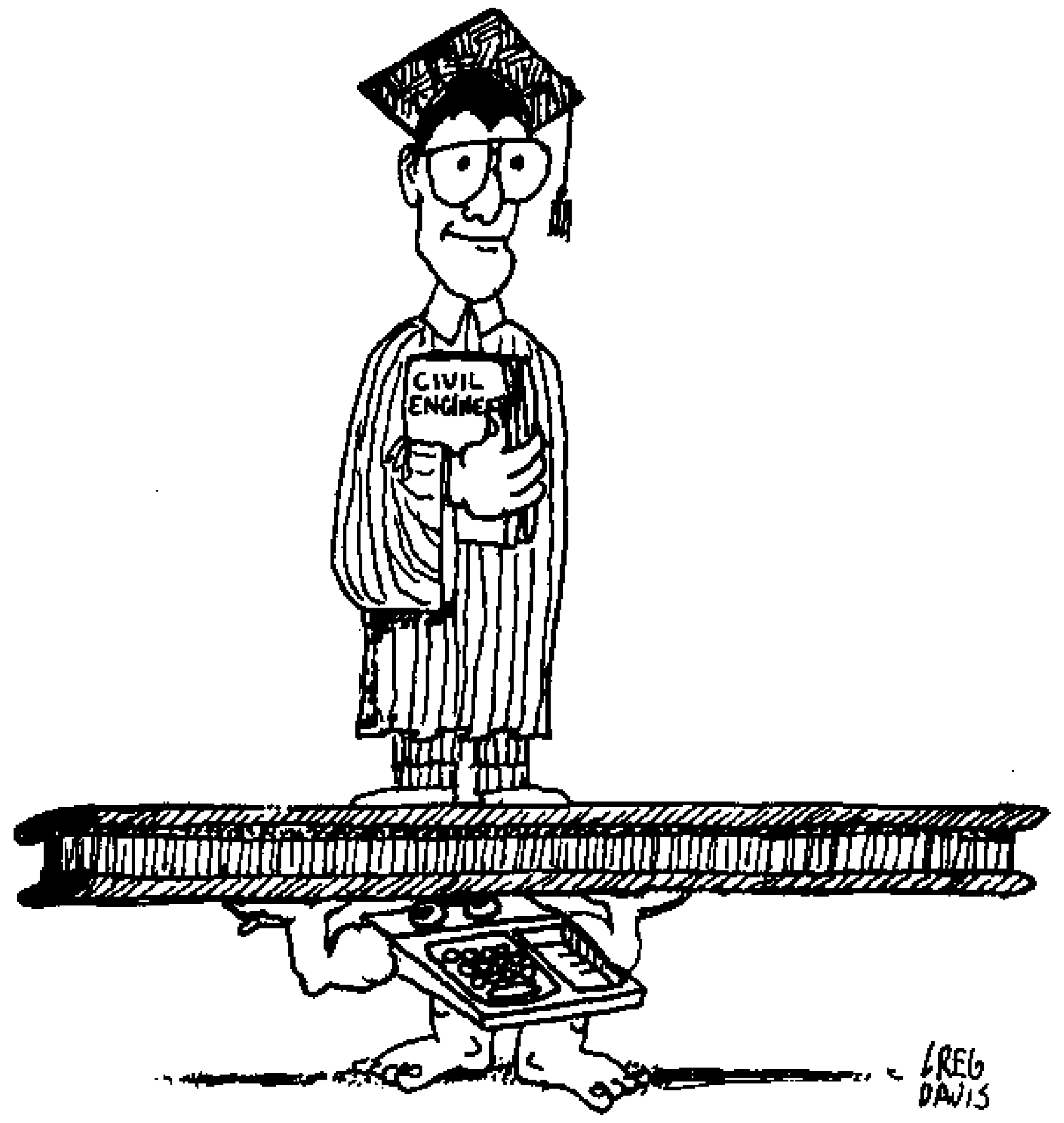
White to move and mate in several moves. Can you find the fewest necessary?

Problem No. 2

White	Pawns:	A2, C3, H2
	Knights:	none
	Bishops:	E3, E4
	Rooks:	none
	Queen:	H4
	King:	H1
Black:	Pawns:	A7, B6, C5, H7
	Knights:	none
	Bishops:	C6
	Rooks:	none
	Queen:	F7
	King:	H8



Black to move and mate in three moves.



Civil Engineering Fundamentals

Simple Beams

By Regena

The purpose of this program is to tutor civil engineering students who are studying statics or structures. It is limited to a simple determinate beam supported at the ends, and loaded with a concentrated load, a uniform load, or a combination of a concentrated load and a uniform load. A basic knowledge of elementary statics is prerequisite.

There are six sections in the program. The student's objective is to learn how to solve for the reaction forces A and B at each end of the loaded beam.

1. Concentrated load at the center

Newton's laws of force and moments are reviewed. The general solution of a load P applied at the center on a beam of length L is developed for the reaction forces A and B at each end of the beam. The student then does two problems. The load P and length L are chosen randomly for the problems. If he enters an incorrect solution, the correct solution is given, and he is given another problem.

2. Concentrated load anywhere

Newton's laws of force and moments are reviewed. The general solution of a load P applied a distance D from end A on a beam of length L is derived for the reaction forces A and B at each end. An example problem is given and solved. Then a problem is given for which the student enters his answers. The program prints the method of solution. For the next problem the student enters his solution. If he is incorrect, the program shows him how to solve the problem, and he is given another problem to solve.

3. Uniform load

The uniform load is considered as an equivalent concentrated load acting at the centroid of the loading pattern. The first example is a uniform load for the length of the beam, and is solved in general terms. The student is then given a problem. If he enters an incorrect answer, he is shown the correct solution and given another problem.

If the student is correct, a sample problem with uniform loading over part of the beam is presented. Then he is given a problem of this type and asked to solve it. If he is incorrect, the solution is shown and he is given another problem with this type of loading.

4. Combination loads

Instructions are provided for how to solve a beam with one concentrated load and one uniform load. The student is then given a problem with combination loads chosen randomly. The program draws and labels the beam for each problem. If the student enters an incorrect solution, the correct solution is printed and he is given another problem.

5. Problems only

No instruction is given. The program randomly chooses a beam length and loading pattern, and prints the problem. It then draws and labels the beam. The student enters his answers; if he is incorrect, the correct answers are given and another problem is printed.

6. Your own problems

The student enters the beam length and loading specifications. The program computes the reaction forces A and B at the ends.

After each section has been completed with correct solutions, the student is given the choice of having more of the same kind of problems, entering his own problems, or returning to the menu screen.

Programming Techniques

This program is a teaching aid or tutor, so it incorporates pauses, allowing the student to work on the problem before continuing. The student must enter a correct solution to the problem before he or she can go on to a different kind of problem. If the student enters an incorrect solution, the correct answers are printed and another problem of the same type is presented.

The numbers for each problem are chosen randomly (yet appropriately) for each beam. The length of the beam is between 10 and 20 feet. The concentrated load is 100 times a random number from one to twenty (i.e., 100 to 2000 pounds), and is placed at a distance D from end A (randomly chosen within the bounds of the length of the beam).

The uniform load is 10 times a random number from one to ten (i.e., 10 to 100 pounds per foot). For some of the problems, the uniform load is acting over the length of the beam. For more advanced problems, it is between two points on the beam measured by the distance from end A (as L1 and L2). L1 must be equal to or greater than zero, and less than the total length of the beam. L2 must be greater than L1, and less than or equal to the total length of the beam.

The problems are written in "story problem" form by using print statements in subroutines, with the program using only the statements that are necessary for each loading condition.

After the student has had time to draw and label the problem on his own paper, he can "PRESS ENTER TO CONTINUE" and the beam will be drawn on the screen with approximate proportions.

The general beam is drawn with a pin at end A and a roller at end B.

The distances are approximated by using a variable y-coordinate—an integer value of the fraction of the distance (D or L1) divided by the total beam length multiplied by the number of characters printed in the general beam. For example, Statement 6750 is

Y=INT(L1/LL* 22)+6

Y is the y-coordinate used in CALL HCHAR or CALL VCHAR statements. 6 is the displacement of the end of the beam from the left side of the screen.

In statement 6760:

Z=INT(D2/LL* 22)–1

Z is the number of characters to be printed horizontally for the uniform load. D2 is the distance L2–L1.

The labels for the values on the beam are variable and are printed using string variables. For example, the concentrated load P may be three or four digits long (100 pounds to 2000 pounds) in the written problems, but the student may input an even longer number. This label is printed by using statements 5850-5930 (see Figure 1).

Figure 1.

LB$=STR$(PP)	Converts PP to a string variable.
FOR II=1 TO LEN(LB$)	LEN finds the length of LB$.
JJ=II+J–4	Calculation for y-coordinate.
CALL HCHAR(I–5,JJ,ASC(SEG$(LB$,II,1)))	Prints each digit in order.
NEXT II	
CALL HCHAR(I–5,JJ+1,32)	Prints a space after last digit.
CALL HCHAR(I–5,JJ+2,76)	Prints L.
CALL HCHAR(I–5,JJ+3,66)	Prints B.
CALL HCHAR(I–5,JJ+4,83)	Prints S.

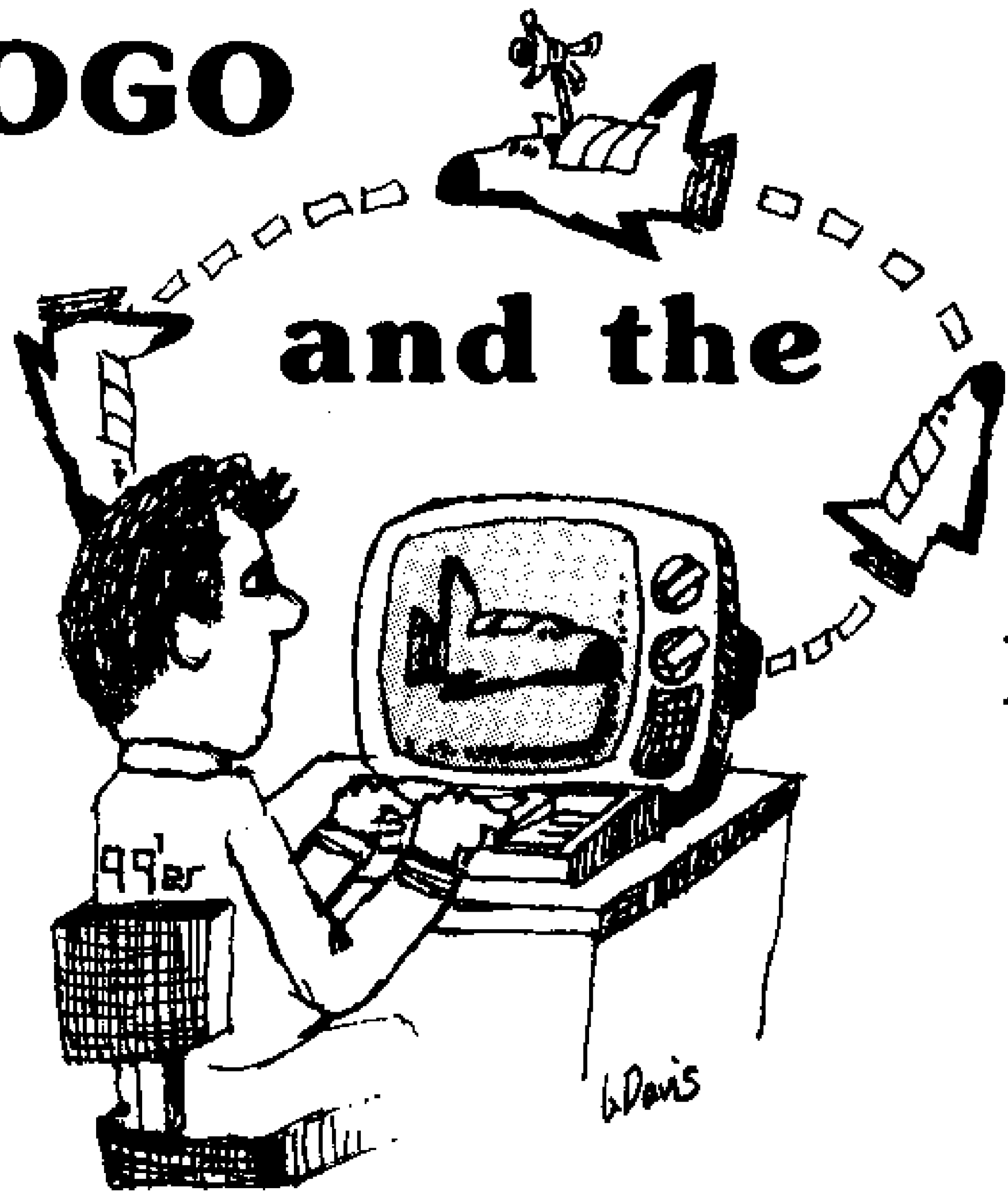
EXPLANATION OF THE PROGRAM

Line Nos.	
100-250	Prints the title screen.
260-330	Blinks a blue border.
340	Clears the screen.
350-540	Defines special graphics characters for drawing the beam and loading and sets colors for them.
550-680	Prints second screen, diagram of simple beam.
690	Goes to menu screen for choice of problems.
700	Choices 1 and 2, concentrated loads, branch to here.
710-810	Prints instruction screen.
820-900	Prints second instruction screen.
910	For Choice 2, branches to 1720
920-1000	Prints problem.
1010-1070	Draws and labels general beam.
1080-1180	Shows solution of reaction forces in general terms.
1190-1270	Draws and labels beam with centrally applied load.
1280-1330	General statement for central load.
1340-1370	Chooses random numbers for problem.
1380-1400	Writes the problem.
1410-1440	Draws and labels the beam.
1450	Asks for A and B from student.
1460-1540	Compares student's answers with calculated solution and prints appropriate remark.
1550-1580	Has another problem.
1590-1610	Asks if student wants more problems and branches accordingly.
1620-1700	Draws and labels a beam for student's problem.
1710	Solves and checks it.
1720-1790	Prints instructions for second type of beam, concentrated load anywhere.
1800-1870	Draws and labels beam.
1880-1970	Solves the problem.
1980-2050	Chooses a problem and prints it.
2060-2160	Draws and labels the beam.
2170-2190	Solves the problem.
2210-2240	Compares input answers with calculated solution.
2250-2270	If student is incorrect, solves the problem in detail.
2280-2290	Returns for another problem.
2300-2330	Solution was correct. If it is the second problem, another problem is chosen.
2340-2360	Offers the student the choice for more problems.
2370-2530	Solves a problem the student enters.
2540-2600	Prints the general problem for a uniform load.

```
100 REM  ** CIVIL ENGINEERING **
110 REM
120 REM
130 REM  BY REGENA
140 REM
150 REM  99'ER VERSION 7.81.1
160 REM
170 CALL CLEAR
180 PRINT TAB(7);"CIVIL ENGINEERING"
190 PRINT ::TAB(9);"FUNDAMENTALS"
200 CALL COLOR(2,1,1)
210 PRINT ::::::TAB(7);"****************"
220 PRINT TAB(7);"*              *"
230 PRINT TAB(7);"* SIMPLE BEAMS *"
240 PRINT TAB(7);"*              *"
250 PRINT TAB(7);"****************"::::::
260 FOR E=1 TO 10
270 CALL COLOR(2,6,5)
280 FOR DELAY=1 TO 75
290 NEXT DELAY
300 CALL COLOR(2,5,6)
310 FOR DELAY=1 TO 75
320 NEXT DELAY
330 NEXT E
340 CALL CLEAR
350 CALL COLOR(2,2,1)
360 CALL COLOR(9,2,1)
370 CALL COLOR(12,11,1)
380 CALL CHAR(120,"0F0F0F0F0F0F0F0F")
390 CALL CHAR(121,"FFFFFFFFFFFFFFFF")
400 CALL CHAR(122,"F0F0F0F0F0F0F0F0")
410 CALL CHAR(99,"1824242442428181")
420 CALL CHAR(100,"1824428181422418")
430 CALL CHAR(101,"0F09122449")
440 CALL CHAR(102,"FF24499224")
450 CALL CHAR(103,"F89020408")
460 CALL CHAR(104,"1010383854549292")
470 CALL CHAR(105,"1010101010101010")
480 CALL CHAR(106,"9292545438381010")
490 CALL CHAR(112,"FFFFFFFFFFFFFFFF")
500 CALL CHAR(113,"F0F0F0F0F0F0F0F0")
510 CALL CHAR(114,"0F0F0F0F0F0F0F0F")
520 CALL CHAR(115,"FF")
530 CALL CHAR(98,"FF601806083040FF")
540 CALL COLOR(11,6,1)
550 PRINT TAB(8);"SIMPLE BEAM":::
560 PRINT TAB(5);"SUPPORTED AT ENDS":::::::::::::
570 I=17
580 GOSUB 5380
590 J=12
600 GOSUB 5560
610 CALL HCHAR(I-5,12,80)
620 FOR L=I-3 TO I-1
630 CALL HCHAR(L,17,112,10)
640 NEXT L
650 CALL VCHAR(I-3,27,113,3)
660 CALL HCHAR(I-2,19,87)
670 PRINT "  FIND THE REACTION FORCES"
```

Continued on p. 79

TI LOGO and the Space Shuttle:



AN UPDATE ON LAMPLIGHTER ACTIVITIES

By Henry Gorman Jr.

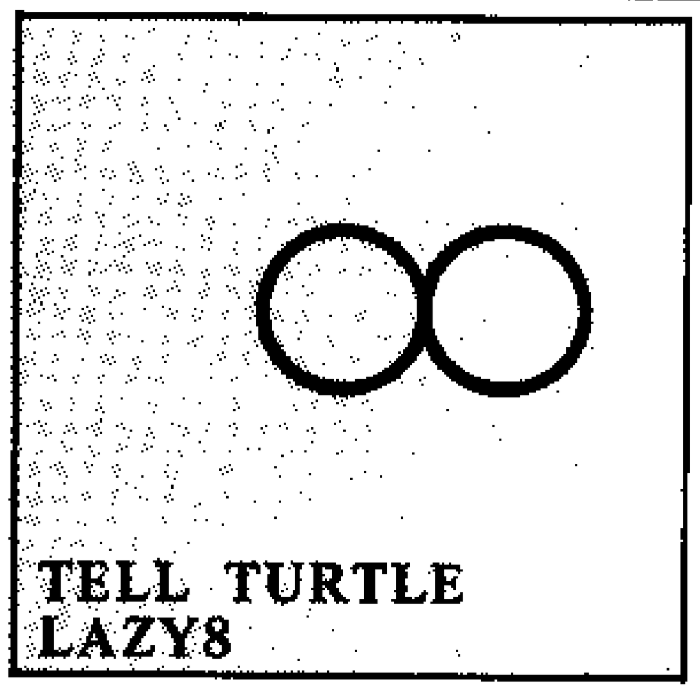
Department of Psychology, Austin College, Box 1584, Sherman, TX 75090

The previous issue of *99'er Magazine* (May/June 1981) described how the Lamplighter school (a private school in Dallas for children from age 3 through the fourth grade) became the very first to provide its students with a truly computer-rich learning environment through its TI LOGO implementations. At the time the article was written, most of the students had been working steadily in LOGO for only six months. Children in the nursery school and kindergarten were using teacher-written programs to explore LOGO, and the other children in grades one through four were writing their own programs. There was, as expected, an age-related trend in programming, with the fourth graders generally doing the most elaborate work—although many third graders, a few second graders, and a couple first graders have indeed produced sophisticated programs.

A few children even acquired the skill of using **subprocedures**—i.e., breaking a complex program down into its several component parts. This is one of the most important features of procedural languages such as LOGO. Most students had discovered **recursive programming**, or "cursives" as a few called it. In recursive programs one of the program lines calls for a new stack to execute the program again. You do this by including the name of the program within the program itself. All the recursive programs written by the students, however, had the recursive step in the last line. [When the recursive step occurs in the last line before END, the procedure is said to have "tail-end recursion." For an example of somewhat more sophisticated usage, see the LAZY8 procedure in Figure 1—Ed.]

Figure 1.

```
TO LAZY8
FORWARD 4
RIGHT 10
TEST HEADING = 0
IFF LAZY8
FORWARD 4
LEFT 10
END
```



Comment:

TEST checks the heading of the Turtle. If it's not 0 (North), the Turtle continues to draw the LAZY8.

After finishing the right-hand circle, the heading becomes 0 and the left-hand circle is drawn.

To really understand why the left-hand circle ever gets completed, you have to know something about microprocessors and stack operations. Keeping in line with the scope of this article, however, a simple anthropomorphic explanation will have to suffice at this time. Watch forthcoming articles for an in-depth look at the technical aspect of the language.

Think of the job of drawing the LAZY8 as being given to a group of little workmen inside the computer. The first workman carries out the first four lines then decides he needs a rest before continuing. Notice that in his initial contract TO LAZY8 he has agreed to eventually carry out the FORWARD 4 and LEFT 10 specifications. The work must go on while he rests, so he subcontracts out the next stage to another little man. This workman also carries out the first four lines, then he too decides to rest. So before he gets to the FORWARD 4 and LEFT 10 tasks, he decides to subcontract out the balance of the work on the right-hand circle. This process goes on with enough little workmen (36 in this case) until HEADING=0. At that time, the last little man carries out his FORWARD 4 and LEFT 10 tasks, and gives the job responsibility back to the next-to-last workman who also carries out his remaining FORWARD 4 and LEFT 10 tasks. This reverse process of finishing the last two tasks and relinquishing responsibility goes on until the original contractor finishes his original job with a single FORWARD 4 and LEFT 10, thus completing the left-hand circle in the LAZY8—GMK

A number of programs produced exciting video scenes. In EXPLODE, 32 differently colored balls splay out from the center of the screen to form a circle and then return back to the center before repeating the entire procedure. One third grader saw how he could place a program which printed a message inside of EXPLODE, and thus combined recursion and subprocedures. RAINBOW had one or more sprites continuously change colors for an attractive visual effect. There were also programs which had the TV monitor take on a series of sixteen colors, and programs which changed the background of the screen to black and created unusual perceptual illusions by shooting light-colored shapes across the screen. Some even had jets, rockets, or airplanes spouting fires from their engines.

Other children wrote programs which put shapes together to create scenes, such as a home with a car driving down the street in front of the home. Most students had written utilitarian programs like VANISH (Figure 2) which caused the sprites to move off screen, take on the clear color, carry an empty shape, and which caused all the printing to be cleared from the screen.

```
TO VANISH
TELL :ALL
CARRY 0
SETCOLOR 0
SETSPEED 0
SETHEADING 0
END
```

Figure 2.

After spring break, several things happened which caused a quantum leap in the computer work of the students. First, the children were shown how to save their programs and shapes on cassette tape. Until then, the students had to write in their computer notebooks anything they wanted to save. That meant that any elaborate shape had to be reproduced on a grid in an arduous manner, and long programs or complex programs required a very long time typing. (Remember these children are elementary pupils with little typing experience before computers!).

Students had not used much of their work as foundations for future work simply because loading the old material took so much of their time. Now, with the recorders, they could use and improve each session's programs just by taping and playing back a cassette. Also, they could design and SAVE complex shapes instead of seeing them lost when the computers were shut off.

The children were also shown the TELL TURTLE mode. This opened up all of the turtle geometry features of LOGO. (Turtle geometry is such a powerful idea that some Pascal systems have adopted it.) This newly acquired mode, coupled with the previously learned SPRITE MODE, allowed the students to produce many interesting programs and visual effects. As a result of these new developments, many of the students soon exhibited a feeling of mastery over the computers.

In the final eight weeks of school there was an exponential explosion in the complexity of the students' programs and in their ease with the machines. They quickly learned to use variables as inputs, and consequently "discovered" the famous turtle geometry POLYgon program which can generate any regular polygon. [See Figure 3] Then one student found that changing the angle of the turn on each recursion could produce beautiful patterns--including a striking nested curl in a star pattern. Many students now began putting programs together in subordinate and superordinate structures. Programs contained the unique LOGO controls of TEST, IFT, and IFF, as well as the conditionals IF . . . THEN . . . ELSE, plus BOTH and EITHER for conjunctive and disjunctive branching. One of the third graders wrote a CAI (Computer-Assisted Instruction) program to quiz his first grade friends on addition facts using these control commands! He then added visual displays of the addends, and encouraging remarks when a student made a mistake, or a colorful scene as a reward for the correct answer.

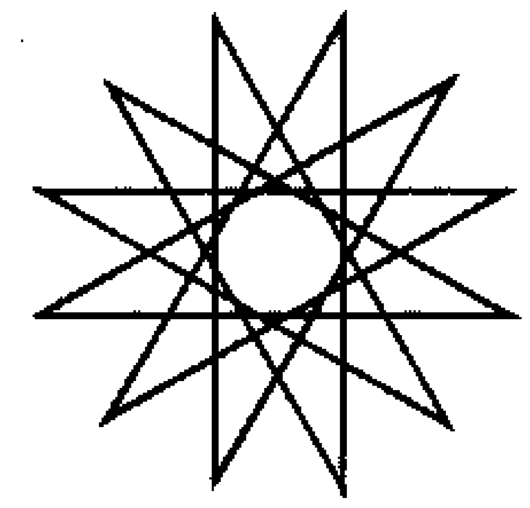
Using combinations of several user-drawn shapes, students began constructing very elaborate composite pictures. One third grade student also discovered how to change the characters associated with each console key [by redesigning the characters on a grid "tile" with the MAKECHAR primitive—Ed.], and decided to tease the teacher. She replaced the *3* with a *2*, and then called a teacher over for a demonstration. While instructing the computer to print 3 + 3 (which now looked like a request for the sum of 2 + 2), she remarked to the teacher: "Look how dumb this computer is . . . it doesn't know 2 + 2."

The activity among the third grade students was exciting to witness. One began programming dramas in which text was printed at the bottom of the screen while the story was enacted in SPRITE and TELL TURTLE modes at the top of the screen. One other third grader was so intrigued by the space shuttle's landing that on the same afternoon of the landing, he began working on a shuttle program. First, he used MAKESHAPE to construct a faithful replica of the shuttle, complete with USA monogram, black-and-white coloring, and auxiliary rocket engines. Then he worked for part of the afternoon and a little of the next morning to write and debug his programs. His final superprocedure launched the shuttle with flames shooting from the engines, jettisoned the auxiliary tanks, orbited the shuttle among planets in outer space, returned the shuttle to a dry lake-bed runway, taxied it to the end of the runway, and stopped it for a perfect landing. His programs are shown here in Figure 4.

The gains made by the Lamplighter children with LOGO have indeed been impressive. They confirm Papert's dictum [*Mindstorms*, Semour Papert, Basic Book 1980] that children should program computers and not vice-versa.

Figure 3.

```
TO POLY DISTANCE ANGLE
HIDETURTLE
FORWARD :DISTANCE
RIGHT :ANGLE
POLY :DISTANCE :ANGLE
END
```



```
TELL TURTLE
POLY 90 150
```

Figure 4

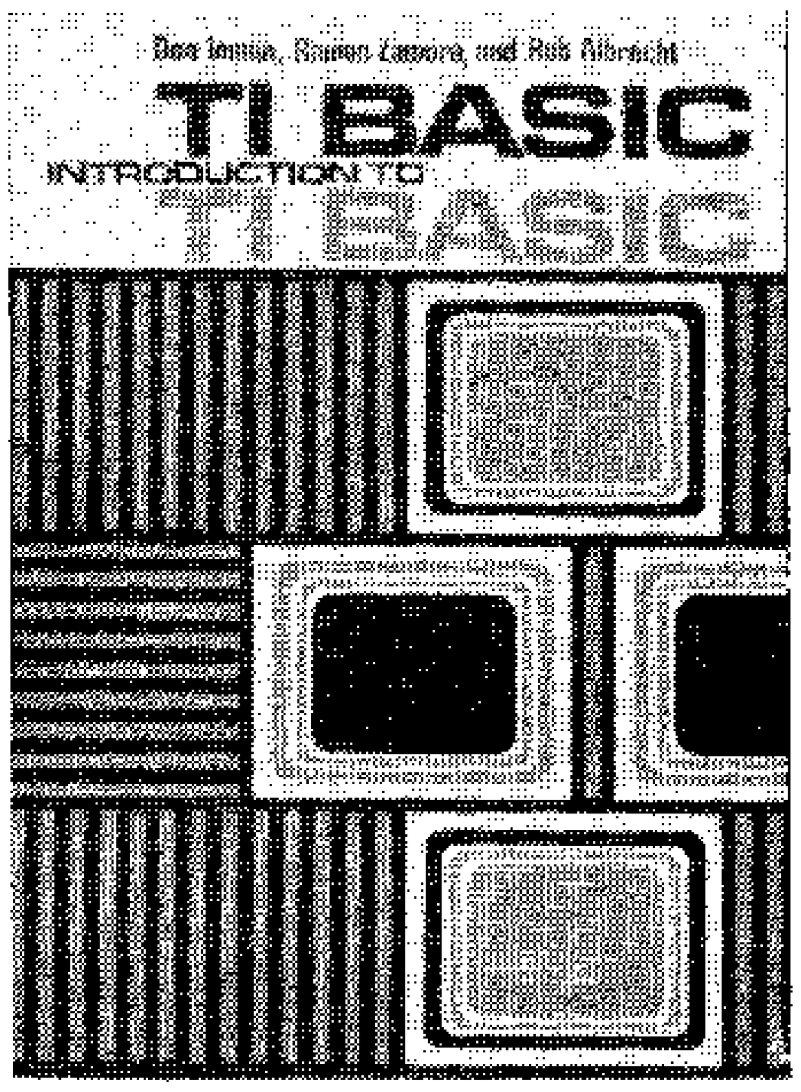
```
TO SHUTTLE
TANK
PLANE
FIRE
END

TO TANK
TELL 1 CARRY 20 SC 15 HOME
SH 90 FD 16
TELL 2 CARRY 21 SC 15 HOME
TELL 3 CARRY 22 SC 15 HOME SH 270
FD 16
TELL 4 CARRY 23 SC 6 HOME SH 270
FD 32
END

TO PLANE
TELL 5 CARRY 24 SC 15 HOME
SH 0 FD 16 SH 90 FD 16
TELL 6 CARRY 25 SC 15 HOME
SH 0 FD 16
TELL 7 CARRY 26 SC 15 HOME
SH 0 FD 16 SH 270 FD 16
TELL 8 CARRY 27 SC 15 HOME
SH 0 FD 32
SH 270 FD 16
TELL 9 CARRY 4 SC 11 HOME
SH 180 FD 47 SH 90 FD 12
TELL 10 CARRY 4 SC 8 HOME
SH 0 FD 45
SH 270 FD 23
TELL [1 2 3 4 5 6 7 8]
SH 90 SS 20
TELL BG SC 1
END

TO FIRE
TELL 4
WAIT 30
SC 0
WAIT 15 SC 6
FIRE
END
```

Note :
BG = BACKGROUND
FD = FORWARD
SC = SETCOLOR
SH = SETHEADING
SS = SETSPEED

Note:

Listings of TI LOGO procedures are just that—listings of *procedures.* There's no way to print out a transcription of the data needed to MAKESHAPE and MAKECHAR as can be done with the HEX Codes in TI BASIC and Extended BASIC. The only way to show the graphics that a program contains is to *show it as drawn* on a series of "tiles" on the grids that appear on screen when the shapes and characters are first designed. This is similar to CHARDEF routine in *Programming Aids 1.* Space in this present issue doesn't permit these graphics to be shown. In future articles when TI LOGO becomes generally available, we will reproduce the tiles associated with each program featured in the magazine. The listing of the Space Shuttle program was included (without the tiles) in this article to demonstrate the elegant simplicity of the language structure. —GMK

99'er BOOKSTORE



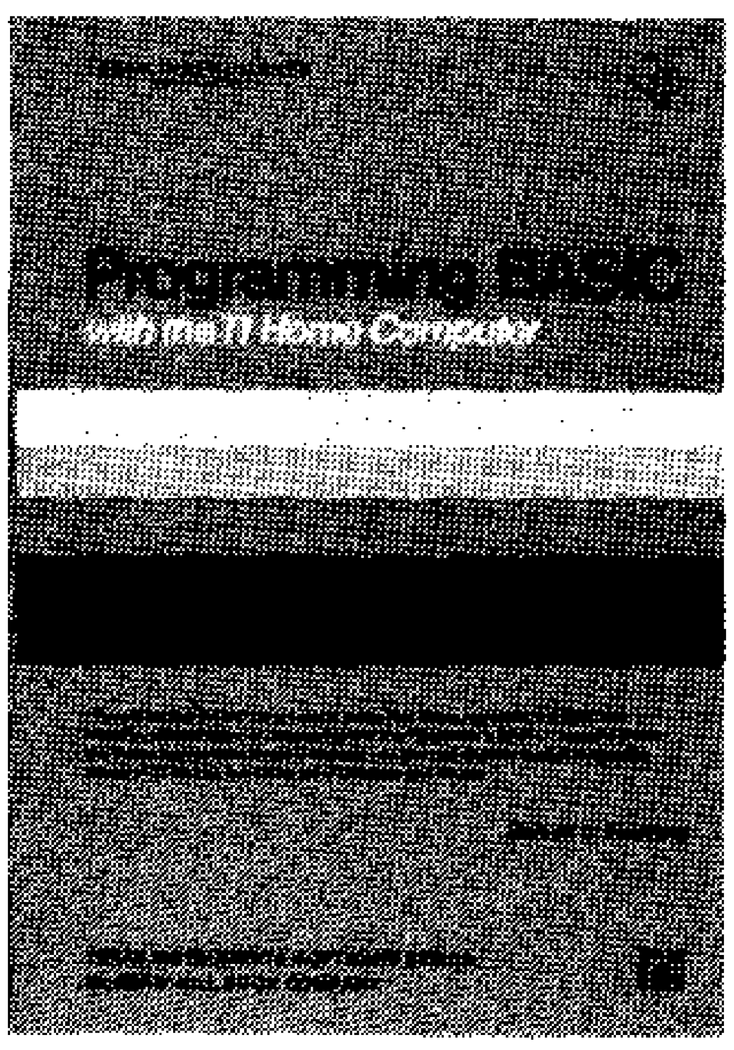
INTRODUCTION TO TI BASIC

By D. Inman, R. Zamora, and R. Albrecht.

This comprehensive work, written by three of the foremost microcomputing programming experts in the country, 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.

CONTENTS: Gateway to Adventure. Do It Now: Sound and Color Graphics. Simple Programming. Looping Sound and Color. More Programming Power. Beginning Simulation. More Program Control Statements. Using Data Files. One Dimensional Arrays. Two Dimensions and Beyond. Color, Graphics, Sound, and Animation. More Strings. Editing. Subroutines and Your Personal Library.

paper, **$10.95**
1980, 320 pages, $7^1/_8$ x $9^3/_4$.

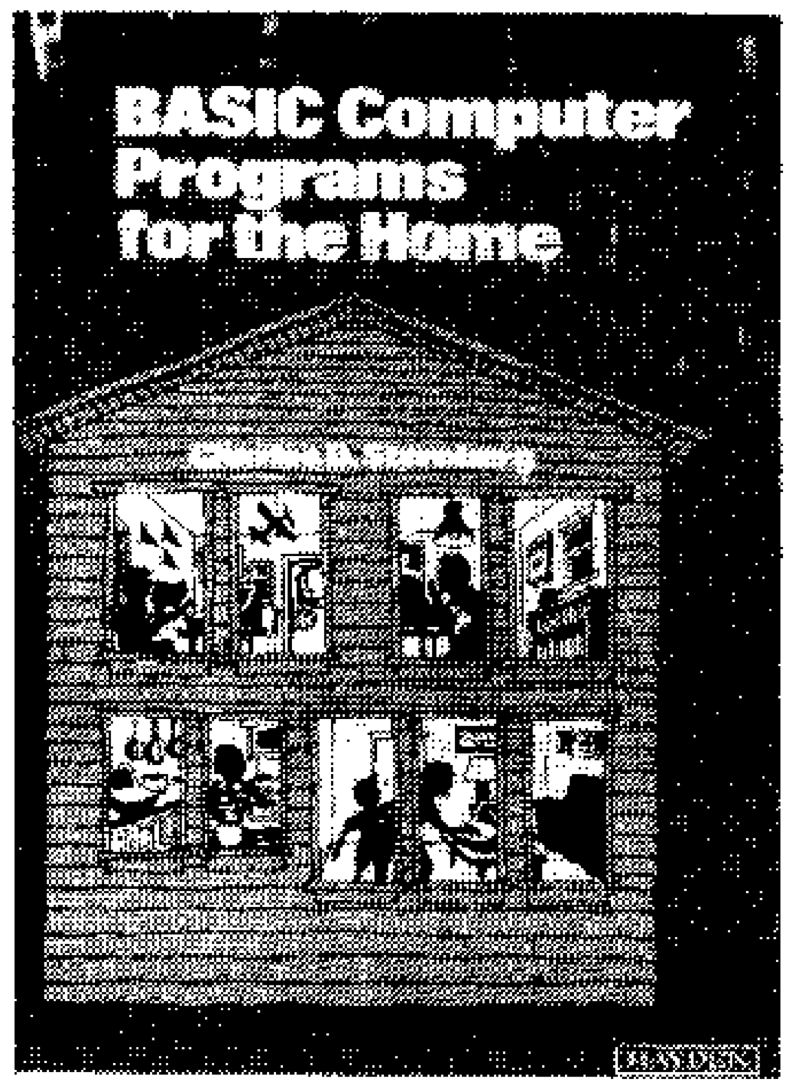


PROGRAMMING BASIC WITH THE TI HOME COMPUTER

By Herbert D. Peckham.

A tutorial guide that helps you learn TI BASIC in a friendly, relaxed manner. It goes beyond *Beginner's BASIC* furnished with the TI-99/4, and introduces the full range of TI BASIC features including color graphics and sound. Its 11 chapters are written in a complete-the-blanks, programmed instruction format.

paper, $10.95
1979, 306 pages, 6 x 9



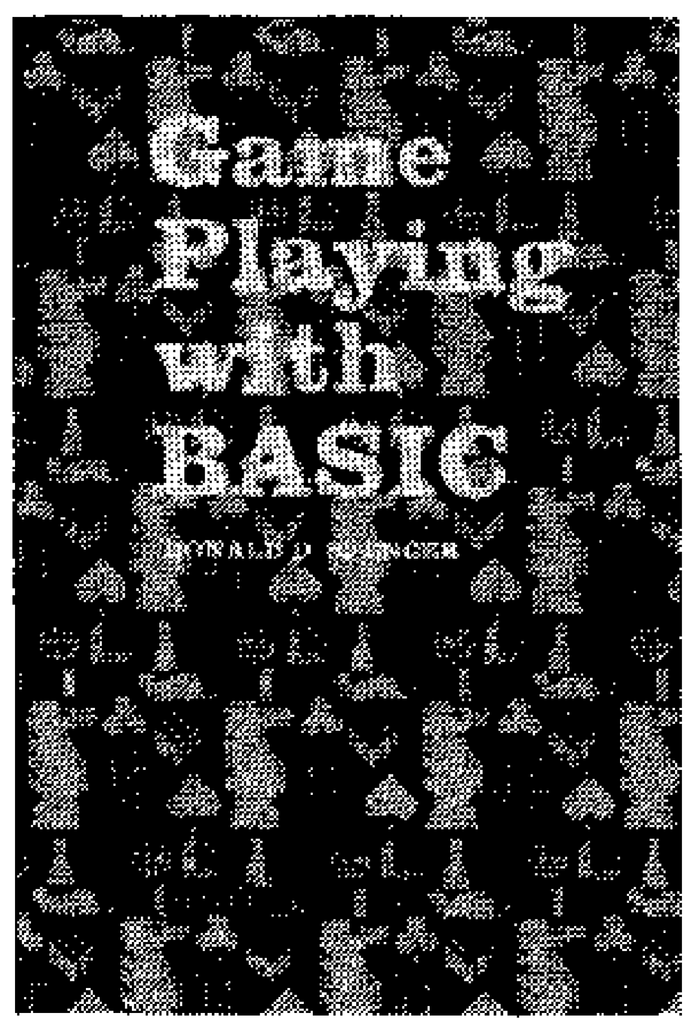
BASIC COMPUTER PROGRAMS FOR THE HOME

By Charles D. Sternberg.

The only book named in an update article in the Personal Business section of *Business Week* (June 23, 1980)!

An invaluable book at a great price, it contains over 75 practical home application programs that will be helpful to the novice or experienced owner in increasing the usefulness of any home computer. Each program is documented with a description of its functions and operation, a listing in BASIC, a symbol table, sample data, and one or more samples. Programs included are: Home Financial Programs; Automobile Related Programs; Kitchen Helpmates; Scheduling Programs for Home Use; List Programs for Every Purpose; Miscellaneous Programs for the Home; Tutorial Programs for Home Use; Conversion Program; and Hobbyist's Diaries.

paper, **$9.95**
1979, 336 pages, $7^1/_8$ x $9^3/_4$



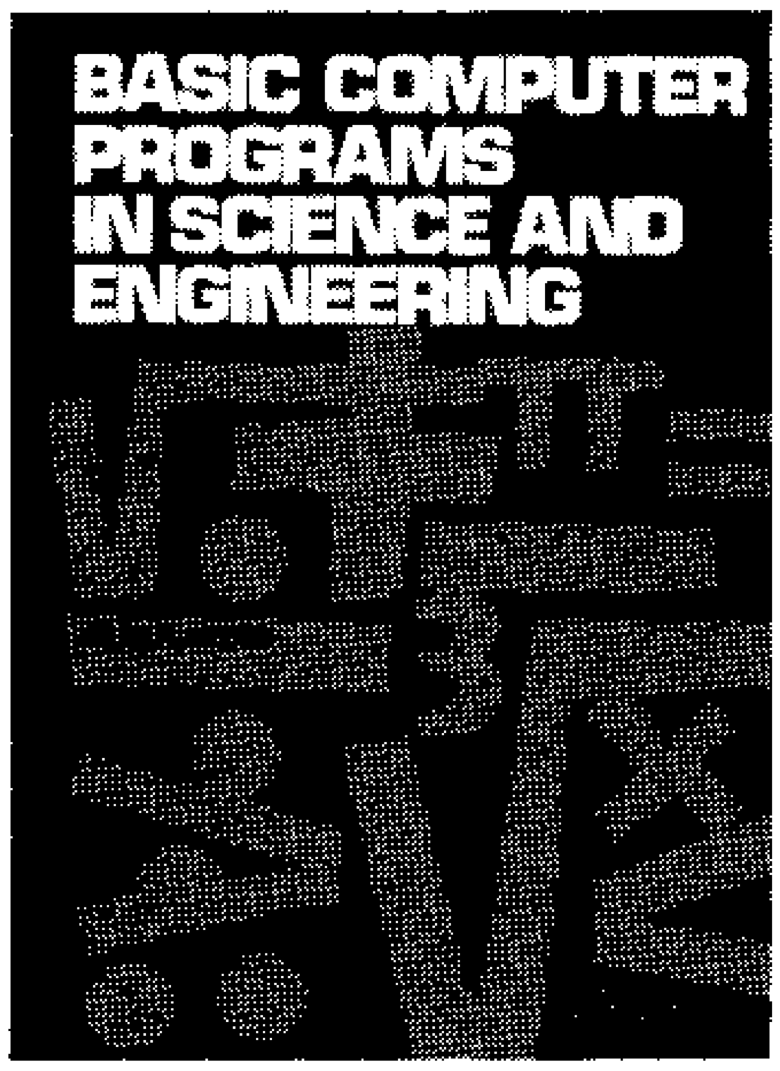
GAME PLAYING WITH BASIC

By Donald D. Spencer, Abacus Computer Corporation.

Enjoy the challenge of competition with your computer. Amuse yourself with such games and puzzles as 3-D Tic-Tac-Toe, Nim, Roulette, Magic Squares, the 15 Puzzle, Baccarat, Knight's Magic Tour, and many others. The writing is nontechnical, allowing almost anyone to understand computerized game playing. The book includes the rules of each game, how each game works, illustrative flowcharts, diagrams, and the output produced by each program. The last chapter contains 26 games for reader solution.

paper, **$9.55**
1977, 176 pages, 6 x 9, illus.

99'er BOOKSTORE



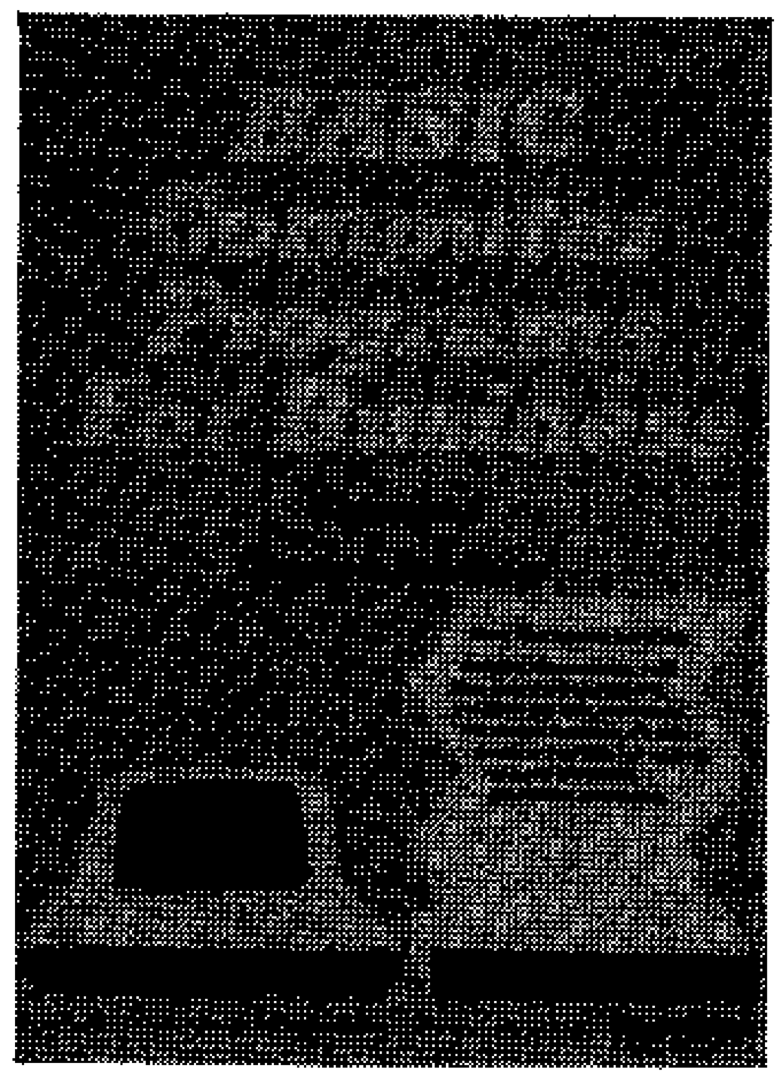
BASIC COMPUTER PROGRAMS IN SCIENCE AND ENGINEERING

By Jules H. Gilder.

Save time and money with this collection of 114 ready-to-run BASIC programs for the hobbyist and engineer. There are programs to do such statistical operations as means, standard deviation averages, curve-fitting, and interpolation. There are programs that design antennas, filters, attenuators, matching networks, plotting, and histogram programs. There is even a justified typing program that can be used in typesetting. All programs in the book have been tested and are fairly universal; so you should have no difficulty running them on your system. You won't find anywhere a more comprehensive collection of usable, ready-to-run BASIC programs!

See also software section.

paper, **$9.95**
1980, 160 pages, 6 x 9, illus.



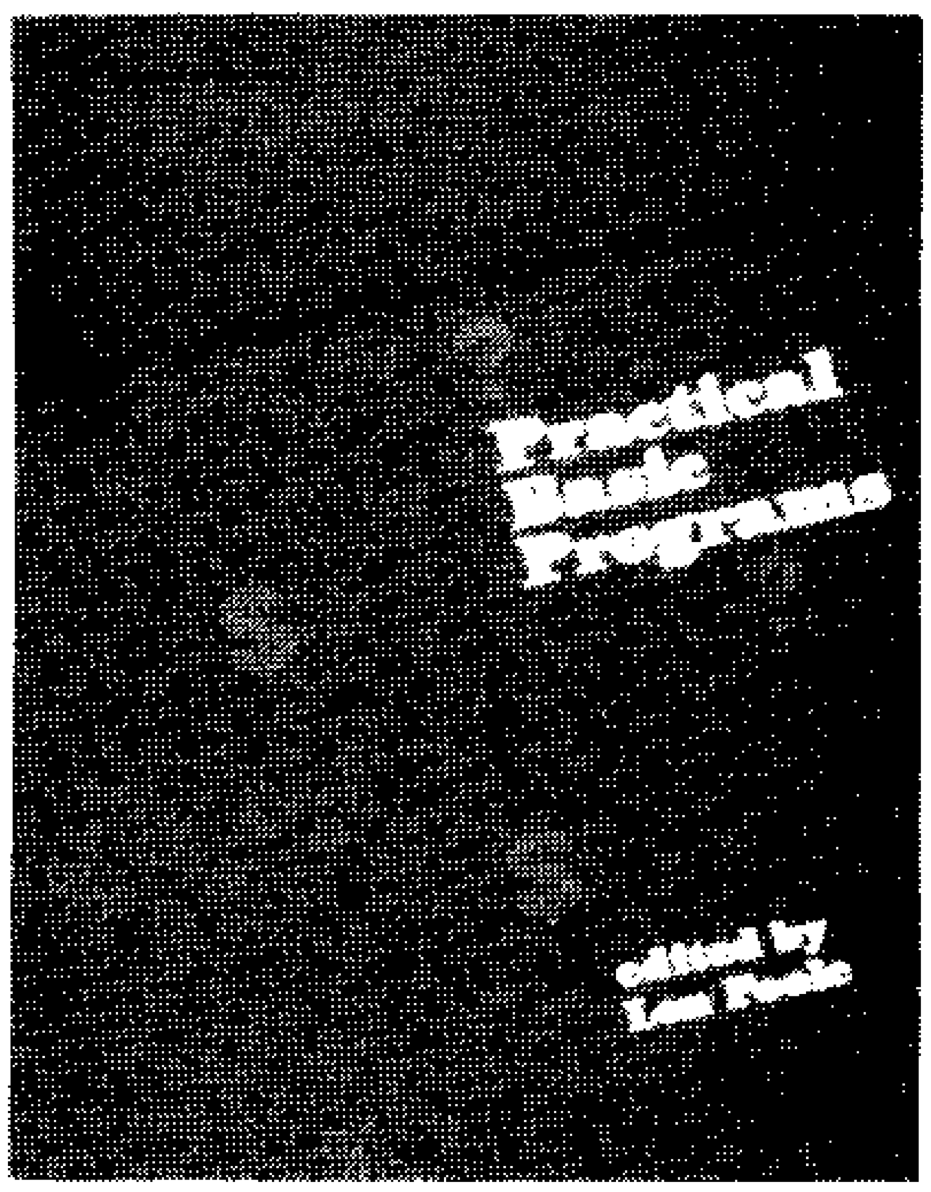
BASIC COMPUTER PROGRAMS FOR BUSINESS: VOL. 1

By Charles D. Sternberg.

A must for small businesses utilizing micros as well as for entrepreneurs, these two volumes provide a wealth of practical business applications. Each program is documented with a description of its functions and operation, a listing in BASIC, a symbol table, sample data, and one or more samples.

Volume 1 contains over 35 programs covering: budgets, depreciation, cash flow, property comparisons, accounts payable, order entry, warehouse locations, inventory turnover analysis, job routing, resource allocation, production scheduling, etc.

volume 1, paper, **$10.95 (t)**
1980, 384 pages, 7 x 9¼



PRACTICAL BASIC PROGRAMS

Edited by Lon Poole

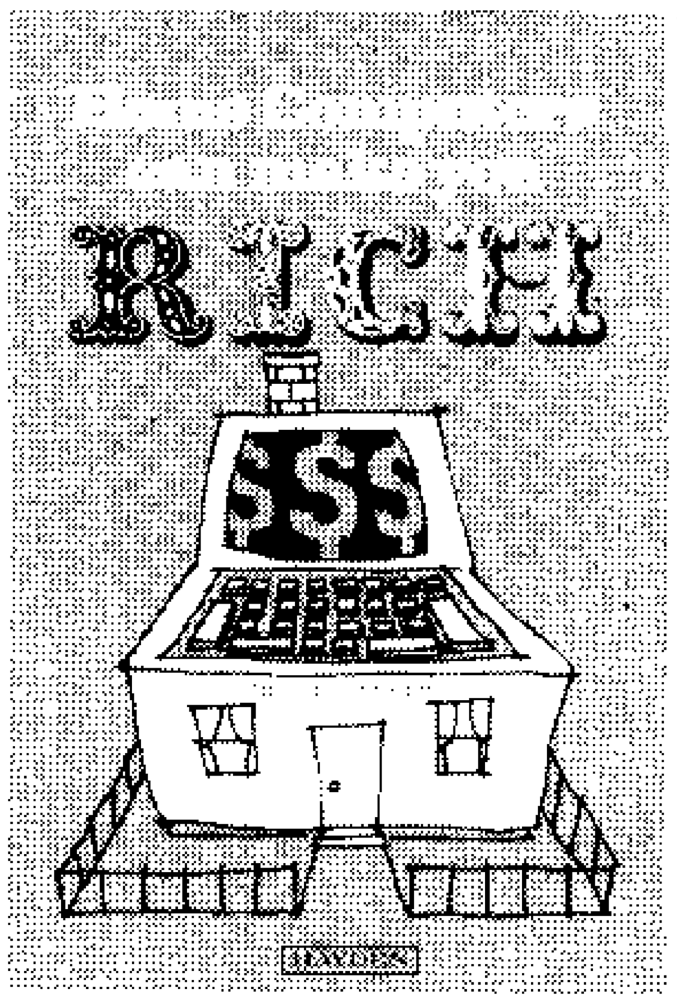
Here is a new collection of 40 programs you can easily key in and use on most microcomputers. Each program does something useful. *Practical BASIC Programs* is especially useful in small business applications. It solves problems in finance, management decision, mathematics and statistics. It requires no prior programming knowledge. Each program is thoroughly documented. The book contains sample runs, practical problems, BASIC source listings, and an easy to follow narrative to help you realize the potential uses of each program. This book is a valuable reference for anyone who needs a wide range of useful programs: income averaging, present value of a tax shield, lease/buy decision, financial statement ratio analysis, checkbook reconciliation, home budgeting, nonlinear breakeven analysis, Program Evaluation and Review Technique (PERT), statistics, data forecasting divergence, musical transposition, Bayesian decision analysis, etc. **paper $15.99**

1980, 200 pages, 8½ x 11

Use the order card in the back of this magazine, or itemize your order on a separate piece of paper and mail to: 99'er Magazine/Book Dept., 2715 Terrace View Drive, Eugene, OR 97405. Be sure to include check or detailed credit card information. No. C.O.D. orders accepted. Add $1.50 postage & handling for 1 book, $2.00 for 2 books, or $2.50 for 3 or more books. Please allow 4-6 weeks for delivery. If there is a question regarding your order please write to Customer Service at the above address.

PRICES SUBJECT TO CHANGE WITHOUT NOTICE.

99'er BOOKSTORE



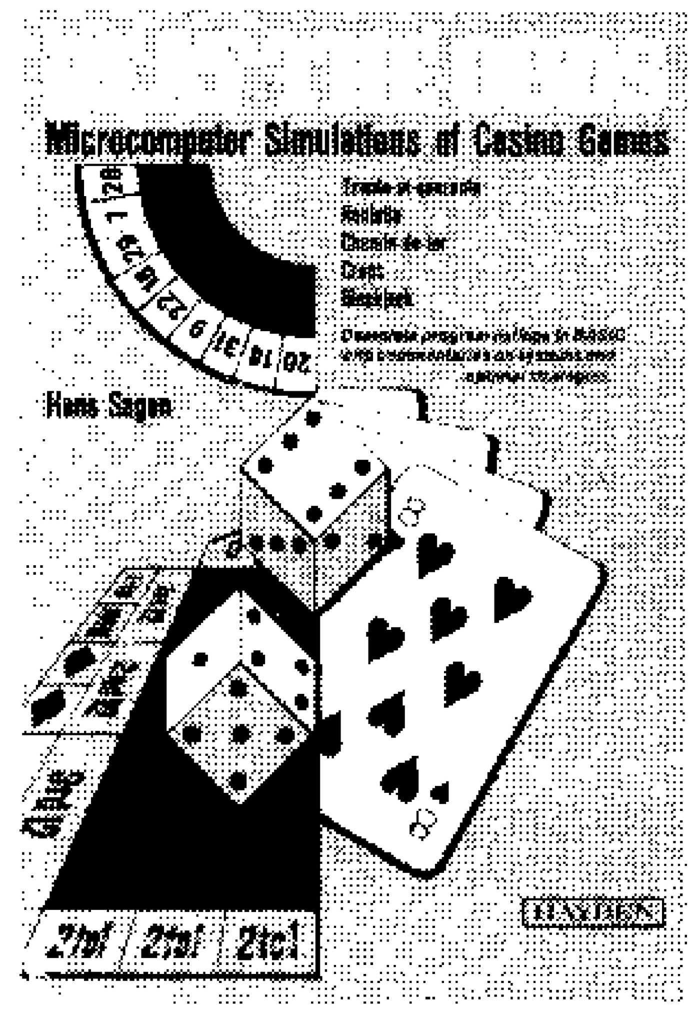
HOME COMPUTERS CAN MAKE YOU RICH

By Joe Weisbecker.

Here's a valuable text for every home computer owner and nonowner interested in spare-time income opportunities. You'll be introduced to the microcomputer industry, and the types of people involved in it. You'll find out how to learn more about this new industry. Discussed are basic principles of making money, freelance writing, programming, consulting, inventing, computer-made products, investing, and much more. The goal of this book is to stimulate computer designers and microcomputer companies to direct more effort to the home computer market.

CONTENTS: The Microcomputer Industry. What You Need to Know About Making Money. Resources You Can Use. Choosing Your Hardware. Writing for Money. Creating and Selling Programs. Services for Sale. Use Your Imagination. Invent Your Way to Success. Making Your Money Grow. Working at Home.

5177-8, paper, **$6.50**
1980, 128 pages, 6 x 9



BEAT THE ODDS: MICROCOMPUTER SIMULATIONS OF CASINO GAMES

By Hans Sagan.

Here's an extremely useful programming guide that provides realistic simulations of five popular Casino games: Trente-et-Quarante (Thirty and Forty), Roulette, Chemin-de-Fer, Craps, and Blackjack. Each of the five chapters has the same structure. It begins with a computer run, displaying facets of the programs, followed by an explanation of the objectives and the physical execution of the game. Acceptable bets and how to place them are discussed and systems and/or strategies laid out. Finally, the computer program is developed and various modifications of the program are detailed.

All programs are written in BASIC and heavily REM'd for readability and conversion. A comprehensive bibliography, a glossary of French gambling terms and phrases, and hints on the discrepancies between BASIC dialects are included, as well as a summary of maxims of probability theory.

5181-6, paper, **$7.95(t)**
1980, 128 pages, 6 x 9

MINDSTORMS: CHILDREN, COMPUTERS AND POWERFUL IDEAS

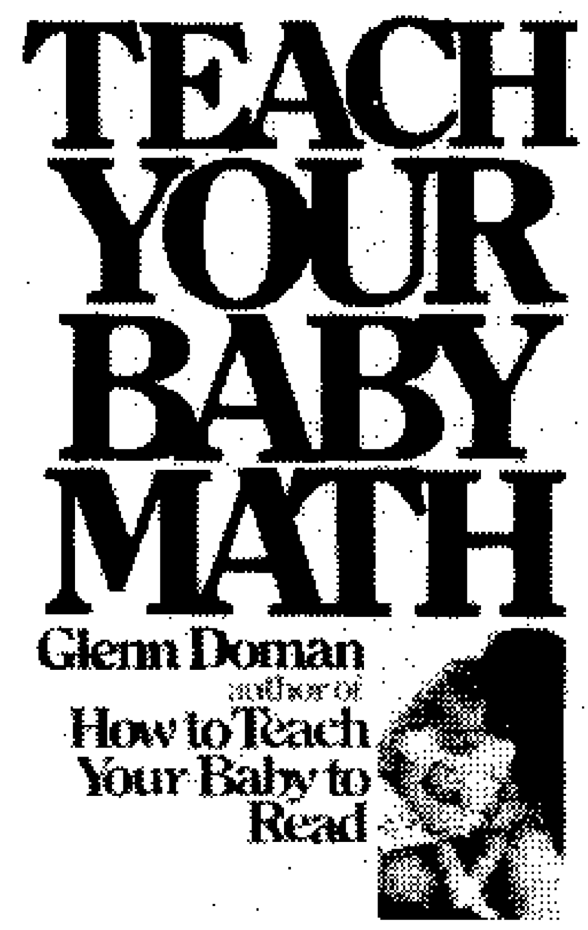
By Seymour Papert

The definitive work on the philosophy behind LOGO. Excerpted in the May/June issue of this magazine. **hardcover, $12.95**
1980, 230 pages, 6 x 9

TEACH YOUR BABY MATH

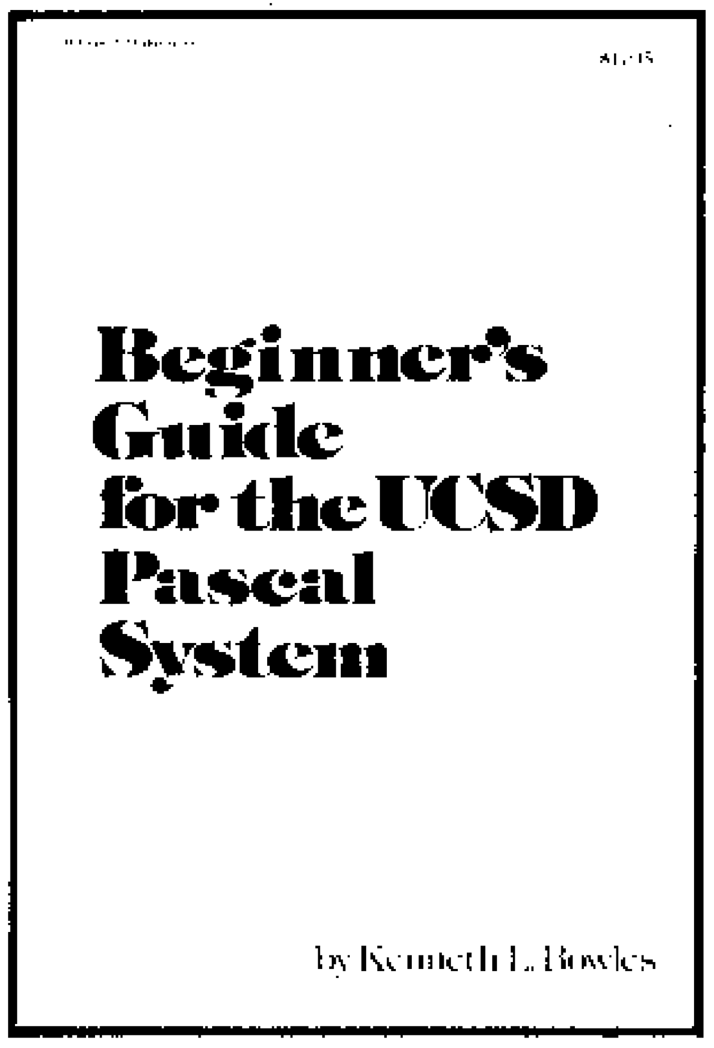
By Glenn Doman

The book upon which the *Tiny Math I* program (in the May/June issue of this magazine) is based. **hardcover, $8.95**
1969, 110 pages, 6 x 9



Use the order card in the back of this magazine, or itemize your order on a separate piece of paper and mail to: 99'er Magazine/Book Dept., 2715 Terrace View Drive, Eugene, OR 97405. Be sure to include check or detailed credit card information. No. C.O.D. orders accepted. Add $1.50 postage & handling for 1 book, $2.00 for 2 books, or $2.50 for 3 or more books. Please allow 4-6 weeks for delivery. If there is a question regarding your order please write to Customer Service at the above address. PRICES SUBJECT TO CHANGE WITHOUT NOTICE.

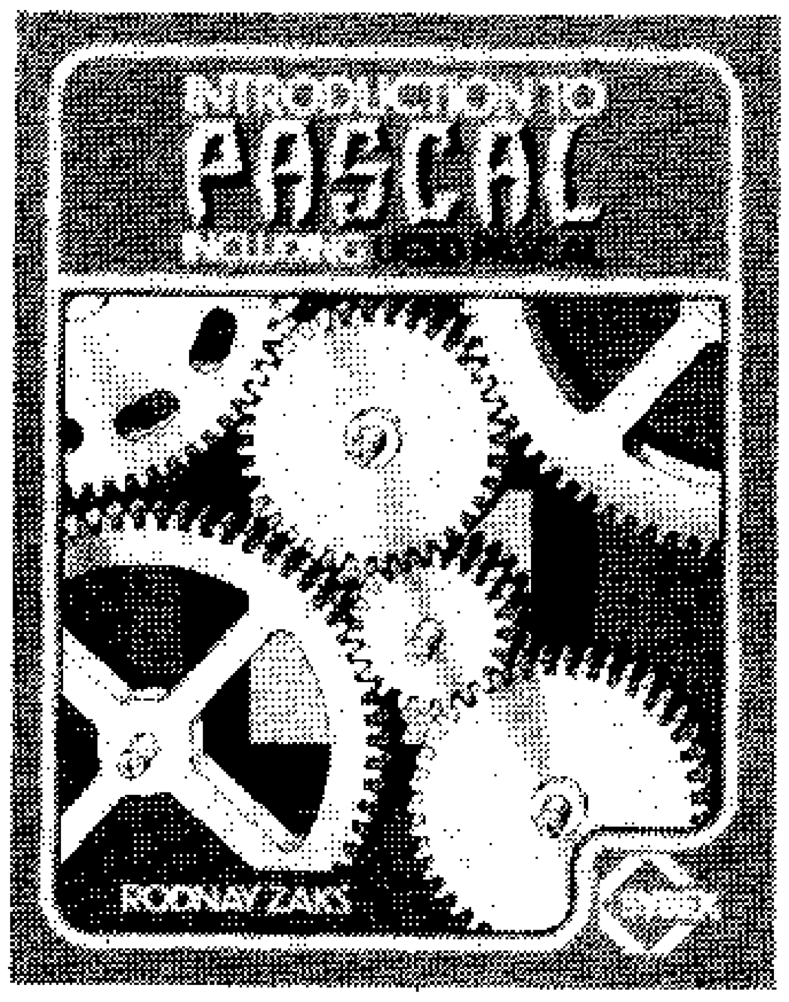
99'er BOOKSTORE



BEGINNER'S GUIDE FOR THE UCSD PASCAL SYSTEM

By Kenneth Bowles

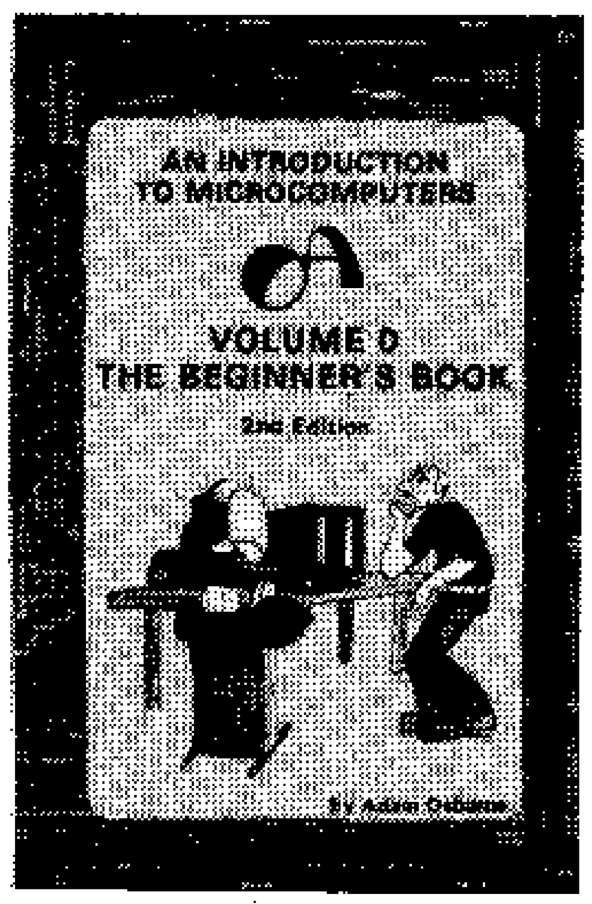
This highly informative book is written by the originator of the UCSD Pascal System. It is designed as an orientation guide for learning to use the UCSD Pascal System, and features tutorial examples of programming tasks in the form of self-study quiz programs. Once familiar with the system you will find the guide an invaluable reference tool for creating advanced applications. paper $11.95
1980, 204 pages, 6 x 9

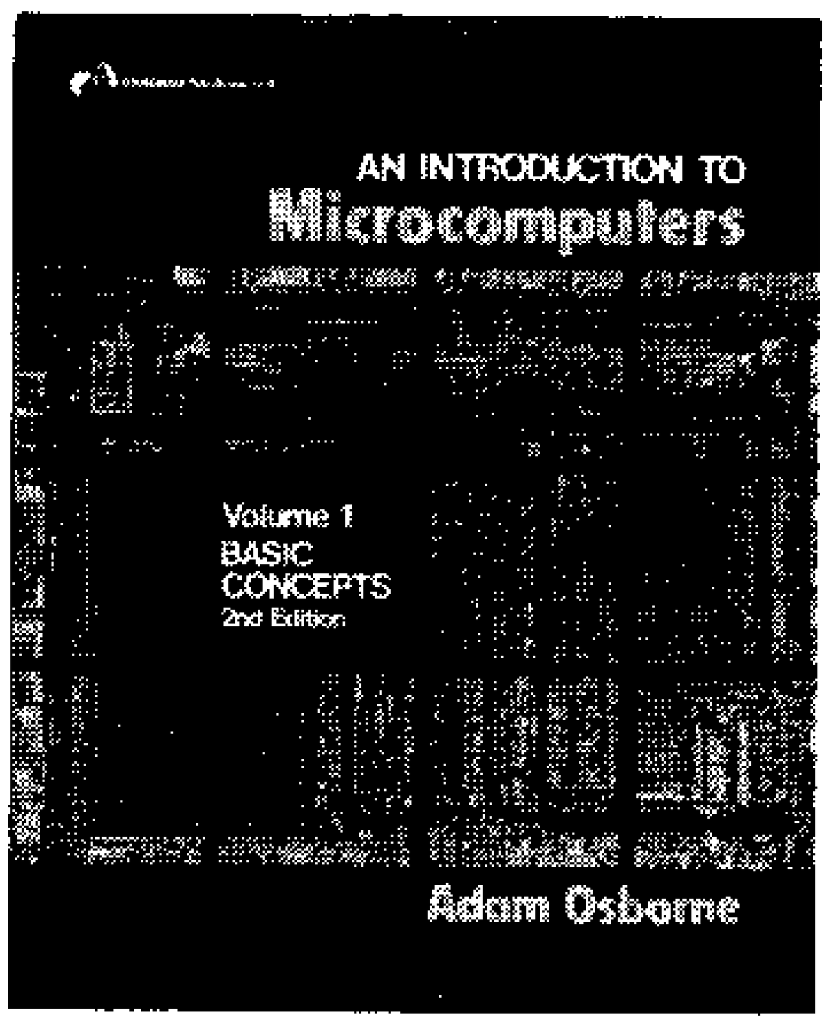


INTRODUCTION TO PASCAL (INCLUDING UCSD PASCAL)

By Rodnay Zaks

This is the first book on Pascal that can be used by persons who have never programmed before, but more generally it is a simple and comprehensive introduction to standard and UCSD Pascal for anyone—beginner to experienced programmer--who wants to learn the language rapidly. The logical progression and graduated exercises—designed to provide practice as well as test skill and comprehension—enable the reader to begin writing simple programs almost immediately. This book presents all concepts and techniques in a clear and simple style, making it accessible to beginners and useful to experienced programmers. All Pascal features are covered in detail, from basic definitions to complex data structures. An extensive appendix section presents a listing of all symbols, keywoods and rules of syntax for programming in Pascal, providing a concise summary and important reference tool. paper $14.95
1981, 440 pages, 7 x 9





AN INTRODUCTION TO MICROCOMPUTERS — VOLUME O: THE BEGINNER'S BOOK

By Adam Osborne

Here's the book to start with if you know nothing about microcomputers but wish to learn about them. With the help of numerous illustrations and a wonderfully lighthearted text, *Volume O* will help give you a sound understanding of the basics of microcomputing. You'll learn about the microcomputer's construction, terminology, internal logic, and application. If you have plans to program microcomputers, or if you must make decisions related to microcomputer applications, *The Beginner's Book* will provide the terminology and general concepts you'll need. This volume also provides an excellent background for the beginner wanting to go on to *An Introduction to Microcomputers: Volume 1 — Basic Concepts.* paper, $7.95
1979, 240 pages, 5 x 8

AN INTRODUCTION TO MICROCOMPUTERS — VOLUME 1 — BASIC CONCEPTS

By Adam Osborne

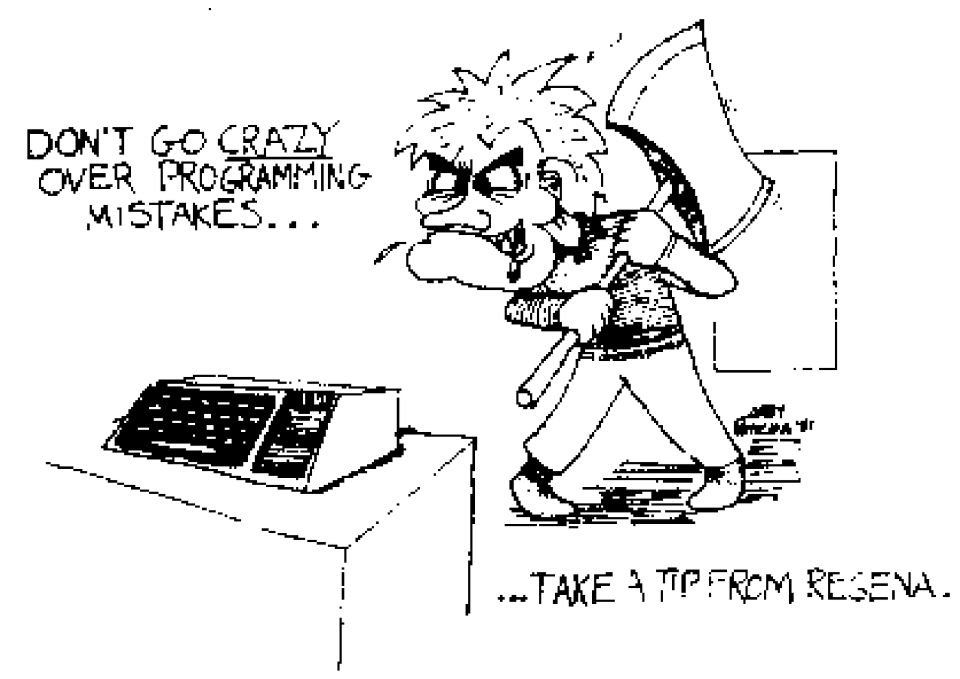
Using concepts that are common to all microprocessor systems, *Volume 1* develops a detailed picture of what a microcomputer can do, how it does what it does, and how the capabilities of microcomputers can best be applied. *Basic Concepts* presents the fundamental logic framework upon which microcomputer systems are built, so that the reader can evaluate the applicability of microcomputers to any practical problem. This new revised edition incorporates all recent microprocessor developments. Concepts are discussed in terms of modern hardware configurations, and examples of common applications are drawn from today's most popular devices. For example, the logic instructions and programming concepts of the new 16-bit microprocessors are discussed in detail, and current logic distribution configurations are used throughout the text with numerous illustrations and examples. Programming mnemonics conform to the newly proposed IEEE standard. This is the first book in print to use them. paper, $12.99
1980, 320 pages, 7 x 9

Home Secretary . . . from p. 47

```
820 CALL SOUND(200,800,4)
830 PRINT :"** WARNING ** MEMORY GETTING
    FULL":" (LSIZE=";STR$(LSIZE)&"/3800)":
840 RETURN
850 SC=4
860 GOSUB 3060
870 PRINT "PRESS":::"1 - TO ADD MORE
    DATA"::"2 - TO ALTER THE DATA"
880 PRINT :"3 - TO DISPLAY THE
    DIRECTORY"::"4 - TO DISPLAY ONE ENTRY":
890 PRINT :"5 - TO USE THE DATA"::"6 - TO
    STORE DATA FILE"::"7 - FOR PRINTER LISTING"::
900 PRINT "8 - TO END PROGRAM"
910 IF FLAG1=0 THEN 930
920 PRINT "  *** UPDATE DIRECTORY ***"
930 GOSUB 3120
940 IF KEY<49 THEN 850
950 IF KEY>56 THEN 850
960 GOSUB 3060
970 ON (KEY-48)GOSUB 990,1010,1430,1560,3150,
    2140,4280,4450
980 RETURN
990 GOSUB 430
1000 RETURN
1010 REM  DATA ALTERATION
1020 INPUT "WHICH ONE? ":M$
1030 IF M$=""THEN 1410
1040 PRINT ::"  ENTER"::"     NEW DATA AT
     CURSOR"::"      'D' TO DELETE THE ITEM"
     ::"      'ENTER' FOR NO CHANGES":
1050 GOSUB 1800
1060 PRINT ::
1070 FLAG1=1
1080 TI$="?   "
1090 I=M
1100 GOSUB 770
1110 T=-T
1120 GOSUB 800
1130 INPUT A1$(M)&TI$:TMP$
1140 IF TMP$=""THEN 1220
1150 IF TMP$<>"D"THEN 1200
1160 A1$(M)="__"
1170 GOSUB 2410
1180 N=N-1
1190 RETURN
1200 A1$(M)=TMP$
1210 FLAG2=1
1220 INPUT A2$(M)&TI$:TMP$
1230 IF TMP$=""THEN 1250
1240 A2$(M)=TMP$
1250 INPUT A3$(M)&TI$:TMP$
1260 IF TMP$=""THEN 1280
1270 A3$(M)=TMP$
1280 INPUT A4$(M)&TI$:TMP$
1290 IF TMP$=""THEN 1310
1300 A4$(M)=TMP$
1310 INPUT A5$(M)&TI$:TMP$
1320 IF TMP$=""THEN 1340
1330 A5$(M)=TMP$
1340 GOSUB 770
1350 IF T<192 THEN 1380
1360 PRINT :"** REENTER LAST SET **"
1370 GOSUB 1130
1380 GOSUB 800
1390 GOSUB 1650
1400 ON T GOTO 1130,1020,1410
1410 RETURN
1420 REM  DISPLAY ENTIRE DIRECTORY
1430 IF FLAG2=0 THEN 1450
1440 GOSUB 2410
1450 FOR I=1 TO N
1460 M=29-LEN(A2$(I))
1470 T$=STR$(I)&"."
1480 PRINT TAB(4-LEN(T$));T$;A1$(I);TAB(M);A2$(I)
1490 IF I=20 THEN 1520
1500 IF I=40 THEN 1520
1510 GOTO 1530
1520 GOSUB 3100
1530 NEXT I
1540 GOSUB 3100
1550 RETURN
1560 REM  SINGLE ITEM LISTING
1570 INPUT "WHICH ONE?  ":M$
1580 IF M$=""THEN 1640
1590 GOSUB 1800
1600 GOSUB 3060
1610 PRINT A1$(M)::A2$(M)::A3$(M)::A4$(M)::A5$(M)::
1620 GOSUB 1650
1630 ON T GOTO 1600,1570,1640
1640 RETURN
1650 PRINT :"PRESS":"  E TO LIST UP":"  X TO
     LIST DOWN":"  S TO SEARCH MORE"
1660 GOSUB 3100
1670 T=3
1680 IF KEY<>69 THEN 1720
1690 T=1
1700 IF M=1 THEN 1780
1710 M=M-1
1720 IF KEY<>88 THEN 1760
1730 T=1
1740 IF M=N THEN 1780
1750 M=M+1
1760 IF KEY<>83 THEN 1780
1770 T=2
1780 RETURN
1790 REM   SEARCH ROUTINE FOR SINGLE ITEM LISTING
1800 IF ABS(ASC(M$)-53)>4 THEN 1850
1810 M=VAL(M$)
1820 IF M<=N THEN 1840
1830 M=N
1840 RETURN
1850 FOR I=1 TO N
1860 M=I
1870 IF M$<=A1$(I)THEN 1890
1880 NEXT I
1890 RETURN
1900 REM  LOAD DATA
1910 PRINT "ENTER"::"1. CS1"::"2. DSK1"::"3. OTHER"
1920 INPUT DEV
1930 IF DEV<>1 THEN 1960
1940 DEV$="CS1"
1950 GOTO 2010
1960 IF DEV<>2 THEN 2000
1970 INPUT "ENTER FILE NAME:":FIL$
1980 DEV$="DSK1."&FIL$
1990 GOTO 2010
2000 INPUT "ENTER DEVICE NAME:":DEV$
2010 OPEN #2:DEV$,INTERNAL,INPUT ,FIXED 192
2020 INPUT #2:OPT,N,FIL$,DATE$,LSIZE
2030 IF OPT=1 THEN 2050
2040 READ CAT$(1),CAT$(2),CAT$(3),CAT$(4),CAT$(5)
2050 PRINT ::FIL$::"LSIZE(3800)=";LSIZE::"LAST
     UPDATE: ";DATE$::
2060 FOR I=1 TO N
2070 INPUT #2:A1$(I),A2$(I),A3$(I),A4$(I),A5$(I)
2080 NEXT I
2090 IF DEV=1 THEN 2120
2100 FOR TD=1 TO 1000
2110 NEXT TD
2120 CLOSE #2
2130 RETURN
2140 REM   SAVE DIRECTORY
2150 IF FLAG2=0 THEN 2170
2160 GOSUB 2410
2170 PRINT "ENTER 1. CS1"
2180 PRINT "      2. DSK1"
2190 PRINT "      3. OTHER"::::
2200 INPUT "YOUR CHOICE?":ANS
2210 IF (ANS<1)+(ANS>3)THEN 2170
2220 ON ANS GOTO 2230,2250,2310
2230 DEV$="CS1"
2240 GOTO 2320
2250 INPUT "ENTER FILE NAME.":NAM$
2260 IF LEN(NAM$)<11 THEN 2290
2270 PRINT "ENTER NO MORE THAN TEN      LETTERS
     PLEASE."
2280 GOTO 2170
2290 DEV$="DSK1."&NAM$
2300 GOTO 2320
2310 INPUT "ENTER DEVICE NAME.":DEV$
2320 INPUT "ENTER DATE.":DATE$
2330 OPEN #3:DEV$,INTERNAL,OUTPUT,FIXED 192
2340 PRINT #3:OPT,N,FIL$,DATE$,LSIZE
2350 FOR I=1 TO N
2360 PRINT #3:A1$(I),A2$(I),A3$(I),A4$(I),A5$(I)
2370 NEXT I
2380 CLOSE #3
2390 FLAG1=0
2400 RETURN
2410 REM  SORTING ROUTINE
2420 FLAG2=0
2430 CALL SOUND(100,800,6)
2440 PRINT :::" ***** SORTING DATA *****":::
2450 IF (N-1)THEN 2470
2460 RETURN
2470 FOR I=1 TO N
2480 NEXT I
2490 N2=INT(N/2)
2500 N21=N2+2
2510 ICT=1
2520 I=2
```

Home Secretary . . .

```
2530 N1=N21-I
2540 NN=N
2550 IK=N1
2560 GOSUB 2880
2570 JK=2*IK
2580 IF JK>NN THEN 2660
2590 IF JK=NN THEN 2620
2600 IF A1$(JK+1)<=A1$(JK)THEN 2620
2610 JK=JK+1
2620 IF A1$(JK)<=C$ THEN 2660
2630 GOSUB 2940
2640 IK=JK
2650 GOTO 2570
2660 GOSUB 3000
2670 ON ICT GOTO 2680,2780
2680 IF I>=N2 THEN 2710
2690 I=I+1
2700 GOTO 2530
2710 ICT=2
2720 NP2=N+2
2730 I=2
2740 N1=NP2-I
2750 NN=N1
2760 IK=1
2770 GOTO 2560
2780 IK=1
2790 GOSUB 2880
2800 JK=N1
2810 GOSUB 2940
2820 IK=N1
2830 GOSUB 3000
2840 IF I>=N THEN 2870
2850 I=I+1
2860 GOTO 2740
2870 RETURN
2880 C$=A1$(IK)
2890 M$=A2$(IK)
2900 T$=A3$(IK)
2910 TMP$=A4$(IK)
2920 TI$=A5$(IK)
2930 RETURN
2940 A1$(IK)=A1$(JK)
2950 A2$(IK)=A2$(JK)
2960 A3$(IK)=A3$(JK)
2970 A4$(IK)=A4$(JK)
2980 A5$(IK)=A5$(JK)
2990 RETURN
3000 A1$(IK)=C$
3010 A2$(IK)=M$
3020 A3$(IK)=T$
3030 A4$(IK)=TMP$
3040 A5$(IK)=TI$
3050 RETURN
3060 REM   CLEAR & SET SCREEN
3070 CALL CLEAR
3080 CALL SCREEN(SC)
3090 RETURN
3100 REM  KEY RETURN
3110 PRINT ::"PRESS ANY KEY TO
     CONTINUE"
3120 CALL KEY(0,KEY,STATUS)
3130 IF STATUS=0 THEN 3120
3140 RETURN
3150 REM  UTILITY PROGRAMS
3160 IF OPT=1 THEN 3290
3170 SUM=0
3180 FOR I=1 TO N
3190 SUM=SUM+VAL(A2$(I))
3200 NEXT I
3210 PRINT "TOTAL COST OF ALL THE
     ITEMS":::
3220 PRINT ::::TAB(10);SUM::::::::
3230 CALL HCHAR(11,7,36,18)
3240 CALL HCHAR(19,7,36,18)
3250 CALL VCHAR(12,7,36,7)
3260 CALL VCHAR(12,24,36,7)
3270 GOSUB 3100
3280 RETURN
3290 REM  DIAL PHONE
3300 REM  CLOCK TIME DELAYS FOLLOW
3310 RESTORE 3320
3320 DATA 2,57,55,53,51
3330 READ D1,D2,D3,D4,D5
3340 PRINT ::"YOU WANT ME TO DIAL"::
3350 GOSUB 1570
3360 IF M$=""THEN 3610
3370 CALL CLEAR
3380 H=23
3390 V=8
3400 T$=A2$(M)
3410 L=LEN(T$)
3420 PRINT A1$(M):A2$(M):A3$(M):
     A4$(M):A5$(M)::::
3430 IF L<10 THEN 3460
3440 L=L+1
3450 T$="1"&T$
3460 FOR J=1 TO L
3470 TMP$=SEG$(T$,J,1)
3480 CALL HCHAR(H,V,ASC(TMP$),1)
3490 V=V+1
3500 IF ASC(TMP$)<48 THEN 3600
3510 IF ASC(TMP$)>57 THEN 3600
3520 T=VAL(TMP$)
3530 IF T<>0 THEN 3560
3540 CALL SOUND(300,941,0,1336,2)
3550 GOTO 3590
3560 I=INT((T-1)/3)+1
3570 IJ=T-3*(I-1)
3580 CALL SOUND(300,P1(I),0,P2(IJ),2)
3590 CALL SOUND(250,44000,29)
3600 NEXT J
3610 PRINT ::"PRESS ":"       R TO
     REDIAL":"       S TO START
     STOPWATCH":"      N FOR NEW
     NUMBER"
3620 GOSUB 3100
3630 IF KEY=82 THEN 3370
3640 IF KEY=78 THEN 3350
3650 IF KEY<>83 THEN 3690
3660 PRINT :"HOLD DOWN ":"R TO DIAL
     AGAIN":"ANY KEY TO CONTINUE":::
3670 GOSUB 3700
3680 IF T=82 THEN 3290
3690 RETURN
3700 REM  STOP WATCH
3710 H=23
3720 V=23
3730 JS=1
3740 S=4
3750 DELAY=D1
3760 FOR J1=0 TO 15
3770 A1=48+J1
3780 FOR J2=0 TO 9
3790 A2=48+J2
3800 FOR J3=0 TO 5
3810 A3=48+J3
3820 FOR J4=JS TO 9
3830 A4=48+J4
3840 GOSUB 4020
3850 IF STATUS<>0 THEN 4010
3860 S=4
3870 DELAY=D2
3880 NEXT J4
3890 JS=0
3900 DELAY=D3
3910 NEXT J3
3920 DELAY=D4
3930 S=11
3940 NEXT J2
3950 A2=48
3960 DELAY=D5
3970 S=10
3980 GOSUB 4020
3990 NEXT J1
4000 GOTO 3760
4010 RETURN
4020 REM  COMPENSATED CLOCK
4030 CALL HCHAR(H,V+2,32,1)
4040 CALL KEY(0,T,STATUS)
4050 FOR J=1 TO DELAY
4060 IF STATUS<>0 THEN 4010
4070 NEXT J
4080 CALL SCREEN(S)
4090 CALL HCHAR(H,V,A1,1)
4100 CALL HCHAR(H,(V+1),A2,1)
4110 CALL HCHAR(H,V+2,58,1)
4120 CALL HCHAR(H,V+3,A3,1)
4130 CALL HCHAR(H,V+4,A4,1)
4140 RETURN
4150 REM  SCREEN PRINTING
4160 IF RP=0 THEN 4270
4170 OPEN #1:"RS232"
4180 FOR IJ=1 TO 20
4190 PRINT #1:T$
4200 T$=""
4210 FOR J=1 TO 32
4220 CALL GCHAR(IJ,J,CH)
4230 T$=T$&CHR$(CH)
4240 NEXT J
4250 NEXT IJ
4260 CLOSE #1
4270 RETURN
4280 REM  COMPLETE LISTING ON PRINTER
4290 INPUT "ENTER DEVICE NAME.":DEV$
4300 PRINT "IS ";DEV$;" READY?(Y/N)."
4310 GOSUB 3120
4320 IF KEY<>89 THEN 4440
4330 OPEN #3:DEV$,OUTPUT
4340 T=INT(N/9)
4350 FOR J=0 TO T
4360 FOR M=1 TO 9
4370 I=9*J+M
4380 PRINT #3:TAB(5);I;")";TAB(10);
     A1$(I):TAB(10);A2$(I):TAB(10);
     A3$(I):TAB(10);A4$(I):TAB(10);
     A5$(I)::
4390 IF I=N THEN 4430
4400 NEXT M
4410 GOSUB 3100
4420 NEXT J
4430 CLOSE #3
4440 RETURN
4450 PRINT "DO YOU WISH TO HALT
     THE      PROGRAM AND LOSE ALL
     DATA INMEMORY? (Y/N)."
4460 CALL KEY(0,K,S)
4470 IF S=0 THEN 4460
4480 IF K=ASC("N")THEN 850
4490 IF K<>ASC("Y")THEN 4440
4500 END
```



DEALER DIRECTORY

Goleta, CA

See our complete line of personal and business computers including: Texas Instruments, Atari, Apple, Altos, Data General. Also All accessories, software and supplies. Plus all that's new in personal electronics. **Personal Electronics, 5674 Calle Real Center, Goleta, CA 93117, (805) 967-5322.**

Los Angeles, CA

A software/systems house primarily engaged in the sale of software for TI and Polymorphic computers. Hardware sales are usually included in a combined turnkey package. Prospective customers are seen by prior telephone appointment only. **AAAA Discount Computer How's, 13022 Psomas Way, Los Angeles, CA 90066, (213) 391-8777.**

Waterloo, IA

Texas Instruments Large selection of Accessories and Software for 99/4 at Dhein's True Value, 7 W. Airline Hwy, Waterloo, IA 50701. Sale prices at store are also given on mail orders. Write for price list. **Dhein's True Value, 7 W. Airline Highway, Waterloo, IA 50701, (319) 232-6225.**

Evansville, IN

Independent Register 2414 N. Governor (across from Town Center Mall) is the tri-state location for TI-99/4 Hardware/Software, Supplies etc. We supply the tri-state with popular magazines and software for other units. Open Mon - Sat. 8-5. **Independent Register, 2414 N. Governor, Evansville, IN 47711. (812) 424-8246.**

Lexington, KY

A complete line Texas Instruments dealer. We stock all Texas Instruments calculators and accessories, learning aids, home computer products, accessories, peripherals and software. Atari computers and Hewlett Packard calculators in stock. **CBM Incorporated, 198 Moore Drive, Lexington, KY 40503, (606) 276-1519.**

Marlow Heights, MD

Specializing in Texas Instruments 99/4 computer and accessories. We have a large selection of Software for the TI-99/4, TRS-80, Apple, and Atari. We also stock Lobo Drives and Epson Printers. **Wills Computer Store, 4410 Stamp Road, Marlow Heights, MD 20031, (301) 423-4525.**

Minneapolis, MN

Authorized Texas Instruments TI-99/4 computer dealer offering you the best prices in the U.S.A. on the computer and all peripherals. Also, a wide variety of printers. CRT's available at lowest prices. No collect calls, please. **Calculators, Inc., 7409 Fremont Avenue South, Minneapolis, MN 55423, (612) 866-8908.**

Great Falls, MT

We've got the best prices because we want your business. Complete line of TI products and Epson printers. Monthly SALES on all items. Call now or write: **IRCA Inc., N.E. Bypass Hwy. 87, P. O. Box 6125, Great Falls, MT 59406, (406) 727-8835.**

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**

Salem, OR

J. Harvey's Video Clubhouse VCR's, Tapes, Home Computers TI-99/4, Atari 400, 800. **J. Harvey's Video Clubhouse 3295 Triangle Drive #142, Salem, OR 97302, (503) 581-1003.**

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.**

Call a Dealer Today

Olean, NY

Texas Instruments TI-99/4 Home Computer Systems, software, hardware, and accessories at discount prices. APF-Imagination Machine. Video Sales and Service. **BLOISE Electronics, 114 Center Street, Olean, NY 14760. (716) 372-8063.**

Rye, NY

Wide assortment of TI software, hardware and accessories all at discount prices. Also authorized Scott, Foresman dealer. Write or phone for nine-page catalog with hundreds of items. **Microcomputers Corporation, P.O.Box 191, Rye, NY 10580, (914) 967-8370.**

Electric City, WA

Texas Instruments 99/4 Home Computer Systems — Wholesale prices and not on just the main console! Also good prices on Epson printers and Televideo Terminals. For price list, write to: **Komputar Works, P. O. Box 483, Electric City, WA 99123, (509) 633-2653.**

Green Bay, WI

We specialize in the Texas Instruments 99/4 computer. We handle a complete accessory line for it, at discount prices. Custom programming for the 99/4 also done. Write for a complete price list. **The Micro House, 527 Simonet Street, Green Bay, WI 54301, (414) 432-2871.**

Dealers:
Listings are **$25** per issue; minimum insertion, 3 issues (6 months). Prepayment of **$75** 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.

Local Networking with Shared Peripherals

The TI-99/4 at North Texas State University

By Dr. Kathleen M. Swigger

Department of Computer Sciences, NT Box 13886
North Texas State University, Denton, TX 76203

The Computer Sciences Department at North Texas State University has been noted for its research and educational utilization of microcomputers. Efforts began in the early 1970s with the school's fabrication of Motorola-based systems for assembly language programming. In the fall of 1980, North Texas installed twenty-five TI-99/4 systems. These systems are currently being used for both student programming projects and faculty research.

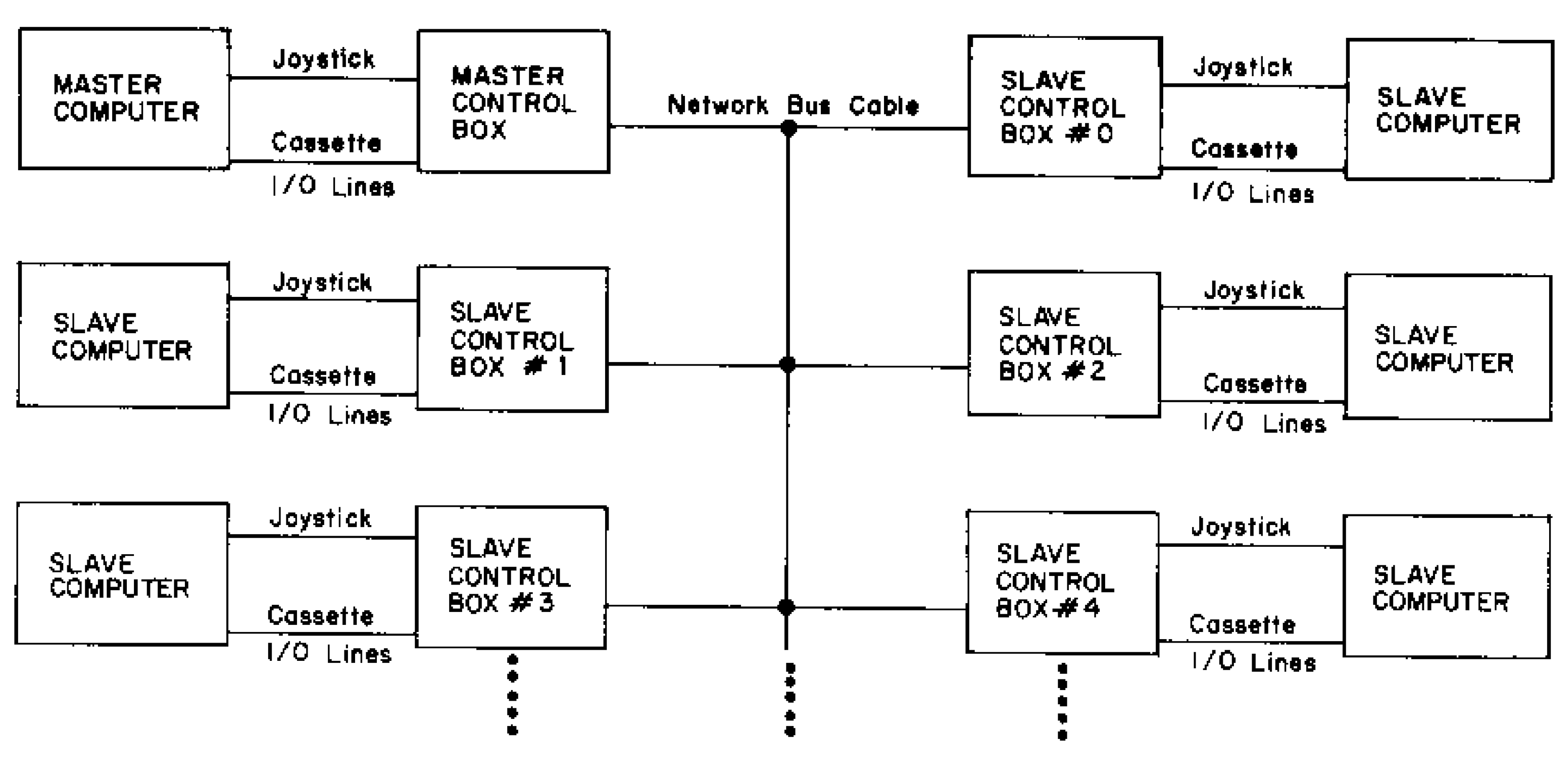
The enrollment in the Computer Sciences Department now numbers over 750 majors, making North Texas one of the largest departments of its kind in the country. By requiring all introductory students to complete their programming assignments on a TI-99/4, rather than on a larger machine, North Texas is able to handle its already large (and steadily increasing) enrollment. Making the microcomputer an integral part of the computer sciences curriculum has several educational advantages: First, students learn to use a system that is readily accessible to them at school, in the home, and in a business. The ease of using the smaller systems enable students to begin their coursework with a healthy, positive feeling about computers. Additionally, the ability to play music, use graphics and even create speech, provides students with both challenging and enjoyable programming assignments. Finally, since microcomputers are "total" computer systems, students therefore learn and more fully understand basic computer concepts such as storage, memory, input, and output.

Another successful project that NTSU has recently completed is the networking of several TI-99/4s, allowing multiple access to a single disk and printer. The system was designed mainly for educational environments that operate within limited budgets. An increasing number of educational institutions at all levels (elementary, secondary, and college) are interested in acquiring microcomputers. But these schools need to find systems that allow for disk input, provide printed output for teacher grading and analysis, and cost as little as possible. In a *school* environment, disk storage and access is not an on-line problem. Both teachers and students need to load and store programs, but they generally do not have to gain continual access to data files on disk. Therefore, a networking system like the one implemented at North Texas should solve many of the critical problems facing school officials.

The NTSU networking system includes the equipment necessary to connect up to sixteen "slave terminals" to a "master computer." The hardware for the network consists of three major parts: (1) a master control box, (2) a slave control box, and (3) a network bus cable. The master control box houses a power switch and LED hex display that tells users which terminal is being accessed, as well as the network bus cable connection to the slaves. The slave control box houses a toggle switch, two single LEDs, cable connections to the slave computers, and a cable connection to the network bus. Finally, the network bus cable consists of a cable with up to seventeen 9-pin male connectors that can be connected to the master control box. With this system, students can obtain a hard copy listing of their programs by merely sending their programs from the slave computer to the master computer. The master system, consisting of a floppy disk and printer, handles the necessary input/output.

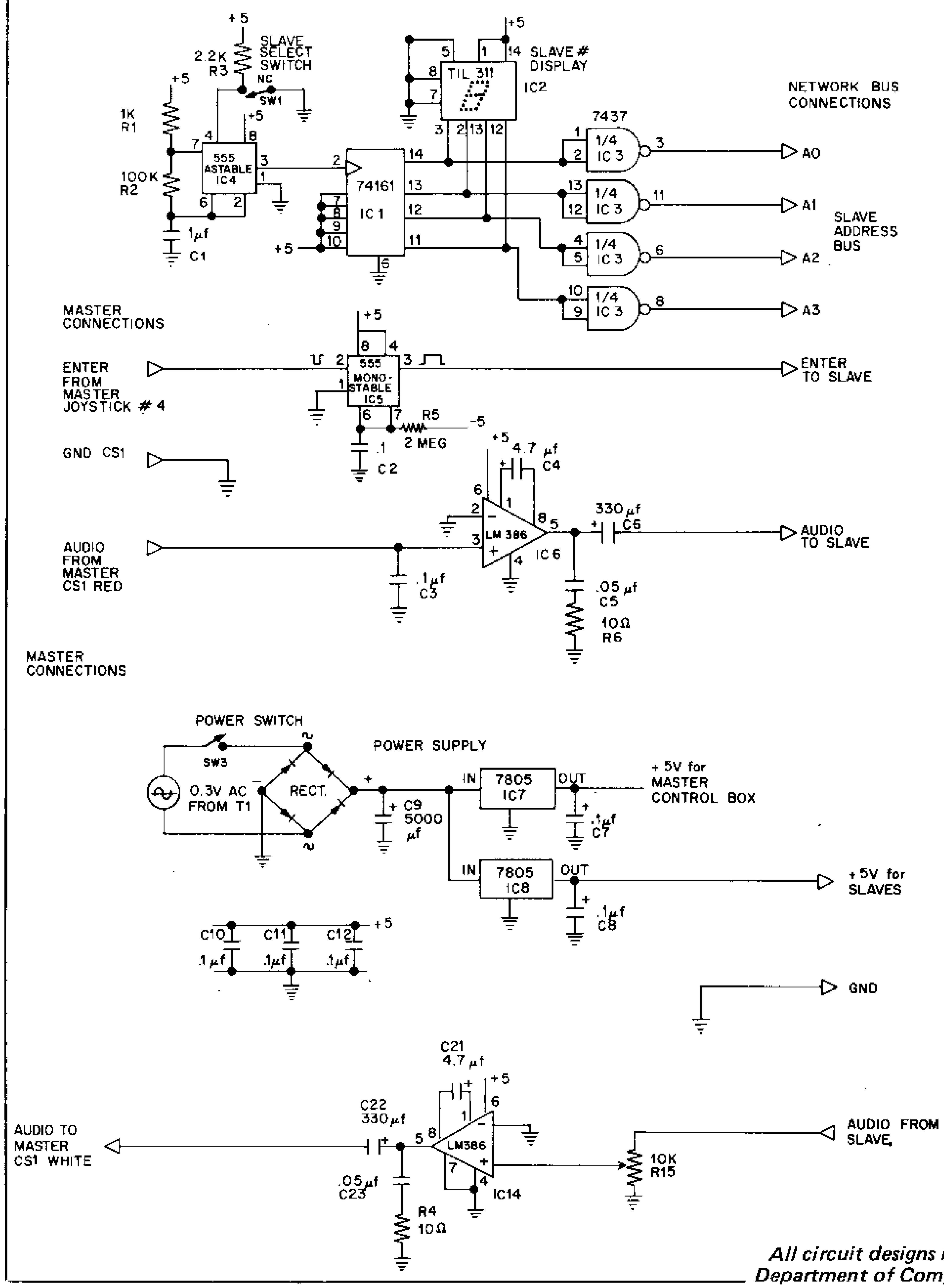
The Computer Sciences Department at NTSU has found the TI-99/4 network to be an easy-to-use educational system, and plans to continue using the small microcomputers as tools for teaching programming languages and as instruments for educational research. It has saved the department extensive duplication and the expense of purchasing additional I/O equipment.

N.T.S.U. DATANET 1 SYSTEM BLOCK DIAGRAM



See p. 82 for complete N.T.S.U. DATANET 1 PARTS LIST.

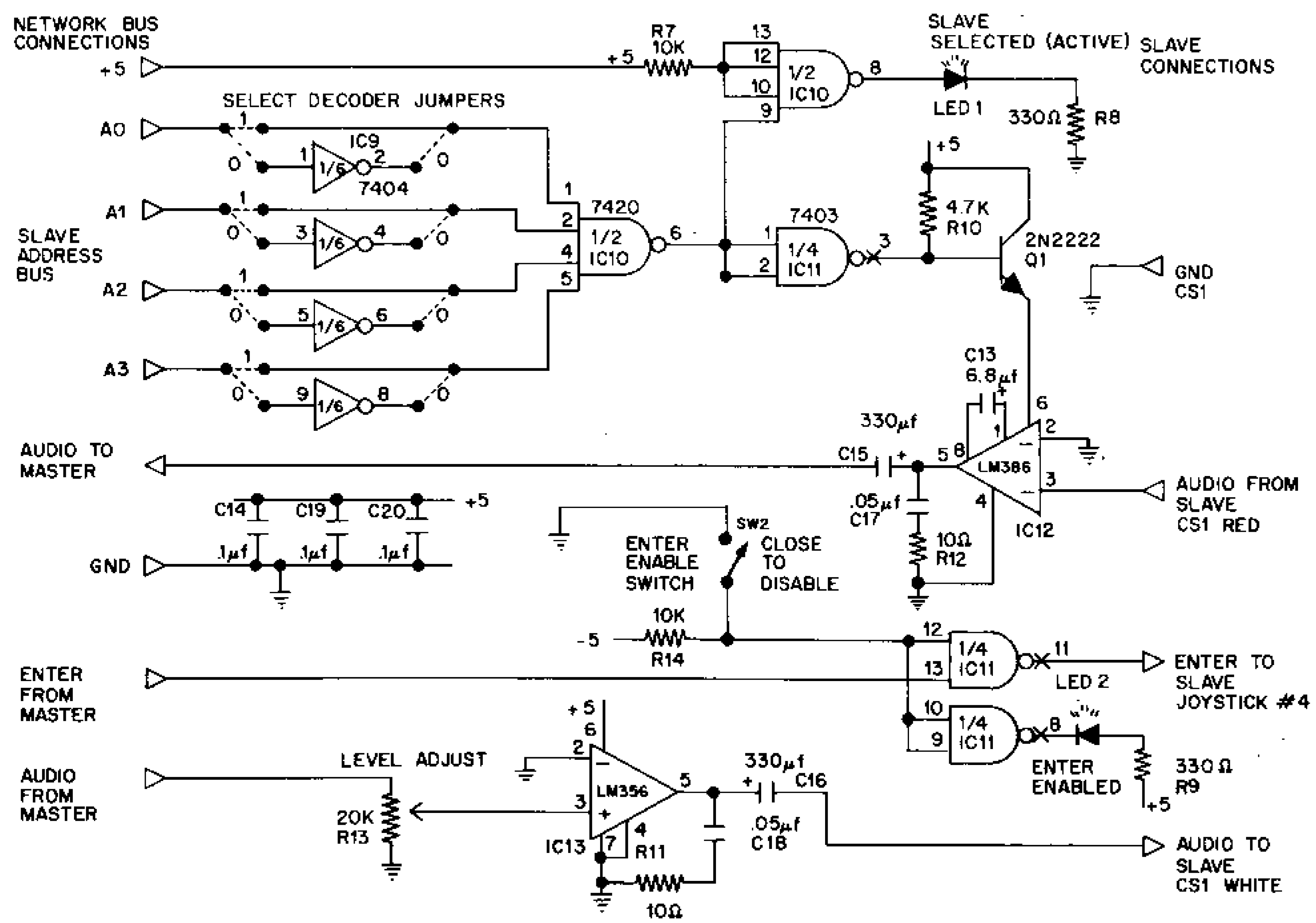
MASTER CONTROL BOX



NETWORK BUS CABLE

MASTER	PIN NUMBERS		WIRE COLOR	SLAVES
	1	+5 V	red	POWER
	6	GND	black	
	2	AO	white	SLAVE ADDRESS BUS
	3	A1	gray	
	4	A2	green	
	5	A3	blue	
	7	audio to master	orange	AUDIO CONNECTIONS
	8	audio from master	yellow	
	9	ENTER	purple	ENTER PRESSED SIGNAL

SLAVE CONTROL BOX



All circuit designs by David R. Dawson
Department of Computer Sciences, NTSU

Business . . . from p. 24

is the same under lease or purchase. You expect that after three years you would need to trade in this one on a larger model. If you buy it, the trade-in allowance will be $6000. Assuming that either depreciation or lease payments would cost a net of only 60% of the actual amounts because of an assumed 40% tax rate, the input to the program would therefore be

	(a)	(b)	(c)	(d)
1st Component	12000	0	1	1
2nd Component	–1200	12	12	3
3rd Component	–180	0	1	36
4th Component	–6000	36	1	1

If you want to check, this example gives an effective interest rate of about 14.1%. Presumably, if money costs you less than 14.1%, it would be advantageous to purchase the widget grinder instead of leasing it.

Effective Interest Rate Program: Table of Variables

Arrays:

- A1: amount of each payment in an investment component*
- T1: time at which the first payment of that component is made (in months, from current time = 0)
- F1: number of months between the payments in this component
- N1: number of payments in this component

* An investment component is a series of one or more equal payments made at fixed intervals. Payments may be paid out (+) or received (–).

Parameters:

- U9: upper limit for effective annual rate
- L9: lower limit for effective annual rate
- T9: tolerance: when the interval between upper and lower limits (L1, U1) is less than this, the program stops
- U9: tolerance: when the residual present value at a trial interest rate, divided by the sum of the absolute values of all components, is less than this, the program stops
- C: number of components
- I: index of the current investment component under consideration (always goes from 1 to C)
- L1: current lower bound on effective rate
- U1: current upper bound on effective rate
- R: trial interest rate, on which to calculate residual present value V
- V: residual present value, based on trial interest rate R
- L2: resudual present value at lower limit L1
- U2: residual present value at upper limit U1
- V3: sum of absolute values of component present values
- V4: present value of a component at rate R
- V5: temporary variable used in computing V4
- R1: monthly increase factor, using rate R

PROGRAM OUTLINE:

Line Nos.	
200-230	Set parameters
250-370	Obtain input data from user
400-560	Set lower and upper limits, and the residual present value at each
590-700	Iterate: interpolate to get a new trial interest rate R; replace either upper or lower bound by R
720-920	Subroutine: computes residual present value at the trial rate R; also computes V3
930-950	Report final result

```
100 REM  ****************************
110 REM  * EFFECTIVE INTEREST RATE *
120 REM  *           AND            *
130 REM  *  RETURN ON INVESTMENT    *
140 REM  ****************************
150 REM  99'ER VERSION 7.81.1
160 REM  BY GEORGE STRUBLE
170 REM
180 REM
190 DIM A1(10),T1(10),F1(10),N1(10)
200 U9=30.0
210 L9=0
220 T9=0.05
230 P9=1.0E-4
240 REM  ACCEPT INPUT
250 PRINT "ENTER NUMBER OF PAYMENT      COMPONENTS";
260 INPUT C
270 FOR I=1 TO C
280 PRINT
290 PRINT "ENTER AMOUNT OF PAYMENT";
300 INPUT A1(I)
310 PRINT "ENTER TIME OF FIRST OF       THESE
    PAYMENTS";
320 INPUT T1(I)
330 PRINT "ENTER PERIOD BETWEEN THESE  PAYMENTS,
    IN MONTHS";
340 INPUT F1(I)
350 PRINT "ENTER NUMBER OF THESE        PAYMENTS";
360 INPUT N1(I)
370 NEXT I
380 PRINT
390 REM  SET LOWER & UPPER BOUNDS FOR EFFECTIVE
    RATE
400 L1=L9
410 U1=U9
420 REM  GET RESIDUAL VALUE AT LOWER BOUND
430 R=L1
440 GOSUB 720
450 L2=V
460 IF ABS(V/V3)<P9 THEN 930
470 REM  GET RESIDUAL VALUE AT UPPER BOUND
480 R=U1
490 GOSUB 720
500 U2=V
510 IF ABS(V/V3)<P9 THEN 930
520 REM  RESIDUAL VALUES MUST HAVE OPPOSITE SIGNS
    AT THE BOUNDS
530 IF U2*L2<0 THEN 590
540 PRINT "EFFECTIVE RATE NOT BETWEEN ";L9;" AND ";U9
550 PRINT "CHECK YOUR INPUT OR CHANGE  BOUNDS L9
    AND U9"
560 GOTO 950
570 REM   INTERPOLATE BETWEEN LOWER & UPPER BOUNDS
580 REM  FOR NEW TRIAL RATE R
590 R=(L1*U2-U1*L2)/(U2-L2)
600 GOSUB 720
610 IF ABS(V/V3)<P9 THEN 930
620 REM  TRIAL RATE REPLACES WHICHEVER BOUND HAS
    RESIDUAL VALUE WITH THE SAME SIGN
630 IF V*L2>0 THEN 670
640 U1=R
650 U2=V
660 GOTO 690
670 L1=R
680 L2=V
690 IF U1-L1<T9 THEN 930
700 GOTO 590
710 REM   SUBROUTINE TO COMPUTE RESIDUAL VALUE V AT
    RATE R
720 V=0
730 V3=0
740 FOR I=1 TO C
750 IF N1(I)>1 THEN 790
760 REM  COMPUTE RESIDUAL VALUE IF ONLY ONE PAYMENT
770 V4=(1+R/1200)^(-T1(I))*A1(I)
780 GOTO 880
790 IF R<>0 THEN 840
800 REM  SPECIAL CASE WHEN R=0
810 V4=N1(I)*A1(I)
820 GOTO 880
830 REM  COMPUTE RESIDUAL VALUE OF SERIES OF PAYMENTS
840 R1=1+R/1200
850 V5=(1-R1^(-N1(I)*F1(I)))/(1-R1^(-F1(I)))
860 V4=A1(I)*R1^(-T1(I))*V5
870 REM  IN ALL CASES, INCLUDE V4 IN V AND V3
880 V=V+V4
890 V3=V3+ABS(V4)
900 NEXT I
910 PRINT "RESIDUAL PRESENT VALUE AT  ";R;"% IS ";V
920 RETURN
930 PRINT
940 PRINT "EFFECTIVE INTEREST RATE,     COMPOUNDED
    MONTHLY, IS ";R
950 END
```

Typing Symbols . . . from p. 59

```
850 R1$="   "&R$
860 CALL SOUND(B*T,494,2)
870 CALL SCREEN(8)
880 GOSUB 4370
890 PRINT ::::"   LEARNING THE KEYBOARD":::
    TAB(11);"SYMBOLS"::::
900 CALL SOUND(40,523,3)
910 CALL SOUND(40,587,3)
920 CALL SOUND(2*T,523,3)
930 CALL SOUND(T,494,3)
940 CALL HCHAR(11,23,44)
950 CALL SOUND(2*T,659,2)
960 CALL HCHAR(7,23,43)
970 CALL HCHAR(7,21,45)
980 CALL SOUND(T,740,2)
990 CALL HCHAR(5,22,40)
1000 CALL SOUND(3*T,784,1)
1010 CALL HCHAR(5,24,41)
1020 CALL HCHAR(5,20,42)
1030 CALL HCHAR(9,22,47)
1040 CALL HCHAR(9,20,94)
1050 CALL SOUND(3*T,740,1)
1060 CALL HCHAR(9,24,61)
1070 CALL HCHAR(7,17,62)
1080 CALL HCHAR(9,18,60)
1090 CALL HCHAR(7,25,34)
1100 CALL SOUND(3*T,659,1)
1110 CALL HCHAR(5,12,36)
1120 CALL HCHAR(5,18,38)
1130 CALL HCHAR(11,21,59)
1140 CALL HCHAR(11,19,58)
1150 CALL SOUND(3*T,740,2)
1160 CALL HCHAR(11,17,63)
1170 CALL HCHAR(5,16,39)
1180 CALL HCHAR(5,14,37)
1190 CALL HCHAR(5,10,35)
1200 CALL SOUND(4*T,587,3)
1210 CALL HCHAR(5,8,64)
1220 CALL HCHAR(5,6,33)
1230 CALL HCHAR(7,19,95)
1240 CALL SOUND(T/2,554,4)
1250 CALL SOUND(T/2,587,4)
1260 CALL SOUND(T/2,659,4)
1270 CALL SOUND(T/2,587,4)
1280 GOSUB 4040
1290 PRINT "THE PERIOD OR DECIMAL":"IS THE ONLY SPECIAL SYMBOL"
1300 PRINT "THAT DOES NOT REQUIRE THE   SHIFT KEY.  THE MOST COMMON"
1310 PRINT "USE IS IN DECIMAL NUMBERS."::"USE THE RIGHT MIDDLE FINGER."
1320 PRINT "CAREFUL--DO NOT HIT "&CHR$(128)&CHR$(129)&CHR$(130)&"!":::::::::::
1330 GOSUB 4460
1340 CALL HCHAR(14,18,46)
1350 GOSUB 4610
1360 PRINT "PRESS <.> FOR THE DRILL."
1370 CALL KEY(0,K,S)
1380 IF K<>46 THEN 1370
1390 CALL COLOR(10,12,1)
1400 GOSUB 4370
1410 CALL COLOR(2,2,12)
1420 CALL COLOR(3,2,12)
1430 CALL COLOR(4,2,12)
1440 G=49
1450 FOR Y=6 TO 22 STEP 2
1460 CALL HCHAR(17,Y,G)
1470 G=G+1
1480 NEXT Y
1490 CALL HCHAR(17,24,48)
1500 CALL HCHAR(23,23,46)
1510 PRINT :::"   COPY THE GIVEN NUMBER":"   THEN PRESS "&CHR$(128)&CHR$(129)&CH
     R$(130):::::::::
1520 RANDOMIZE
1530 FOR I=1 TO 5
1540 A=(INT(1000*(100*RND)))/1000
1550 A$=STR$(A)
1560 B$=""
1570 FOR K=1 TO LEN(A$)
1580 CALL HCHAR(18,K+8,ASC(SEG$(A$,K,1)))
1590 NEXT K
1600 CALL SOUND(150,1397,4)
1610 FOR L=1 TO 10
1620 CALL KEY(0,KEY,S)
1630 IF S<1 THEN 1620
1640 IF KEY=13 THEN 1680
1650 CALL HCHAR(19,L+8,KEY)
1660 B$=B$&CHR$(KEY)
1670 NEXT L
1680 IF B$=A$ THEN 1710
1690 GOSUB 4320
1700 GOTO 1720
1710 GOSUB 4120
1720 CALL HCHAR(18,9,32,10)
1730 CALL HCHAR(19,9,32,10)
1740 NEXT I
1750 GOSUB 3980
1760 CALL COLOR(2,2,1)
1770 CALL COLOR(3,2,1)
1780 CALL COLOR(4,2,1)
1790 PRINT "THE COMMA IS PERHAPS THE":"MOST USED SPECIAL SYMBOL"
1800 PRINT "IN PROGRAMMING."::"IT IS USED TO SEPARATE":"NUMBERS IN SERIES."
1810 PRINT :"USE RIGHT MIDDLE FINGER":"AND SHIFT KEY."::::::::::::
1820 GOSUB 4460
1830 CALL HCHAR(14,18,44)
1840 GOSUB 4610
1850 PRINT "PRESS <,> FOR DRILL."
1860 CALL KEY(0,KEY,S)
1870 IF KEY<>44 THEN 1860
1880 CALL COLOR(10,12,1)
1890 DATA "PRINT I,J,K","DATA 10,12,65","ON N GOTO 130,205,340"
1900 DATA "ON J GOSUB 250,280,420,460","READ X,Y,G","RES 10,5"
1910 DATA "BREAK 120,140,180,205","UNBREAK 130,195","INPUT X,Y,T"
1920 RESTORE 1890
1930 GOSUB 4370
1940 CALL HCHAR(23,23,44)
1950 GOSUB 3840
1960 GOSUB 4370
1970 CALL HCHAR(19,23,43)
1980 CALL HCHAR(19,21,45)
1990 PRINT :::"MINUS OR HYPHEN AND PLUS":::"USE THE SHIFT KEY AND"
2000 PRINT "   MIDDLE FINGER FOR -":"   RING  FINGER FOR +":::::"PRESS - OR + F
     OR DRILL."
2010 CALL KEY(0,KEY,S)
```

Typing Symbols . . .

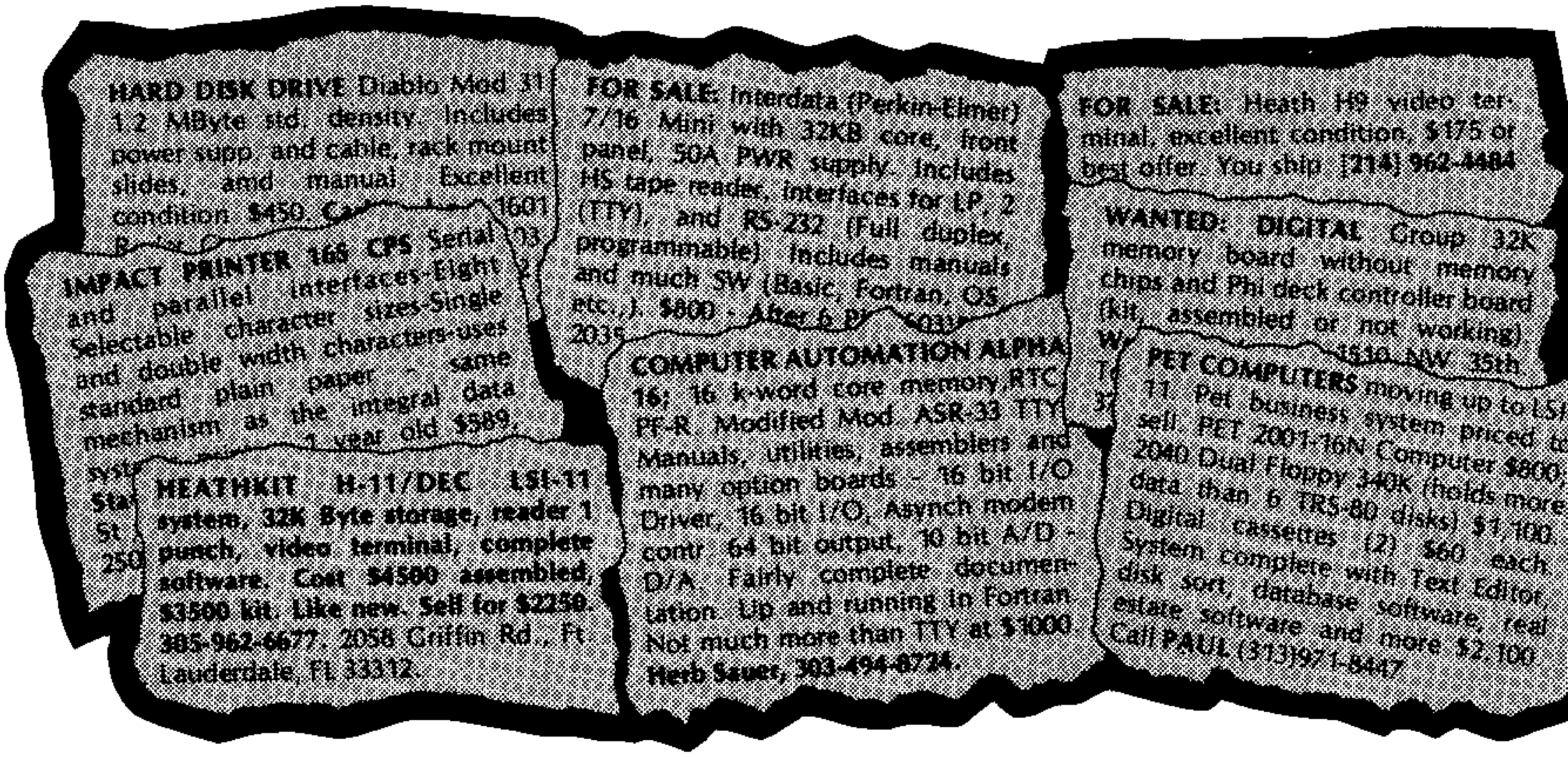
```
2020 IF KEY=45 THEN 2040
2030 IF KEY<>43 THEN 2010
2040 CALL CLEAR
2050 DATA "I+J-K","9-5+2","ON K+9 GOTO 150,160,210","LIST 120-250","-147.25"
2060 DATA "IF J-2 THEN 750","IF A+B THEN 150","PRINT I+J","ON J+2-B GOSUB
     250,320,450"
2070 RESTORE 2050
2080 GOSUB 4370
2090 CALL HCHAR(19,23,43)
2100 CALL HCHAR(19,21,45)
2110 GOSUB 3840
2120 PRINT "PARENTHESES":::R1$
2130 G=49
2140 CALL COLOR(3,2,12)
2150 CALL COLOR(4,2,12)
2160 FOR Y=6 TO 20 STEP 2
2170 CALL HCHAR(23,Y,G)
2180 G=G+1
2190 NEXT Y
2200 CALL HCHAR(23,22,40)
2210 CALL HCHAR(23,24,41)
2220 PRINT CHR$(131)&CHR$(132)&CHR$(133):::"PARENTHESES MUST ALWAYS":"BE
     IN PAIRS."
2230 PRINT ::"USE  RING  FINGER FOR (":"USE LITTLE FINGER FOR )"
2240 PRINT ::"REMEMBER TO "&CHR$(131)&CHR$(132)&CHR$(133)&CHR$(46)::::::
2250 PRINT "PRESS ( OR ) TO START DRILL."
2260 CALL KEY(0,KEY,S)
2270 IF KEY<40 THEN 2260
2280 IF KEY>41 THEN 2260
2290 CALL COLOR(3,2,1)
2300 CALL COLOR(4,2,1)
2310 GOSUB 4370
2320 CALL HCHAR(17,22,40)
2330 CALL HCHAR(17,24,41)
2340 DATA "CALL HCHAR(X,Y,G)","CALL KEY(0,K,S)","INT(A+.05)","PRINT
     SGN(A)","ATN(THETA)"
2350 DATA "EXP(LOG(2))","(A+B)-(D+E)","CALL SOUND(500,440,2)","DIM A(20),B(30)"
2360 RESTORE 2340
2370 GOSUB 3840
2380 GOSUB 4370
2390 CALL HCHAR(17,20,42)
2400 CALL HCHAR(21,22,47)
2410 CALL HCHAR(21,20,94)
2420 PRINT :::"*  MULTIPLY  MIDDLE FINGER":"/  DIVIDE    MIDDLE
     FINGER":"^  EXPONENT  POINTER FINGER"
2430 PRINT ::"REMEMBER TO "&CHR$(131)&CHR$(132)&CHR$(133)
2440 PRINT ::::::"PRESS *  /  OR  ^  FOR DRILL"
2450 CALL KEY(0,KEY,S)
2460 IF KEY=42 THEN 2510
2470 IF KEY=47 THEN 2510
2480 IF KEY<>94 THEN 2450
2490 DATA "3*4/6","INT(6*RND)+1","X*Y^Z","A^B*2/K","S*R/T^2"
2500 DATA "ON INT(X3/2+1) GOTO 25,50,80","2^3/4*F","2*SIN(B)","INT(G*N*RND)"
2510 RESTORE 2490
2520 GOSUB 4370
2530 CALL HCHAR(17,20,42)
2540 CALL HCHAR(21,20,94)
2550 CALL HCHAR(21,22,47)
2560 GOSUB 3840
2570 PRINT "=  USE RING FINGER":"<  USE POINTER FINGER":">  USE POINTER
     FINGER"
2580 PRINT ::"=  EQUALS":"<  LESS THAN":">  GREATER THAN":"<= LESS THAN
     OR EQUAL TO"
2590 PRINT ">= GREATER THAN OR EQUAL TO":"<> NOT EQUAL
     TO"::::"PRESS < = OR > FOR DRILL"
2600 CALL KEY(0,KEY,S)
2610 IF KEY<60 THEN 2600
2620 IF KEY>62 THEN 2600
2630 DATA "IF K<>84 THEN 230","LET J=A>B","IF A<=12 THEN 180",
     "(A+B)<32","I=I+2"
2640 DATA "J=INT(24*RND)","DEF PI=4*ATN(1)","G(X,Y)=D(M,N)",
     "FOR D=1 TO 560"
2650 RESTORE 2630
2660 GOSUB 4370
2670 CALL HCHAR(19,17,62)
2680 CALL HCHAR(21,18,60)
2690 CALL HCHAR(21,24,61)
2700 GOSUB 3840
2710 PRINT "QUOTES MUST BE IN PAIRS.":"USE YOUR RIGHT PINKY FINGER."
2720 PRINT ::"THE DOLLAR SIGN IS ABOVE 4.":"THIS SYMBOL IS USED FOR"
2730 PRINT "STRING VARIABLES."::"TRY TO SHIFT WITH YOUR"
2740 PRINT "LEFT LITTLE FINGER AND AT":"THE SAME TIME PRESS $"
2750 PRINT "WITH YOUR LEFT POINTER.":::::::"PRESS $ OR "" FOR DRILL"
2760 CALL KEY(0,KEY,S)
2770 IF KEY=36 THEN 2790
2780 IF KEY<>34 THEN 2760
2790 GOSUB 4370
2800 CALL HCHAR(17,12,36)
2810 CALL HCHAR(19,25,34)
2820 DATA "PRINT ""HI""","CALL CHAR(96,""FF"")","A$=""BOB""",
     "P=VAL(P$)","A$=STR$(A)"
2830 DATA "POS(M$,K$,1)","K=LEN(NAME$)","PRINT CHR$(42)","CALL
     CHAR(104,B$)"
2840 RESTORE 2820
2850 GOSUB 3840
2860 PRINT ":  ;  &  ARE SYMBOLS TYPED":"WITH THE POINTER
     FINGER.":::::::::::::::
2870 GOSUB 4460
2880 FOR I=1 TO 15
2890 CALL COLOR(9,7,1)
2900 CALL COLOR(9,12,1)
2910 NEXT I
2920 CALL COLOR(9,7,1)
2930 PRINT ::"PRESS : ; OR & FOR DRILL"
2940 CALL KEY(0,KEY,S)
2950 IF KEY=58 THEN 2980
2960 IF KEY=59 THEN 2980
2970 IF KEY<>38 THEN 2940
2980 CALL COLOR(9,12,1)
2990 GOSUB 4370
3000 CALL HCHAR(17,18,38)
3010 CALL HCHAR(23,19,58)
3020 CALL HCHAR(23,21,59)
3030 RESTORE 3040
3040 DATA "INPUT ""NUMBER"":N","PRINT A;B;X","PRINT X;","INPUT "
     "PROMPT"":R","PRINT ""X""&CHR$(E)"
3050 DATA "LEN(NAME$&AGE$)","PRINT :""NAME = "";N$","PRINT :L,"
     "FEET""::","N$&""E""&""12"""
3060 GOSUB 3840
3070 PRINT "# IS ANOTHER SYMBOL":"THAT NEEDS FLEXIBLE FINGERS."
3080 PRINT :"PRESS SHIFT WITH THE LITTLE":"FINGER AND # WITH THE"
3090 PRINT "LEFT MIDDLE FINGER."::"THE MAIN USE OF # IS IN"
3100 PRINT "STATEMENTS FOR FILE NUMBERS.":::
3110 CALL COLOR(3,2,12)
3120 PRINT R1$::CHR$(131)&CHR$(132)&CHR$(133)
3130 CALL HCHAR(21,6,49)
3140 CALL HCHAR(21,8,50)
```

Continued on p. 84

Attention

BARGAIN HUNTERS

Receive Hundreds of Classified Ads Like These Every Month



COMPUTER SHOPPER, the new buy, sell, and trade publication, is ready to help you with the latest information on personal, small business and large-system computers, accessories and software.

Each ad-packed issue is full of bargains you are looking for. Included are ads from individuals throughout the United States who are selling their good, pre-owned equipment just so they can trade-up to new equipment coming on the market.

But, COMPUTER SHOPPER'S bargains won't be yours unless you subscribe. This useful, money-saving publication can become your way to communicate with other buyers, sellers, and traders all over the nation.

Whether you are a hobbyist or a part-time user, COMPUTER SHOPPER will put you in touch with the nationwide computer marketplace in time for you to take advantage of bargain opportunities.

Have something to sell? A COMPUTER SHOPPER subscriber probably wants to buy it.

Looking for a part, component or even a complete system? A COMPUTER SHOPPER subscriber probably wants to sell it.

COMPUTER SHOPPER is THE marketplace for anything in computers and is read by thousands of people who are ready to buy.

COMPUTER SHOPPER offers a unique format in which classified ads are categorized for fast location of specific items. Combining this with low individual ad rates — 10 cents a word — makes it the ideal place for buyers and sellers to communicate. And, its mix of individual, dealer, and manufacturer ads enable subscribers to find what they want at the best price possible.

COMPUTER SHOPPER will work for you in other ways, too. If you are just thinking about getting into computers, it can help you learn product availability and prices before you make a decision. And, through the timely ads, COMPUTER SHOPPER will keep you abreast of changes in the market which could create bargain opportunities for you.

But COMPUTER SHOPPER cannot work for you unless you subscribe. Complete the subscription form today and put COMPUTER SHOPPER on your monthly reading list.

Yes, I Want To Get The Best Deal! Enter my subscription to the worldwide computer equipment & software publication, COMPUTER SHOPPER, today. If I'm not 100% satisfied with my first issue, my money will be refunded in full. But I keep that first issue. It's my FREE sample.

Mail my 13 issues (12 regular issues, plus 1 FREE sample)

☐ Bulk Rate at $10.00 a year.

Name ____________________

Address ____________________

City ____________ State ________ Zip ________

CHARGE TO MY: ☐ MASTERCHARGE. ☐ VISA

Card # ____________________

M/C, Interbank # ____________ Exp. Date ____________

The Nationwide Marketplace for Computer Equipment

COMPUTER SHOPPER

P. O. Box F108 • Titusville, Florida 32780
Telephone 305-269-3211

For faster service, MasterCard & VISA subscriptions only, call TOLL FREE 1-800-528-6050, Ext. 184

Civil Engineering . . . from p. 63

2610-2860	Gives instructions and illustration for solving this type of loading.
2870-2930	Writes a problem for the student to solve.
2940-2960	Draws and labels a beam with a uniform load.
2970	Asks for student's answers.
2980-3070	Solves for the reactions and compares answers. Prints appropriate remarks.
3080-3090	Chooses another problem.
3100-3410	Solves example problem of uniform load on beam that is between two points on the beam.
3420-3470	Writes problem for uniform load between two points on the beam and solves intermediate steps.
3480-3510	Draws and labels the beam for uniform load.
3520	Asks for student's answers.
3530-3550	Calculates solution.
3560-3640	Corrects student's input.
3650	If the answer is incorrect another problem is given.
3660-3680	If the problem is correct, asks the student if he wants more problems.
3690-3890	Student enters his own problem.
3900-3910	Solves and checks the problem.
3920-3980	Gives instructions to solve for combination loads.
3990-4120	Writes the problem for combination loads.
4130-4220	Draws and labels the beam for combination loads.
4230	Asks for student's answers.
4240-4260	Solves for reactions.
4270-4300, 4340-4370	Checks student's answers.
4310-4330	If the solution was correct, asks the student if he wants more problems.
4380	If the solution was incorrect, does another of the same type problem.
4390-4450	Instructions for entering problems.
4460-4640	Draws and labels a general beam.
4650-4850	Asks for student's problem.
4860-4930	Solves the problem and prints the reaction forces.
4940-4980	Asks if there is another problem.
4990	END.
5000-5300	Prints the menu screen of the choices of the types of loading for the simple beams.

Subroutines

5310-5360	Prints error message for length input.
5370-5520	Draws basic beam, supports, and reaction forces.
5530-5550	Delay subroutine.
5560-5580	Prints concentrated load, P.
5590-5620	Subroutine for "Press enter to continue".
5630-5680	Writes problem, length of beam.
5690-5770	Prints concentrated load in problem.
5780-5800	Prints uniform load in problem.
5810-5840	Prints final statements of problem.
5850-5940	Labels concentrated load, P.
5950-6030	Labels length L of beam.
6040-6100	Prints $\Sigma M = 0$ equation.
6110-6180	Shows solution of concentrated load reactions by summing the moments about end A.
6190-6240	Shows solution of A from sum of forces.
6250-6280	Asks for student's answers.
6290-6360	Prints options at the end of each major section.
6370-6410	Draws uniform load for a width Z from Y.
6420-6490	Draws and labels beam with uniform load.
6500-6600	Labels uniform load, W.
6610-6650	Writes problem for uniform load between two points on the beam.
6660-6720	Randomly sets values for problem with uniform load between L1 and L2.
6730-6770	Intermediate algebraic solutions for problems with uniform load; Y and Z are used to draw the beam.

```
680 GOSUB 5530
690 GOTO 5000
700 CALL CLEAR
710 PRINT "NEWTON'S LAWS"
720 PRINT "ARE NECESSARY TO"
730 PRINT "SOLVE REACTION PROBLEMS.":
740 PRINT "1. EQUILIBRIUM OF FORCES"
750 PRINT "        SUM OF FORCES = 0"
760 PRINT "        SUM OF MOMENTS = 0"
770 PRINT :"2. FORCES ALWAYS OCCUR IN"
780 PRINT "    PAIRS  OF EQUAL AND"
790 PRINT "    OPPOSITE FORCES;"
800 PRINT "    ACTION = REACTION":::
810 GOSUB 5590
820 CALL CLEAR
830 PRINT "TO SOLVE A PROBLEM:"::
840 PRINT "DRAW AND LABEL THE PROBLEM."
850 PRINT :"WITH 2 UNKNOWN REACTIONS,"
860 PRINT "SOLVE 2 EQUATIONS:"::
870 PRINT "   SUM OF MOMENTS = 0"
880 PRINT "   SUM OF FORCES = 0"
890 PRINT :"USE CORRECT UNITS.":::
900 GOSUB 5590
910 IF CHOICE=2 THEN 1720
920 CALL CLEAR
930 PRINT "PROBLEM:"
940 PRINT ::"GIVEN A SIMPLE BEAM"
950 PRINT "SUPPORTED AT THE ENDS."
960 PRINT :"IT IS LENGTH L."
970 PRINT :"A CONCENTRATED LOAD P"
980 PRINT "IS AT THE CENTER."
990 PRINT :"IGNORE WEIGHT OF THE BEAM."
1000 GOSUB 5810
1010 I=16
1020 GOSUB 5370
1030 CALL HCHAR(I+1,12,76)
1040 J=16
1050 GOSUB 5560
1060 CALL HCHAR(I-5,16,80)
1070 GOSUB 6040
1080 PRINT "TAKING MOMENTS AT A,":
1090 PRINT "P*L/2 - B*L = 0"
1100 PRINT "         B*L = P*L/2"
1110 PRINT TAB(11);"B = P/2":
1120 PRINT :"NOW TAKE SUM OF FORCES=0"
1130 GOSUB 5590
1140 PRINT "A + B - P = 0"
1150 PRINT "    A + B = P"
1160 PRINT "         A = P - B = P - P/2"
1170 PRINT TAB(9);"A = P/2"
1180 GOSUB 5590
1190 GOSUB 5370
1200 CALL HCHAR(I+1,12,76)
1210 GOSUB 5560
1220 CALL HCHAR(11,16,80)
1230 FOR Y=4 TO 26 STEP 22
1240 CALL HCHAR(22,Y,80)
1250 CALL HCHAR(22,Y+1,47)
1260 CALL HCHAR(22,Y+2,50)
1270 NEXT Y
1280 PRINT "IF THE CONCENTRATED LOAD"
1290 PRINT "IS IN THE CENTER,"
1300 PRINT "A = B = P/2"::
1310 PRINT "FOR EXAMPLE, IF P=1000 LBS.,"
1320 PRINT "   A=500 LBS. AND B=500 LBS."
1330 GOSUB 5590
1340 RANDOMIZE
1350 EX=2
1360 PP=100*(INT(20*RND)+1)
1370 LL=INT(6*RND)+10
1380 GOSUB 5630
1390 GOSUB 5690
1400 GOSUB 5810
1410 GOSUB 5370
1420 GOSUB 5560
1430 GOSUB 5850
1440 GOSUB 5950
1450 GOSUB 6250
1460 IF A<>PP/2 THEN 1510
1470 IF B<>PP/2 THEN 1510
1480 PRINT ::"*** CORRECT!! ***"
1490 GOSUB 5590
1500 GOTO 1560
1510 PRINT :"SORRY, THE REACTIONS ARE"
1520 PRINT :"A =";PP/2
1530 PRINT "B =";PP/2
1540 GOSUB 5590
1550 GOTO 1570
1560 IF EX>2 THEN 1590
```

Continued on p. 80

Civil Engineering . . . from p. 79

```
1570 EX=EX+1
1580 GOTO 1360
1590 GOSUB 6290
1600 IF KEY=49 THEN 1360
1610 IF KEY=51 THEN 5000
1620 I=16
1630 J=16
1640 GOSUB 5370
1650 GOSUB 5560
1660 CALL HCHAR(I+1,12,76)
1670 CALL HCHAR(I-5,16,80)
1680 INPUT "LENGTH OF BEAM = ":LL
1690 GOSUB 5310
1700 INPUT "LOAD P = ":PP
1710 GOTO 1410
1720 CALL CLEAR
1730 PRINT "GIVEN A BEAM OF LENGTH L"
1740 PRINT "SUPPORTED AT ENDS A AND B."
1750 PRINT "A CONCENTRATED FORCE OF"
1760 PRINT "P POUNDS IS APPLIED"
1770 PRINT "D FEET FROM A. IGNORE"
1780 PRINT "THE WEIGHT OF THE BEAM.":::
1790 GOSUB 5810
1800 I=16
1810 J=12
1820 D=5
1830 GOSUB 5370
1840 GOSUB 5560
1850 CALL HCHAR(I-5,J,80)
1860 CALL HCHAR(I-1,9,68)
1870 CALL HCHAR(I+1,16,76)
1880 GOSUB 6040
1890 PRINT :"TAKING MOMENTS AT A"
1900 PRINT :"P*D - B*L = 0"
1910 PRINT TAB(7);"B*L = P*D"
1920 PRINT TAB(9);"B = P*D/L"
1930 PRINT :"NEXT SUM FORCES = 0"
1940 GOSUB 5590
1950 PRINT :"A + B - P = 0"
1960 PRINT "          A = P-B = P - P*D/L"
1970 GOSUB 5590
1980 RANDOMIZE
1990 EX=2
2000 PP=100*(INT(20*RND)+1)
2010 LL=INT(6*RND)+10
2020 D=INT(10*RND)+1
2030 GOSUB 5630
2040 GOSUB 5690
2050 GOSUB 5810
2060 GOSUB 5370
2070 J=INT(D/LL*21)+5
2080 GOSUB 5560
2090 GOSUB 5850
2100 GOSUB 5950
2110 DD$=STR$(D)
2120 FOR E=1 TO LEN(DD$)
2130 EE=J-5
2140 CALL HCHAR(I-1,EE+E,ASC(SEG$(DD$,E,1)))
2150 NEXT E
2160 CALL HCHAR(I-1,EE+E,39)
2170 BB=PP*D/LL+.005
2180 BB=1E-2*(INT(BB*1E2))
2190 AA=PP-BB
2200 IF EX=2 THEN 2250
2210 GOSUB 6250
2220 IF AA<>A THEN 2240
2230 IF BB=B THEN 2300
2240 PRINT :"OUR ANSWERS DON'T AGREE."::
2250 GOSUB 6040
2260 GOSUB 6110
2270 GOSUB 6190
2280 EX=3
2290 GOTO 2000
2300 PRINT :::"** YOU ARE CORRECT **"
2310 GOSUB 5590
2320 EX=EX+1
2330 IF EX<4 THEN 2000
2340 GOSUB 6290
2350 IF KEY=49 THEN 2000
2360 IF KEY=51 THEN 5000
2370 CALL CLEAR
2380 I=16
2390 J=12
2400 GOSUB 5370
2410 GOSUB 5560
2420 CALL HCHAR(I-5,J,80)
2430 CALL HCHAR(I-1,9,68)
2440 CALL HCHAR(I+1,16,76)
2450 INPUT "LENGTH OF BEAM = ":LL
2460 GOSUB 5310
```

Civil Engineering . . .

```
2470 INPUT "LOAD P = ":PP
2480 INPUT "DISTANCE FROM A = ":D
2490 IF D>LL THEN 2510
2500 IF D>=0 THEN 2530
2510 PRINT :"SORRY, 0 <= D <= L"::
2520 GOTO 2480
2530 GOTO 2060
2540 CALL CLEAR
2550 PRINT "GIVEN A SIMPLE BEAM"
2560 PRINT "SUPPORTED AT THE ENDS."
2570 PRINT "IT IS LENGTH L."
2580 PRINT "THERE IS A UNIFORM LOAD"
2590 PRINT "OF W POUNDS PER FOOT."
2600 GOSUB 5810
2610 CALL CLEAR
2620 PRINT "A UNIFORM LOAD CAN BE"
2630 PRINT "THOUGHT OF AS AN"
2640 PRINT "EQUIVALENT RESULTANT"
2650 PRINT "LOAD ACTING AT THE"
2660 PRINT "CENTROID OF THE LOADING":::::::::
2670 CALL HCHAR(21,5,112,7)
2680 CALL HCHAR(20,7,87)
2690 CALL HCHAR(20,8,47)
2700 CALL HCHAR(20,9,76)
2710 CALL HCHAR(22,8,76)
2720 CALL HCHAR(21,15,61)
2730 CALL HCHAR(22,19,115,7)
2740 I=22
2750 J=22
2760 GOSUB 5560
2770 CALL HCHAR(18,21,87)
2780 CALL HCHAR(21,23,76)
2790 CALL HCHAR(21,24,47)
2800 CALL HCHAR(21,25,50)
2810 GOSUB 5590
2820 GOSUB 6420
2830 PRINT "EQUIVALENT LOAD IS"
2840 PRINT "W*L ACTING AT CENTER."
2850 PRINT "SOLVING, A=B=W*L/2"
2860 GOSUB 5590
2870 EX=2
2880 RANDOMIZE
2890 WW=10*(INT(10*RND)+1)
2900 LL=INT(10*RND)+10
2910 GOSUB 5630
2920 GOSUB 5780
2930 GOSUB 5810
2940 GOSUB 6420
2950 GOSUB 5950
2960 GOSUB 6500
2970 GOSUB 6250
2980 AA=WW*LL/2
2990 BB=AA
3000 IF AA<>A THEN 3050
3010 IF BB<>B THEN 3050
3020 PRINT :"** CORRECT **"
3030 GOSUB 5590
3040 GOTO 3100
3050 PRINT :"A=B=W*L/2"
3060 PRINT "A=B=";AA;"POUNDS"
3070 GOSUB 5590
3080 EX=EX+1
3090 GOTO 2890
3100 I=18
3110 Y=16
3120 Z=5
3130 EX=EX+1
3140 GOSUB 6450
3150 PRINT "L = 16 FEET"
3160 PRINT "W = 80 LBS/FT"
3170 PRINT "    ACTING 8 FT FROM A"
3180 PRINT "    TO 12 FT FROM A"::
3190 PRINT "EQUIVALENT FORCE IS"
3200 PRINT "80 LBS/FT * (12 FT - 8 FT)"
3210 PRINT "     = 320 LBS"
3220 PRINT "APPLIED 10' FROM A."
3230 PRINT :"SUM MOMENTS ABOUT A."
3240 GOSUB 5590
3250 GOSUB 5370
3260 J=19
3270 GOSUB 5560
3280 LL=16
3290 PP=320
3300 GOSUB 5850
3310 GOSUB 5950
3320 CALL HCHAR(I-1,10,49)
3330 CALL HCHAR(I-1,11,48)
3340 CALL HCHAR(I-1,12,39)
```

Continued on p. 82

Local Networking . . . from p. 74

N.T.S.U. DATANET 1 PARTS LIST

Master Control Box

IC1 - 74161 Counter
IC2 - TIL311 HEX LED Display
IC3 - 7437 NAND Buffer
IC4, IC5 - 555 Timer
IC6, IC14 - LM386 AUDIO OP AMP
IC7, IC8 - 7805 + 5V Regulator

R1 - 1K
R2 - 100K
R3 - 2.2K
R4, R6 - 10 Ohms
R5 - 2 MEG
R15 - 10K PC MOUNT POT
All Resistors are ¼ Watt

C1, C2, C3, C10, C11, C12 - .1 UF Ceramic Disk
C4, C21 - 4.7 UF Tantalum
C5, C23 - .05 UF Ceramic Disk
C6, C22 - 330 UF Electrolytic
C7, C8 - .1 UF Tantalum

SW1 - Normally Closed SPST Pushbutton Switch
SW3 - SPST Toggle Switch
RECT - 2 AMP Bridge Rectifier
T1 - 6.3V 3 AMP Transformer

---MISC.---
Circuit Board
Enclosure
2 - D-9 Nine Pin Female Connectors
2 - Mini Phone Jacks
Power Cord
Wire

— Very Rough Estimate of Parts Cost —

Integrated Circuits	$14.75
Resistors	.80
Capacitors	4.90
Switches, Transformer etc.	8.50
Misc.	12.50

Master Total Estimated Cost $41.45

Slave Control Box

IC9 - 7404 HEX Inverter
IC10 - 7420 Dual 4 Input NAND
IC11 - 7403 Quad NAND (Open Collector)
IC12, IC13 - LM386 OP AMP

Q1 - 2N2222 Transistor

R7, R14 - 10K
R8, R9 - 330 Ohm
R10 - 4.7K
R11, R12 - 10 Ohms
R13 - 20K PC MOUNT POT
All Resistors are ¼ Watt

C13 - 6.8 UF Tantalum
C14, C19, C20 - .1 UF Ceramic Disk
C15, C16 - 330 UF Electrolytic
C17, C18 - .05 UF Ceramic Disk.

SW2 - SPST Toggle Switch
LED1, LED2 - Single LEDS

--MISC.--
Enclosure
2 - D-9 Nine Pin Female Connectors
2 - Mini Phone Jacks
Circuit Board
Wire
MISC. Hardware

— Very Rough Estimate of Parts Cost —

Integrated Circuits	$3.50
Resistors	1.10
Capacitors	2.45
Switch, LEDS, Transistor	1.00
MISC.	12.00
Slave Total Estimated Cost	**$20.05**

— Network Bus Cable Parts —

17 - D-9 Nine Pin Male Connectors	$34.00
Cable	20.00

Price Estimate for A 16 Slave System

16 Slaves @20.05	$320.80
1 Master @41.45	41.45
Cable Connectors	34.00
Cable	20.00
Total Estimated Cost	**$416.25**

Civil Engineering . . . from p. 81

```
3350 PRINT "320# * 10' - B*16' = 0"
3360 PRINT TAB(5);"B=3200/16 LBS = 200 LBS"::
3370 PRINT "NOW SUM FORCES"
3380 GOSUB 5590
3390 PRINT :"320# - 200# - A = 0"
3400 PRINT "   A = 120 LBS"
3410 GOSUB 5590
3420 EX=EX+1
3430 RANDOMIZE
3440 GOSUB 6660
3450 GOSUB 6610
3460 GOSUB 6730
3470 GOSUB 5590
3480 GOSUB 5370
3490 GOSUB 6370
3500 GOSUB 6500
3510 GOSUB 5950
3520 GOSUB 6250
3530 BB=LOAD*(D2/2+L1)/LL+.005
3540 BB=1E-2*(INT(BB*1E2))
3550 AA=LOAD-BB
3560 IF AA<>A THEN 3610
3570 IF BB<>B THEN 3610
3580 PRINT :"** CORRECT **"
3590 GOSUB 5590
3600 GOTO 3660
3610 PRINT :"SORRY.  IT IS "
3620 PRINT "A = ";AA
3630 PRINT "B = ";BB:
3640 GOSUB 5590
3650 GOTO 3420
3660 GOSUB 6290
3670 IF KEY=49 THEN 3420
3680 IF KEY=51 THEN 5000
3690 CALL CLEAR
3700 I=16
3710 Y=16
3720 Z=5
3730 GOSUB 6450
3740 INPUT "LENGTH OF BEAM IN FT = ":LL
3750 GOSUB 5310
3760 INPUT "LOADING W LB/FT = ":WW
3770 IF WW<>0 THEN 3800
3780 PRINT "IF W=0, A=B=0"
3790 GOTO 3760
3800 INPUT "ACTING AT DISTANCE FROM A ":L1
3810 IF L1<0 THEN 3830
3820 IF L1<LL THEN 3850
3830 PRINT "SORRY, 0 <= L1 < LL"
3840 GOTO 3800
3850 INPUT "TO - DISTANCE FROM A ":L2
3860 IF L2<=L1 THEN 3880
3870 IF L2<=LL THEN 3900
3880 PRINT "SORRY, L1 < L2 <= L"
3890 GOTO 3850
3900 GOSUB 6730
3910 GOTO 3480
3920 CALL CLEAR
3930 PRINT TAB(4);"COMBINATION LOADS"::
3940 PRINT :"TO SOLVE THIS TYPE PROBLEM:"
3950 PRINT :"DRAW AND LABEL THE BEAM."
3960 PRINT :"SUM MOMENTS ABOUT A OR B."
3970 PRINT ::"SUM FORCES"::
3980 GOSUB 5590
3990 CALL CLEAR
4000 RANDOMIZE
4010 GOSUB 6660
4020 GOSUB 6730
4030 PP=100*(INT(15*RND))
4040 D=INT(LL*RND)
4050 GOSUB 5650
4060 IF PP=0 THEN 4080
4070 GOSUB 5710
4080 IF WW=0 THEN 4130
4090 GOSUB 5780
4100 IF L1=LL THEN 4120
4110 GOSUB 6630
4120 GOSUB 5810
4130 I=16
4140 J=INT(D/LL*22)+5
4150 GOSUB 5370
4160 GOSUB 5950
4170 IF WW=0 THEN 4200
4180 GOSUB 6370
4190 GOSUB 6500
4200 IF PP=0 THEN 4230
```

Civil Engineering . . .

```
4210 GOSUB 5560
4220 GOSUB 5850
4230 GOSUB 6250
4240 BB=(PP*D+LOAD*(D2/2+L1))/LL+.005
4250 BB=1E-2*(INT(BB*1E2))
4260 AA=PP+LOAD-BB
4270 IF ABS(AA-A)>.01 THEN 4340
4280 IF ABS(BB-B)>.01 THEN 4340
4290 PRINT ::"** CORRECT **"
4300 GOSUB 5590
4310 GOSUB 6290
4320 IF KEY=49 THEN 3990
4330 IF KEY=51 THEN 5000 ELSE 4390
4340 PRINT ::"SORRY, THE ANSWER I GET IS"
4350 PRINT :"A = ";AA
4360 PRINT "B = ";BB
4370 GOSUB 5590
4380 GOTO 3990
4390 CALL CLEAR
4400 PRINT "YOU MAY ENTER YOUR OWN"
4410 PRINT :"PROBLEMS.  JUST ENTER"
4420 PRINT :"ALL VARIABLES AS "
4430 PRINT :"SHOWN ON THE DIAGRAM."
4440 PRINT ::"I WILL GIVE THE ANSWERS."::::::
4450 GOSUB 5590
4460 I=16
4470 J=12
4480 L1=0
4490 L2=0
4500 D2=0
4510 LOAD=0
4520 GOSUB 5370
4530 GOSUB 5560
4540 CALL HCHAR(I-5,J,80)
4550 Y=16
4560 Z=5
4570 GOSUB 6370
4580 CALL HCHAR(I-3,Y+2,87)
4590 CALL HCHAR(I+2,13,76)
4600 CALL HCHAR(I+1,Y,76)
4610 CALL HCHAR(I+1,Y+1,49)
4620 CALL HCHAR(I+1,Y+Z-1,76)
4630 CALL HCHAR(I+1,Y+Z,50)
4640 CALL HCHAR(I-1,8,68)
4650 INPUT "LENGTH OF BEAM L = ":LL
4660 GOSUB 5310
4670 INPUT "FORCE P = ":PP
4680 IF PP=0 THEN 4740
4690 INPUT "DISTANCE D = ":D
4700 IF D>LL THEN 4720
4710 IF D>=0 THEN 4740
4720 PRINT :"SORRY, 0 <= D <= L":
4730 GOTO 4690
4740 INPUT "LOADING W = ":WW
4750 IF WW=0 THEN 4880
4760 INPUT "DISTANCE FROM A,L1 = ":L1
4770 IF L1<0 THEN 4790
4780 IF L1<LL THEN 4810
4790 PRINT "SORRY, 0 <= L1 <L"
4800 GOTO 4760
4810 INPUT "DISTANCE FROM A,L2 = ":L2
4820 IF L2<=L1 THEN 4840
4830 IF L2<=LL THEN 4860
4840 PRINT "SORRY, L1 < L2 <= LL"
4850 GOTO 4810
4860 D2=L2-L1
4870 LOAD=WW*D2
4880 BB=(PP*D+LOAD*(D2/2+L1))/LL+.005
4890 BB=1E-2*(INT(BB*1E2))
4900 AA=PP+LOAD-BB
4910 PRINT ::"A = ";AA;"POUNDS"
4920 PRINT "B = ";BB;"POUNDS"
4930 GOSUB 5590
4940 PRINT :"ANOTHER PROBLEM?(Y/N) "
4950 CALL KEY(0,KEY,ST)
4960 IF KEY=89 THEN 4480
4970 IF KEY=78 THEN 5000
4980 GOTO 4950
4990 END
5000 CALL CLEAR
5010 CALL SCREEN(5)
5020 FOR E=1 TO 8
5030 CALL COLOR(E,12,12)
5040 NEXT E
5050 PRINT ::::::::
5060 PRINT "SELECT"::
```

Continued on p. 84

Typing Symbols . . . from p. 77

```
3150 CALL HCHAR(21,10,35)
3160 CALL HCHAR(21,12,52)
3170 PRINT :::"PRESS # FOR DRILL"
3180 CALL KEY(0,KEY,S)
3190 IF KEY<>35 THEN 3180
3200 CALL COLOR(3,2,1)
3210 GOSUB 4370
3220 CALL HCHAR(17,10,35)
3230 DATA "OPEN #2:""RS232.TW.BA=110""","CLOSE #6","OPEN #3:""CS1"",INPUT","PRIN
T #2:X"
3240 DATA "CLOSE #4:DELETE","INPUT #2:X,Y","OPEN #1:""RS232/2.BA=9600""","PRINT
#1","PRINT #1:CHR$(B)"
3250 RESTORE 3230
3260 GOSUB 3840
3270 PRINT "THESE SYMBOLS ARE USED":"LESS OFTEN:"::"_  RIGHT POINTER FINGER"
3280 PRINT "?  LEFT POINTER FINGER":"'  RIGHT POINTER FINGER":"%  LEFT POINTER F
INGER"
3290 PRINT "@  TRY LEFT MIDDLE FINGER":"!  TRY LEFT MIDDLE FINGER"
3300 PRINT ::"(NO DRILL FOR THESE SYMBOLS)":::"PRESS ANY SYMBOL":"TO CONTINUE."
3310 CALL KEY(0,KEY,S)
3320 IF S<1 THEN 3310
3330 IF KEY<48 THEN 3370
3340 IF KEY<57 THEN 3310
3350 IF KEY<65 THEN 3370
3360 IF KEY<94 THEN 3310
3370 CALL CLEAR
3380 CALL SCREEN(2)
3390 PRINT "YOU SHOULD KNOW ALL THE":"SYMBOLS NOW.":::"PRESS"::"1  TO START OVER
"
3400 PRINT :"2  FOR FINAL REVIEW"::"3  TO END PROGRAM"::::::::
3410 CALL SCREEN(4)
3420 CALL KEY(0,KEY,S)
3430 IF KEY=49 THEN 870
3440 IF KEY=51 THEN 3830
3450 IF KEY<>50 THEN 3420
3460 CALL CLEAR
3470 PRINT "    SCORE:     RIGHT":TAB(16);"WRONG"::::::::::::::::::
3480 RESTORE 3490
3490 DATA "LEN(NAME$&CITY$)","PRINT #2:AREA","PRINT :L;""IS LENGTH"""
3500 DATA "PRINT :A;B;X","CALL JOYST(2,X,Y)","CALL SOUND(800,-4,2)"
3510 DATA "ON K-48 GOTO 30,150,230","IF A<=B THEN 350","CALL KEY(0,K,S)"
3520 DATA "CALL CHAR(96,A$)","PRINT STR$(VAL(P$))","PRINT ""HI"""
3530 DATA "CALL VCHAR(10,12,42,8)","CALL COLOR(I,F,B)","PRINT COS(Y)"
3540 FOR I=0 TO 14
3550 READ P$(I)
3560 NEXT I
3570 R=0
3580 W=0
3590 FOR I=1 TO 10
3600 J=INT(15*RND)
3610 IF P$(J)=""THEN 3600
3620 GOSUB 4180
3630 IF B$=P$(J)THEN 3670
3640 GOSUB 4330
3650 W=W+1
3660 GOTO 3690
3670 GOSUB 4110
3680 R=R+1
3690 IF R=10 THEN 3740
3700 IF W=10 THEN 3760
3710 CALL HCHAR(4,25,R+48)
3720 CALL HCHAR(5,25,W+48)
3730 GOTO 3790
3740 CALL HCHAR(4,24,49)
3750 GOTO 3770
3760 CALL HCHAR(5,24,49)
3770 CALL HCHAR(5,25,48)
3780 CALL HCHAR(4,25,48)
3790 NEXT I
3800 CALL HCHAR(18,1,32,64)
3810 GOSUB 3980
3820 GOTO 3370
3830 STOP
3840 PRINT :::"COPY THE GIVEN
     PHRASE":"THEN PRESS
     <ENTER>."::::::::::
3850 FOR I=1 TO 9
3860 READ P$(I)
3870 NEXT I
3880 FOR I=1 TO 5
3890 J=INT(9*RND)+1
3900 IF P$(J)=""THEN 3890
3910 GOSUB 4180
3920 IF B$=P$(J)THEN 3950
3930 GOSUB 4320
```

Civil Engineering . . . from p. 83

```
5070 PRINT :"1 CONCENTRATED LOAD, CENTER"
5080 PRINT :"2 CONCENTRATED LOAD ANYWHERE"
5090 PRINT :"3 UNIFORM LOADS"
5100 PRINT :"4 COMBINATION LOADS"
5110 PRINT :"5 PROBLEMS ONLY"
5120 PRINT :"6 YOUR OWN PROBLEMS"
5130 PRINT :"7 END PROGRAM":::
5140 CALL VCHAR(1,2,32,24)
5150 CALL VCHAR(1,31,32,24)
5160 CALL VCHAR(1,1,32,24)
5170 CALL VCHAR(1,32,32,24)
5180 FOR E=1 TO 8
5190 CALL COLOR(E,2,12)
5200 NEXT E
5210 CALL KEY(0,KEY,ST)
5220 CHOICE=KEY-48
5230 IF CHOICE<1 THEN 5210
5240 IF CHOICE>7 THEN 5210
5250 CALL CLEAR
5260 CALL SCREEN(8)
5270 FOR E=1 TO 8
5280 CALL COLOR(E,2,1)
5290 NEXT E
5300 ON CHOICE GOTO 700,700,2540,3920,3990,4390,4990
5310 IF LL>=1 THEN 5360
5320 PRINT "HEY, WHAT KIND OF BEAM"
5330 PRINT "HAS A LENGTH LIKE THAT?!!"
5340 INPUT "TRY AGAIN; L = ":LL
5350 GOTO 5310
5360 RETURN
5370 CALL CLEAR
5380 CALL HCHAR(I,5,120)
5390 CALL HCHAR(I,6,121,21)
5400 CALL HCHAR(I,27,122)
5410 CALL HCHAR(I+1,5,99)
5420 CALL HCHAR(I+1,27,100)
5430 FOR K=4 TO 26 STEP 22
5440 CALL HCHAR(I+2,K,101)
5450 CALL HCHAR(I+2,K+1,102)
5460 CALL HCHAR(I+2,K+2,103)
5470 CALL VCHAR(I+3,K+1,104)
5480 CALL VCHAR(I+4,K+1,105,2)
5490 NEXT K
5500 CALL HCHAR(I+1,4,65)
5510 CALL HCHAR(I+1,28,66)
5520 RETURN
5530 FOR DELAY=1 TO 1000
5540 NEXT DELAY
5550 RETURN
5560 CALL VCHAR(I-4,J,105,3)
5570 CALL VCHAR(I-1,J,106)
5580 RETURN
5590 PRINT :"PRESS ENTER TO CONTINUE"
5600 CALL KEY(0,KEY,ST)
5610 IF KEY<>13 THEN 5600
```

Continued on p. 86

Typing Symbols . . .

```
3940 GOTO 3960
3950 GOSUB 4110
3960 NEXT I
3970 CALL HCHAR(18,1,32,64)
3980 CALL SOUND(3*T,740,2)
3990 CALL SOUND(4*T,587,2)
4000 CALL SOUND(T/2,415,3)
4010 CALL SOUND(T/2,440,3)
4020 CALL SOUND(T/2,494,3)
4030 CALL SOUND(T/2,440,3)
4040 CALL SOUND(5*T,740,2)
4050 CALL SOUND(T,659,2)
4060 CALL SOUND(4*T,659,2)
4070 CALL SOUND(6*T,587,2)
4080 CALL SOUND(1,9999,30)
4090 CALL CLEAR
4100 RETURN
4110 P$(J)=""
4120 CALL SOUND(100,392,2)
4130 CALL SOUND(100,494,2)
4140 CALL SOUND(100,587,2)
4150 CALL SOUND(100,494,2)
4160 CALL SOUND(100,392,2)
4170 RETURN
4180 B$=""
4190 CALL HCHAR(17,1,152,128)
4200 FOR K=1 TO LEN(P$(J))
4210 CALL HCHAR(18,K+2,
     ASC(SEG$(P$(J),K,1)))
4220 NEXT K
4230 CALL SOUND(150,1397,4)
4240 FOR L=1 TO 28
4250 CALL KEY(0,KEY,S)
4260 IF S<1 THEN 4250
4270 IF KEY=13 THEN 4310
4280 CALL HCHAR(19,L+2,KEY)
4290 B$=B$&CHR$(KEY)
4300 NEXT L
4310 RETURN
4320 I=I-1
4330 CALL SOUND(800,-8,0,110,4)
4340 FOR DELAY=1 TO 1000
4350 NEXT DELAY
4360 RETURN
4370 CALL CLEAR
4380 PRINT R1$:
     :" "&R1$::" "&CHR$
     (152)&R$&" "&CHR$
     (153)::" "&R1$
4390 CALL HCHAR(23,5,131)
4400 CALL HCHAR(23,6,132)
4410 CALL HCHAR(23,7,133)
4420 CALL HCHAR(23,25,128)
4430 CALL HCHAR(23,26,129)
4440 CALL HCHAR(23,27,130)
4450 RETURN
4460 CALL HCHAR(20,16,152,5)
4470 CALL HCHAR(21,16,152,5)
4480 CALL HCHAR(15,16,97)
4490 CALL VCHAR(16,16,96,4)
4500 CALL VCHAR(15,17,105)
4510 CALL VCHAR(15,18,106)
4520 CALL VCHAR(16,17,104,4)
4530 CALL VCHAR(16,18,107,4)
4540 CALL VCHAR(15,19,124)
4550 CALL VCHAR(16,19,152,4)
4560 CALL VCHAR(17,20,121)
4570 CALL VCHAR(17,21,122)
4580 CALL VCHAR(18,20,120,2)
4590 CALL VCHAR(18,21,123,4)
4600 RETURN
4610 FOR I=1 TO 15
4620 CALL COLOR(10,7,1)
4630 CALL COLOR(10,12,1)
4640 NEXT I
4650 CALL COLOR(10,7,1)
4660 RETURN
4670 END
```


PROGRAMMING HINT (1) **By Regena**

HCHAR Pause Eliminator

Sometimes there is a noticeable pause when you are printing a character, word, or phrase from a data statement in a FOR-NEXT loop. For example:

```
DATA 71,82,69,69,84,73,78,71,83
FOR Y=10 TO 18
READ L
CALL HCHAR(9,Y,L)
NEXT Y
```

The pause is before the last letter is printed.

This problem can be circumvented by adding a non-visible character such as ASCII 32 to the DATA statement and printing it at the end of your series.

```
DATA 71,82,69,69,84,73,78,71,83,32
FOR Y=10 TO 19
READ L
CALL HCHAR(9,Y,L)
NEXT Y
```

Civil Engineering . . . from p. 84

```
5620 RETURN
5630 CALL CLEAR
5640 PRINT "PROBLEM":::
5650 PRINT "GIVEN A SIMPLE BEAM"
5660 PRINT "SUPPORTED AT THE ENDS."
5670 PRINT "IT IS";LL;"FEET LONG."
5680 RETURN
5690 IF CHOICE>2 THEN 5710
5700 PRINT :"IGNORE WEIGHT OF THE BEAM."
5710 PRINT "A CONCENTRATED LOAD OF"
5720 PRINT PP;"POUNDS IS"
5730 IF CHOICE=1 THEN 5760
5740 PRINT D;"FEET FROM END A."
5750 RETURN
5760 PRINT "AT THE CENTER OF THE BEAM."
5770 RETURN
5780 PRINT :"THERE IS A UNIFORM LOAD"
5790 PRINT "OF";WW;"POUNDS/FOOT"
5800 RETURN
5810 PRINT :"FIND THE REACTION FORCES."::
5820 PRINT "DRAW AND LABEL THE PROBLEM."::
5830 GOSUB 5590
5840 RETURN
5850 LB$=STR$(PP)
5860 FOR II=1 TO LEN(LB$)
5870 JJ=II+J-4
5880 CALL HCHAR(I-5,JJ,ASC(SEG$(LB$,II,1)))
5890 NEXT II
5900 CALL HCHAR(I-5,JJ+1,32)
5910 CALL HCHAR(I-5,JJ+2,76)
5920 CALL HCHAR(I-5,JJ+3,66)
5930 CALL HCHAR(I-5,JJ+4,83)
5940 RETURN
5950 FT$=STR$(LL)
5960 FOR II=1 TO LEN(FT$)
5970 JJ=12+II
5980 CALL HCHAR(I+1,JJ,ASC(SEG$(FT$,II,1)))
5990 NEXT II
6000 CALL HCHAR(I+1,JJ+2,70)
6010 CALL HCHAR(I+1,JJ+3,69,2)
6020 CALL HCHAR(I+1,JJ+5,84)
6030 RETURN
6040 CALL HCHAR(23,3,98)
6050 CALL HCHAR(23,4,77)
6060 CALL HCHAR(23,6,61)
6070 CALL HCHAR(23,8,48)
6080 PRINT "WRITE THE EQUATION"
6090 GOSUB 5590
6100 RETURN
6110 PRINT :"TAKING MOMENTS AT A."
6120 PRINT :"P*D - B*L = 0"
6130 PRINT "          B = P * D/L"
6140 PRINT "          B =";PP;"*";D;"/";LL
6150 PRINT "          B = ";BB;"POUNDS"
6160 PRINT :"NOW FIND A."
6170 GOSUB 5590
6180 RETURN
6190 PRINT :"SUM OF FORCES = 0"
6200 PRINT "P-A-B = 0"
6210 PRINT "     A = P-B =";PP;"-";BB
6220 PRINT "     A = ";AA;"POUNDS"
6230 GOSUB 5590
6240 RETURN
6250 PRINT :"WHAT ARE A AND B IN POUNDS?"
6260 INPUT "A = ":A
6270 INPUT "B = ":B
6280 RETURN
6290 PRINT ::"DO YOU WANT MORE PROBLEMS?"
6300 PRINT :"1 YES, SAME KIND"
6310 PRINT "2 YES, MY OWN PROBLEMS"
6320 PRINT "3 NO, DO SOMETHING ELSE"
6330 CALL KEY(0,KEY,ST)
6340 IF KEY<49 THEN 6330
6350 IF KEY>51 THEN 6330
6360 RETURN
6370 CALL HCHAR(I-1,Y,112,Z)
6380 CALL HCHAR(I-2,Y,112,Z)
6390 CALL VCHAR(I-2,Y-1,114,2)
6400 CALL VCHAR(I-2,Y+Z,113,2)
6410 RETURN
6420 I=16
6430 Y=6
6440 Z=21
6450 GOSUB 5370
6460 GOSUB 6370
6470 CALL HCHAR(I-3,16,87)
6480 CALL HCHAR(I+1,16,76)
6490 RETURN
6500 X=INT(Y+Z/2-3)
6510 UL$=STR$(WW)
6520 FOR E=1 TO LEN(UL$)
6530 CALL HCHAR(I-3,X+E-1,ASC(SEG$(UL$,E,1)))
6540 NEXT E
6550 CALL HCHAR(I-3,X+E,76)
6560 CALL HCHAR(I-3,X+E+1,66)
6570 CALL HCHAR(I-3,X+E+2,47)
6580 CALL HCHAR(I-3,X+E+3,70)
6590 CALL HCHAR(I-3,X+E+4,84)
6600 RETURN
6610 GOSUB 5630
6620 GOSUB 5780
6630 PRINT "ACTING FROM";L1;"FEET FROM A"
6640 PRINT "TO";L2;"FEET FROM A"
6650 RETURN
6660 LL=INT(8*RND)+12
6670 WW=10*(INT(4*RND)+5)
6680 LIM1=INT(3*LL/4)
6690 L1=INT(LIM1*RND)
6700 LIM2=LL-L1
6710 L2=INT(LIM2*RND)+L1+1
6720 RETURN
6730 D2=L2-L1
6740 LOAD=WW*D2
6750 Y=INT(L1/LL*22)+6
6760 Z=INT(D2/LL*22)-1
6770 RETURN
```

Battle At Sea . . . from p. 37

```
4830 IF DS(Z4)=LE THEN 4920
4840 IF DS(Z4)=0 THEN 4870
4850 W=W+1
4860 GOTO 4920
4870 SCORE=SCORE+1
4880 IF T=0 THEN 4910
4890 GOSUB 4990
4900 GOTO 4920
4910 GOSUB 4980
4920 NEXT Z4
4930 IF T=0 THEN 4960
4940 W1=W
4950 GOTO 2880
4960 W=W1
4970 GOTO 2880
4980 SCP=SCORE
4990 CALL HCHAR(23,1,32,32)
5000 FOR X3=1 TO LEN(PR$)+10
5010 CALL HCHAR(23,X3+6,ASC(SEG$(PR$&" DESTROYED",
     X3,1)))
5020 NEXT X3
5030 IF T=0 THEN 5070
5040 CALL VCHAR(20,20,SCORE+48)
5050 IF SCORE=5 THEN 5120
5060 RETURN
5070 CALL HCHAR(20,27,SCORE+48)
5080 IF SCORE=5 THEN 5120
5090 RETURN
5100 PRINT "THE COMPUTER WINS AGAIN"
5110 GOTO 5130
5120 PRINT "YOU JUST GOT LUCKY THIS TIME"
5130 PRINT "IF YOU WISH TO PLAY AGAIN"
5140 PRINT "ENTER ""Y"",IF NOT ENTER ""N"""
5150 INPUT NG$
5160 IF NG$="N"THEN 5330
5170 IF NG$="Y"THEN 5200
5180 CALL SOUND(200,110,0)
5190 GOTO 5130
5200 FOR L=1 TO 10
5210 FOR L1=1 TO 10
5220 P(L,L1)=0
5230 O(L,L1)=0
5240 NEXT L1
5250 NEXT L
5260 FOR L=1 TO 5
5270 FOR L1=1 TO 5
5280 SH(L,L1,1)=0
5290 SH(L,L1,2)=0
5300 NEXT L1
5310 NEXT L
5320 GOTO 880
5330 CALL CLEAR
5340 STOP
5350 NNN=0
5360 AAA=0
5370 FOR X=1 TO LE-1
5380 IF NNN=1 THEN 5410
5390 IF SH(S,X,1)=SH(S,X+1,1)THEN 5440
5400 IF AAA=1 THEN 2010
5410 IF SH(S,X,2)<>SH(S,X+1,2)THEN 2010
5420 NNN=1
5430 GOTO 5450
5440 AAA=1
5450 NEXT X
5460 RETURN
5470 END
```

Harried Housewife . . . from p. 39

```
4450 CALL HCHAR(X,Y+1,78)
4460 CALL HCHAR(X,Y+2,69)
4470 RETURN
4480 DATA 80,82,69,83,83,32,69,
     78,84,69,82,32,84,79,32,67,
     79,78,84,73,78,85,69,32
4490 RESTORE 4480
4500 FOR Y=3 TO 26
4510 READ GR
4520 CALL HCHAR(23,Y,GR)
4530 NEXT Y
4540 CALL KEY(0,KEY,ST)
4550 IF KEY<>13 THEN 4540
4560 CALL HCHAR(4,24,32,5)
4570 CALL HCHAR(23,2,32,25)
4580 ON (K1-64)GOSUB 4630,4670,
     4710,4750,4790,4830,4870,
     4910,4950,4990,5030,5070,
     5110,5150,5190,5230
4590 CALL HCHAR(X,Y,K1)
4600 ON (K2-64)GOSUB 4630,4670,
     4710,4750,4790,4830,4870,
     4910,4950,4990,5030,5070,
     5110,5150,5190,5230
4610 CALL HCHAR(X,Y,K2)
4620 GOTO 1130
4630 X=3
4640 Y=4
4650 GOSUB 5270
4660 RETURN
4670 X=3
4680 Y=9
4690 GOSUB 5310
4700 RETURN
4710 X=3
4720 Y=14
4730 GOSUB 5270
4740 RETURN
4750 X=3
4760 Y=19
4770 GOSUB 5310
4780 RETURN
4790 X=8
4800 Y=4
4810 GOSUB 5310
4820 RETURN
4830 X=8
4840 Y=9
4850 GOSUB 5270
4860 RETURN
4870 X=8
4880 Y=14
4890 GOSUB 5310
4900 RETURN
4910 X=8
4920 Y=19
4930 GOSUB 5270
4940 RETURN
4950 X=13
4960 Y=4
4970 GOSUB 5270
4980 RETURN
4990 X=13
5000 Y=9
5010 GOSUB 5310
5020 RETURN
5030 X=13
5040 Y=14
5050 GOSUB 5270
5060 RETURN
5070 X=13
5080 Y=19
5090 GOSUB 5310
5100 RETURN
5110 X=18
5120 Y=4
5130 GOSUB 5310
5140 RETURN
5150 X=18
5160 Y=9
5170 GOSUB 5270
5180 RETURN
5190 X=18
5200 Y=14
5210 GOSUB 5310
5220 RETURN
5230 X=18
5240 Y=19
5250 GOSUB 5270
5260 RETURN
5270 FOR XX=X-1 TO X+2
5280 CALL HCHAR(XX,Y-1,43,4)
5290 NEXT XX
5300 RETURN
5310 FOR XX=X-1 TO X+2
5320 CALL HCHAR(XX,Y-1,44,4)
5330 NEXT XX
5340 RETURN
5350 CH(SS)=HH(1)
5360 X=3
5370 Y=4
5380 RETURN
5390 CH(SS)=HH(2)
5400 X=3
5410 Y=9
5420 RETURN
5430 CH(SS)=HH(3)
5440 X=3
5450 Y=14
5460 RETURN
5470 CH(SS)=HH(4)
5480 X=3
5490 Y=19
5500 RETURN
5510 CH(SS)=HH(5)
5520 X=8
5530 Y=4
5540 RETURN
5550 CH(SS)=HH(6)
5560 X=8
5570 Y=9
5580 RETURN
5590 CH(SS)=HH(7)
5600 X=8
5610 Y=14
5620 RETURN
5630 CH(SS)=HH(8)
5640 X=8
5650 Y=19
5660 RETURN
5670 CH(SS)=HH(9)
5680 X=13
5690 Y=4
5700 RETURN
5710 CH(SS)=HH(10)
5720 X=13
5730 Y=9
5740 RETURN
5750 CH(SS)=HH(11)
5760 X=13
5770 Y=14
5780 RETURN
5790 CH(SS)=HH(12)
5800 X=13
5810 Y=19
5820 RETURN
5830 CH(SS)=HH(13)
5840 X=18
5850 Y=4
5860 RETURN
5870 CH(SS)=HH(14)
5880 X=18
5890 Y=9
5900 RETURN
5910 CH(SS)=HH(15)
5920 X=18
5930 Y=14
5940 RETURN
5950 CH(SS)=HH(16)
5960 X=18
5970 Y=19
5980 RETURN
```

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)	☐ $15	☐ $18	☐ $25	☐ $40
2-yr (12 issues)	☐ $28	☐ $34		
3-yr (18 issues)	☐ $39	☐ $48		

Attention Dealers: Please write or call for bulk subscription rates. Also ask about our inexpensive magazine directory listings to publicize your retail locations. Call: (503)-485-8796

99'er BOOKSTORE

QTY	TITLE	PRICE	TOTAL AMOUNT
		POSTAGE	
		TOTAL	

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.

99'er-ware™

*U.S. ONLY—FOR SHIPMENT OUTSIDE THE U.S. INQUIRE SEPARATELY

QTY	ITEM	PRICE	TOTAL AMOUNT
	TI-SETTE ADAPTOR	$3.50	
	SHIPPING – 75¢ ea.*		
	DIGITAL COMPUTER CASSETTES in LIBRARY ALBUM	$9.95 or 3 for $27.00	
	SHIPPING – $2.00 for 1st ALBUM, 75¢ ea. add.*		
	MINI-FLEX DISKETTE FILE	$24.95	
	SHIPPING – $2.50 ea.*		
	FRICTION-FEED KIT FOR EPSON MX-80	$49.95	
	SHIPPING – $2.00 ea.*		
		TOTAL	

Back issues of *99'er Magazine* available — $3.95 ea. (postpaid) issue(s) desired ________ (month/month, year) ________ quantity TOTAL ________

Address shown is: ☐ Business ☐ Home

NAME ______________________ **PLEASE**

ADDRESS ______________________ **PRINT**

CITY ____________ STATE ______ ZIP ________

99'er MAGAZINE SUBSCRIPTION	
TOTAL 99'er BOOKSTORE	
TOTAL 99'er-ware	
TOTAL	

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

Bill my: ☐ VISA ☐ Master Charge

Account No. ________

MC Only — List 4 digits above your name. ____

Expiration Date ____________

Signature ____________

PLACE STAMP HERE

99'er magazine™

P.O. Box 5537

Eugene, OR 97405

Almost Everything You Ever Wanted to Know About Music. . .

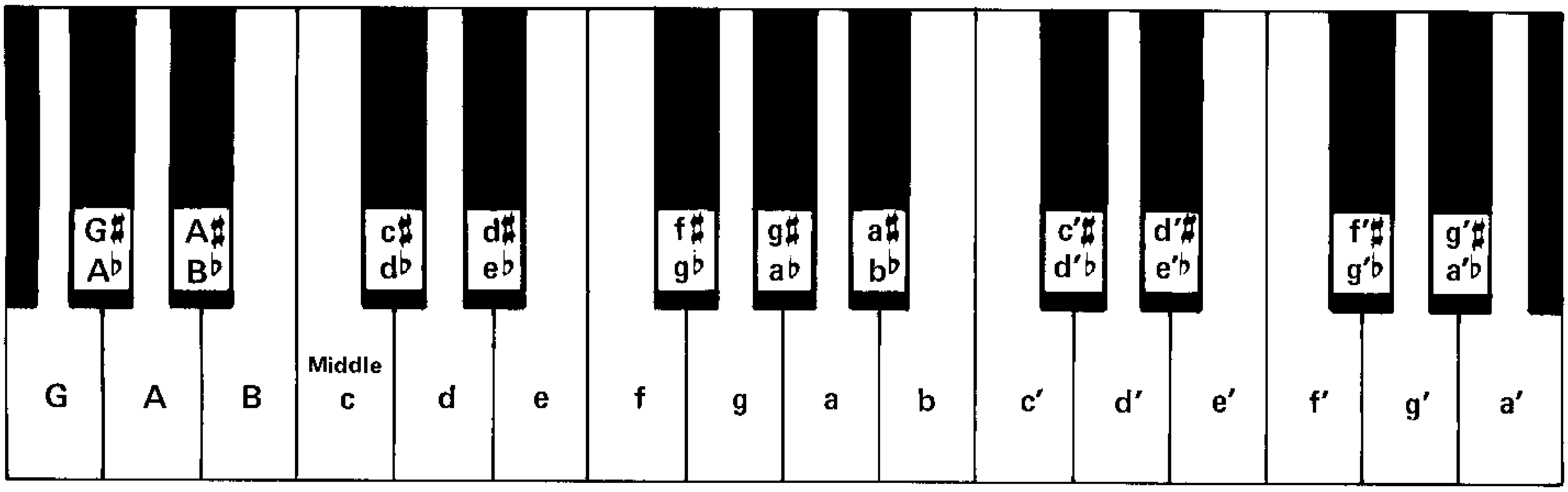


Figure 1. Piano Keyboard

. . . BUT WERE AFRAID TO ASK

By Norma and John Clulow

Is music terminology Greek to you? Do you feel deficient in certain areas of your musical ability? How are your listening skills? If you enjoy music and want to test and improve your abilities, TI's *Music Skills Trainer* can be a valuable tool. This program provides practice in aural recognition of pitches, intervals, and chords, and develops your ability to remember musical phrases. You can control the complexity of each drill by selecting various options including note range, use of sharps (♯) and flats (♭), types of chords and intervals, and the playing of random music between examples.

Since the program is designed to provide drill and does not teach the underlying concepts involved, this article will first cover relevant aspects of music theory. We'll then follow it up with a review of *Music Skills Trainer.*

The Scale

The fundamental concept involved is that of the scale—an ordered group of tones within an octave. The C Major Scale, with which almost everyone is familiar, provides the standard pattern for every major scale (Do-Re-Mi-Fa-Sol-La-Ti-Do). This pattern originated with the Greeks and is based upon the tetrachord. A tetrachord can be thought of as half of a scale, it consists of four tones arranged such that they contain two whole steps followed by a half step. Refer to the diagram of a piano keyboard in Figure 1. Starting at middle c, each progression up the keyboard represents a half step or semitone. For example, all the following represent half steps: c-c♯, c♯-d, d-d♯, d♯-e, e-f, etc. The first tetrachord for the C Major Scale consists of the following two whole steps: c-d and d-e followed by the half step e-f. The second tetrachord begins with g and again consists of two whole steps followed by a half step, ending with c' (an octave above middle c). This tetrachord pattern (1+1+ ½) was referred to as a "diatonic" tetrachord, and the major scale formed by two such tetrachords separated by a whole step is therefore called a "diatonic scale".

In order to accommodate Oriental and other music, Greek theorists modified the two middle tones of the diatonic tetrachord in several ways. One of these, called the "chromatic tetrachord", consisted of the pattern 1½+½+½ (e.g., c, d♯, e, f). Various combinations of these two tetrachords necessitate the division of an octave into the familiar twelve equally spaced intervals referred to as the chromatic scale: c, c♯, d, d♯, e, f, f♯, g, g♯, a, a♯, b, c'.

"Pitch" refers to the location of one of these tones in a scale and is defined by a regular frequency of vibrations. In the United States the standard assignment for "a" is 440 vibrations per second. It happens that a pure octave differs from any reference pitch by a factor of exactly 2, so that a' = 880 and A = 220.

Although knowledge of frequencies is not required for use of the *Music Skills Trainer,* you may be interested to know how frequencies are assigned to other scale positions. Because each octave is divided into twelve equally spaced intervals, the factor $2^{\frac{1}{12}}$ is used to define the relative frequencies of successive tones. For example,

$$\text{if} \quad a = 440;$$
$$a\sharp = 440 \times 2^{\frac{1}{12}},$$
$$b = a\sharp \times 2^{\frac{1}{12}} = a \times 2^{\frac{1}{12}} \times 2^{\frac{1}{12}} = a \times (2^{\frac{1}{12}})^2.$$

Given any reference frequency, f_0, then the relative pitch of any other scale position, f, can be calculated by counting the number of half steps to that position, N, and using the formula:

$$f = f_0(2^{\frac{1}{12}})^N.$$

The following program calculates and plays a chromatic scale beginning with middle c (262).

```
100 REM  ************
110 REM  * MUSIC 1 *
120 REM  ************
130 REM  99'ER VERSION 7.81.1
140 REM  BY NORMA AND JOHN CLULOW
150 REM
160 REM
170 F0=262
180 FOR N=0 TO 12
190 F=F0*(2^(1/12))^N
200 CALL SOUND(-600,F,0)
210 NEXT N
220 STOP
```

Scales in Various Keys

Now let us return to the diatonic (major) scale. A major scale can have a starting or "root" note of any of the twelve chromatic pitches. As in the case discussed above, a major scale is constructed, starting with the root, with two diatonic tetrachords (1+1+½) separated by a whole step. A more convenient way to construct a major scale is simply to remember that half steps occur between the 3rd & 4th and 7th & 8th tones. Referring to Figure 1, a major scale with e♭ as the root would be constructed using the following steps:

e♭ f g a♭ b♭ c' d' e♭'
1 1 ½ 1 1 1 ½

Steps

This scale is referred to as an E♭ Major Scale or a scale in the key of e♭ since e♭ is the root. Similarly, a major scale in the key of G is constructed as follows:

g a b c' d' e' f♯' g'
1 1 ½ 1 1 1 ½

Steps

While there are twelve such different diatonic scales, they all sound the same because they are based on the same pattern of diatonic steps. The following program plays these scales beginning with C Major.

```
100 REM  ***********
110 REM  * MUSIC 2 *
120 REM  ***********
130 REM  99'ER VERSION 7.81.1
140 REM  BY NORMA AND JOHN CLULOW
150 REM
160 REM
170 F0=262
180 FOR I=0 TO 11
190 F=F0*(2^(1/12))^I
200 FOR J=1 TO 8
210 N=2
220 IF J=4 THEN 240
230 IF J=8 THEN 240 ELSE 250
240 N=1
250 F=F*(2^(1/12))^N
260 CALL SOUND(-600,F,0)
270 NEXT J
280 NEXT I
290 STOP
```

Intervals

An interval is the difference in pitch between two notes. Interval names indicate the number of included tones of the major scale. Starting with middle c in Figure 1, the basic interval names are as follows: c-c, unison (prime); c-d, second; c-e, third; c-f, fourth; c-g, fifth; c-a, sixth; c-b, seventh; and c-c', octave. c-f is a fourth because it includes the following diatonic tones of the C Major Scale: c, d, e, and f. Similarly, in the E♭ Major Scale a fourth is e♭ -a♭ , and in the G Major Scale a fourth is g-c'. However, as in the case of scales, an interval in one key sounds like that interval in another.

Four of the eight intervals can exist in one of four forms. If the upper note of the interval lies within the major scale of the lower or root note, the interval may be classified as "major." If the upper note is lowered a half step, however, the interval then becomes "minor." For example, c-e is a major third and c-e♭ a minor third. This rule applies to four intervals; the second, third, sixth, and seventh. The remaining intervals — fourth, fifth, and octave — are classified as "perfect" — they do not exist in major and minor forms. The following program plays all of the intervals above in the C Major scale, i.e., with middle c as the lower or root note.

```
100 REM  ***********
110 REM  * MUSIC 3 *
120 REM  ***********
130 REM  99'ER VERSION 7.81.1
140 REM  BY NORMA AND JOHN CLULOW
150 REM
160 REM
170 DIM N$(11),F(11)
180 FOR I=0 TO 11
190 READ N$(I),F(I)
200 CALL CLEAR
210 PRINT "           "&N$(I)::::::::::::
220 FOR J=1 TO 2
230 CALL SOUND(1000,262,0)
240 CALL SOUND(1000,F(I),0)
250 CALL SOUND(3000,110,30)
260 NEXT J
270 NEXT I
280 DATA UNISON,262,MAJOR SECOND,
    294,MINOR SECOND,277,MAJOR
    THIRD,330,MINOR THIRD,311,
    FOURTH,349,FIFTH,392
290 DATA MAJOR SIXTH,440,
    MINOR SIXTH,415,MAJOR SEVENTH,
    494,MINOR SEVENTH,466,OCTAVE,
    523
300 STOP
```

The remaining two categories of intervals "augmented" and "diminished" are not used in the TI *Music Skills Trainer* and so will not be discussed in detail. They are formed as follows: augmented--a major or perfect interval is made one half step larger; diminished --a minor or perfect interval is made one half step smaller.

Finally, intervals may be classified according to which note is played first. If the lower note is played first, the interval is said to be ascending (c-e), and if the upper note is played first, it is descending (e-c).

Chords

A chord is several notes played simultaneously, usually three or more, When a chord consists of three tones it is called a triad. Given any major scale, four triads can be formed from the starting note (root) of that scale: major, minor, augmented, and diminished. A major triad consists of the root, the third, and the fifth. For example, in a C major scale, starting with the root c, the third is c-e, and the fifth is c-g. The major chord is then c-e-g. Similarly, in the E♭ major scale, given the root e♭ , the third g, and the fifth b♭ , the major chord is e♭-g-b♭.

A major chord is changed to a minor chord by lowering the second note (i.e., the third) one half step. For example, the C major chord c-e-g becomes the c minor chord c-e♭-g and the e♭major chord becomes the e♭minor chord e♭-g♭-b♭.

A minor chord can further be changed to a diminished chord by lowering the third note (i.e., the fifth) one half step. For example, the c minor chord c-e♭-g becomes the c diminished chord c-e♭-g♭ and the e♭minor chord e♭-g♭-b♭ becomes the e♭diminished chord e♭-g♭-b♭♭. (b♭♭ is called b double flat and is the same note as a.)

The augmented chord is formed by raising the third note of the major chord (i.e. the fifth) one half step. For example, the c major chord c-e-g becomes the c augmented chord c-e-g♯ and the e♭ major chord becomes the e♭ augmented chord e♭-g-b.

As in the case of scales and intervals, chords with the same name sound alike. All major chords sound alike; all minor chords sound alike, etc.

If the lowest note of the chord is the root, the chord is said to be in root position. All four types of triads (chords) can be played in inverted form, however. For example, the C major chord c-e-g may be altered from its root position form to one of the following inversions by making the lowest note either the third or the fifth: e-g-c′ and g-c′-e′. Similarly, the inverted forms for the e♭minor, which in root position is written or played e♭-g♭-b♭, are g♭-b♭-e♭′ and b♭-e♭′-g♭′.

Chords of more than three notes can be formed, and there are several different varieties. One of them, the seventh, is used in the *Music Skills Trainer* and so will be described here. The seventh chord contains the root, third, fifth, and the seventh lowered by a half step. For example, a seventh in the key of Cmajor is c-e-g and b lowered by a half step or b♭. Similarly, in the key of e♭ the seventh chord is e♭-g-b♭-d♭ (d lowered by a half step).

While the seventh chord contains four notes, the TI-99/4 can play only three notes simultaneously; therefore, following traditional rules of harmony the fifth of the chord (third note) may be omitted to give a seventh in the form c-e-b♭. As in the case of triads, the seventh may appear in inverted forms.

The following program will allow you to compare and contrast major, minor, augmented, diminished, and seventh chords in their root and invertec forms.

```
100 REM  ***********
110 REM  * MUSIC 4 *
120 REM  ***********
130 REM  99'ER VERSION 7.81.1
140 REM  BY NORMA AND JOHN CLULOW
150 REM
160 REM
170 REM   DISPLAY MENU
180 CALL CLEAR
190 PRINT TAB(10);"1 MAJOR"::TAB
    (10);"2 MINOR"::TAB(10);
    "3 DIMINISHED"::TAB(10);
    "4 AUGMENTED"::
200 PRINT TAB(10);"5 SEVENTH"::::
    :::TAB(8);"YOUR CHOICE?"
210 REM   ACCEPT CHOICE
220 CALL KEY(0,KEY,STATUS)
230 IF KEY<49 THEN 210
240 IF KEY>53 THEN 210
250 CALL HCHAR(23,23,KEY)
260 KEY=KEY-48
270 REM   ASSIGN FREQUENCIES
280 F1=262
290 ON KEY GOTO 300,330,360,390,420
300 F2=330
310 F3=392
320 GOTO 440
330 F2=311
340 F3=392
350 GOTO 440
360 F2=311
370 F3=370
380 GOTO 440
390 F2=330
400 F3=415
410 GOTO 440
420 F2=330
430 F3=466
440 REM   PLAY CHORD
450 CALL SOUND(1000,F1,0)
460 CALL SOUND(1000,F2,0)
470 CALL SOUND(1000,F3,0)
480 CALL SOUND(1000,440,30)
490 CALL SOUND(1000,F1,0,F2,0,F3,0)
500 REM   INVERSIONS
510 PRINT :TAB(6);"INVERSIONS (Y/N)
    ?"
520 CALL KEY(0,KEY,STATUS)
530 IF KEY=78 THEN 170
540 IF KEY<>89 THEN 520
550 CALL HCHAR(23,26,KEY)
560 REM   PLAY INVERSIONS
570 CALL SOUND(1000,F2,0,F3,0,
    2*F1,0)
580 CALL SOUND(1000,440,30)
590 CALL SOUND(1000,F3,0,2*F1,0,
    2*F2,0)
600 CALL SOUND(1000,440,30)
610 GOTO 170
```

TI Music Skills Trainer

The *Music Skills Trainer* from Texas Instruments is a program written in TI BASIC (it will also run in Expended BASIC without modification). The program is available on cassette or diskette.

Four types of drill are provided; Pitch Guess, Interval Recognition, Chord Recognition, and Phrase Recall. The user selects the type of drill desired from a menu.

Pitch Guess

In this drill, you try to identify the pitch of a single note. While it might seem at first that this would require perfect pitch, you will find after several examples that you have "tuned in" and are able to identify pitches by relating each new one to the one that has preceeded. The difficulty of this exercise can be controlled by specifying the starting note and range size in half steps (up to two octaves). In addition, you can choose to have notes selected from either the C major diatonic or chromatic scale by answering no or yes to the option of including sharps and flats. TI has included yet another means of increasing the level of difficulty–"Random Music." If chosen, random music is played between examples, making it more difficult to remember the preceeding note. The program provides up to ten examples and keeps score: 10 points for each correct answer.

We recommend that when first using this drill, you use C as the starting note, a range size of 13 (one octave), no sharps and flats, and no random music. After a little practice, it shouldn't be that difficult to identify notes.

Interval Recognition

This drill helps to develop your ability to recognize intervals. There are three levels, each of which adds more intervals to those included in the drill. For instance, if you choose Level 1, the examples are composed of major thirds, fourths, and fifths. Level 2 adds half steps, whole steps, and minor thirds, and Level 3 sixths, sevenths, and octaves. You can choose to have the intervals presented in ascending or descending order. For an added degree of difficulty, you can choose to have the lower note be random; it is otherwise C each time. You can also choose to have random music play between exercises. Up to ten examples are provided, and you receive 10 points for each correct answer.

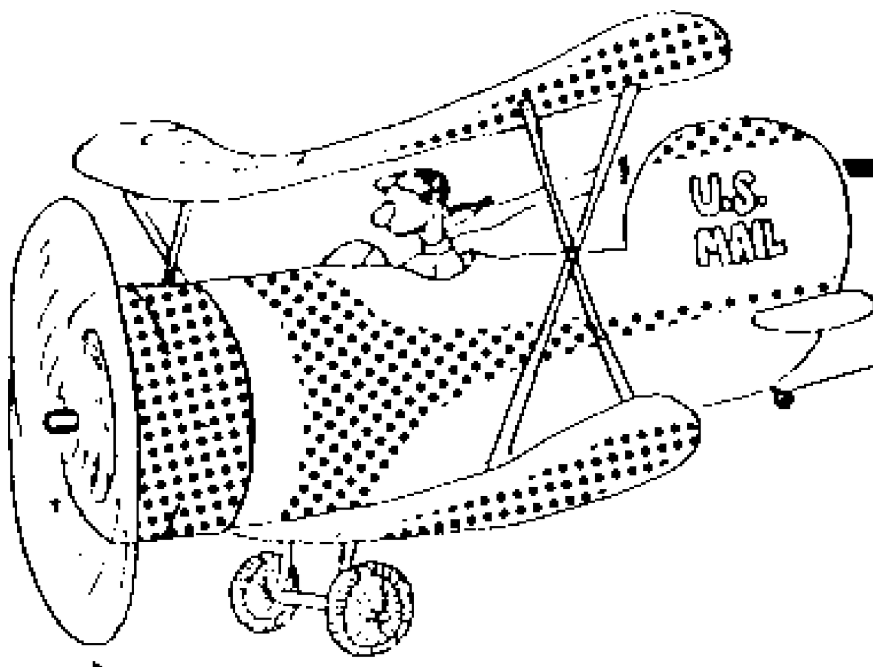
Chord Recognition

This drill provides practice in recognizing chords. Again there are three levels with Level 1 consisting of major and minor chords, Level 2 adding seventh and diminished, and Level 3 adding augmented. If you choose the "Random Bass" option, the root can be any note; otherwise it is a C. If you choose the "Random Inversions" option, inverted chords will be played, otherwise, a root-position chord is always played. If you choose the "Chord Only" option, the three notes will be played simultaneously. If you don't choose it, the notes comprising the chord are first played individually and then together. As in the previous drills, you can select the "Random Music" option. You receive 10 points for each of up to 10 problems.

Phrase Recall

This drill develops your ability to remember a sequence of up to nine random notes. A blank keyboard overlay, provided with the program, is used to label the keys with their corresponding pitch, covering two octaves much like the layout of a piano keyboard. You can select the starting note and range size, and determine whether sharps and flats are to be included in the examples. You can also specify the number of notes which constitute the phrase (1-9). After a phrase is played, you respond by entering notes from the keyboard as if it were a piano. Notes are heard and displayed as you play them, and if you make a mistake, you can use SHIFT T to start over again without penalty. When you have entered the notes that you think correctly represent the phrase, you press ENTER. The correct notes are then displayed below your response, and you are awarded points based on the number of correct notes and the number of notes included in the phrase. Up to ten examples are given with a possible total score of up to 100 points. As in the previous drills, the "Random Music" option can be chosen to make this drill even more difficult.

We feel that TI's *Music Skills Trainer* will be useful for the experienced musician who wants to keep his auditory skills sharp. We would also highly recommend it for the novice who is interested in further developing his knowledge and abilities in those areas of music theory covered by the program.



LETTERS TO THE EDITOR

Dear Sir:
I doubt this letter will ever reach your eyes, but if it does I just would like to extend to you my deepest appreciation for giving me the privilege of receiving, in my opinion, one of the best written, most organized, and thoroughly fascinating magazines introduced on the market. I was very impressed with all the articles and was especially interested in the one on the "Lamplighter" project. I also am thinking of purchasing the book "Mindstorms."

I am very excited about many of the things [in the articles] and I am looking forward to the July/August issue of 99'er Magazine.

Bill Finley
Raleigh, NC

Thanks for the kind words, Bill. Yours are typical of the many letters we received from fellow 99'ers. It's a pleasure to be serving you all.

Dear Sir:
In the article about Extended BASIC, CALL SAY is referred to as a TI BASIC command. Please inform me what this is since it doesn't appear in my user's guide.

Also, you should tell people that when they send away for software/hardware they read about in 99'er Magazine, they should tell the advertiser that they saw their ad. That will help you keep advertisers and gain new ones.

J. Douglas Wellington
Jersey City, NJ

CALL SAY as a TI BASIC command is explained in the instruction booklet that comes with TI's Speech Editor Command Module as well as the documentation for Extended BASIC. By the way, an excellent idea about mentioning the 99'er Magazine to advertisers, Douglas. We certainly hope all 99'ers take your advice.

Dear Sir:
I have just recently purchased Extended BASIC. The features that it offers are excellent. But I have been wondering for some time how to incorporate high resolution plotting into my scientific graphs. This feature would seem to me to be the first and most obvious extension to TI BASIC. However, I can't seem to find it. The SPRITE seems to be the only character to address the 256 by 192 high resolution dots. It is not immediately obvious to me whether this can be done as the sprites erase preceding sprites unless given a number, and 28 of them certainly cannot define a graph. TI's graphing package is of no help since it is too specific. The statistical package gives the type of graphing I would like but its method is a secret. Can you help?

Priscilla Walling
Darien, IL

An excellent question, Priscilla . . . and very timely as well. Watch for help with this in our next issue.

Dear Sir:
It is always a source of irritation for me to have to physically list the contents of each diskette in order to locate and then run a program. So I thought it would be neat if I could include a "LOAD" program with each diskette. This way, all I had to do was insert the diskette, crank up Extended Basic and select a program to run.

The only problem with this approach is that you must tailor each LOAD according to the contents of that particular diskette.

Then the idea came to me—A "General Purpose" LOAD! One that will scan the diskette record 0, pick out the programs, construct a "menu" and let me select the program I wish to run.

A great idea, right? The program to do this is listed on the enclosed page. It works exactly as envisioned, except for one small catch —line 34. Line 34 does not work at all. I tried every possible combination other than RUN "DSK1.PROGNAME" and (as the book says) this is the only way you can load and go.

I would have no quarrel with this were it not for the OPEN command (see line 12) and page 138 of the Extended BASIC Manual. Even though the format of both the OPEN and RUN and INPUT and PRINT are similar, TI has allowed the substitution of literals for device-filename in all but the RUN command. This is not only inconsistent (not the only inconsistency, however, as shown by the various forms of the DISPLAY USING, for example) but a tragedy to limit such a powerful command.

Oh well, so much for my good idea!

Charles Ehninger
Fort Worth, TX

Are there any readers out there in 99'er-land with an idea on how to make Charles' "General Purpose Load" a reality?

```
1 REM ******************
2 REM * GENERAL PURPOSE *
3 REM * PROGRAM LOADER  *
4 REM * BY C.M.EHNINGER *
5 REM ******************
6 OPTION BASE 1
7 DIM PG$(20)
8 IMAGE ##
9 CALL CLEAR
10 DISPLAY AT(12,6)ERASE ALL:"DISK? (1-3): 1";
11 ACCEPT AT(12,19)SIZE(-1)VALIDATE("123"):D$
12 OPEN #1:"DSK"&D$&".",INPUT ,RELATIVE,INTERNAL
13 INPUT #1:N$,A,A,A
14 DISPLAY AT(1,8)ERASE ALL:"DSK"&D$&" - "&N$;
15 I=0
16 FOR X=1 TO 20
17 I=I+1
18 IF I>127 THEN 36
19 INPUT #1:P$,A,B,B
20 IF LEN(P$)=0 THEN 26
21 IF ABS(A)<>5 THEN 19
22 DISPLAY AT(X+2,10):USING 8:X;
23 DISPLAY AT(X+2,14):P$;
24 PG$(X)=P$
25 NEXT X
26 DISPLAY AT(X+2,10):USING 8:X;
27 DISPLAY AT(X+2,14):"TERMINATE";
28 DISPLAY AT(X+3,14):"CHOICE? 1";
29 ACCEPT AT(X+3,22)SIZE(-2)VALIDATE(DIGIT):K
30 IF K=X THEN 35
31 IF K<1 OR K>20 THEN 28
32 IF LEN(PG$(K))=0 THEN 28
33 CLOSE #1
34 RUN "DSK"&D$&"."&PG$(K)
35 CALL CLEAR
36 END
```

GENERAL NOTES:
In the last issue, we featured an article on adding an external keyboard to the TI-99/4. This technique did require slight modifications to the console. Texas Instruments has informed us that any tampering with their hardware will void the warranty. We just want all our readers to be aware of this. *99'er Magazine* will continue to publish articles of this type for those readers who have the technical background, experience, and desire to expand the performance of their own equipment—Ed.

Index to Advertisers

American Software Design & Distribution Co. . . . 81
Anthistle Systems & Programming Ltd. . . . 90
The Bach Company . . . 92
Checks To-Go . . . 13
CBM Inc. . . . 34
Cumberland Technology . . . 85
Computer Shopper . . . 78
Data Systems . . . 23
Denali Data . . . 38
Dow Jones Information Services . . . 18
Dymarc Industries, Inc. . . . 80
Eastbench Computer Products . . . 85
Ehninger Associates, Inc. . . . 23
Electronic Specialists, Inc. . . . 23
Epson America, Inc. . . . 7
Extended Software Co. . . . 50
Eyring Research Institute, Inc. . . . 50
Fantasy Computing . . . 31
Florida 99/4 User's Group . . . 91
Houston Instrument . . . 55
Instant Software . . . 80
Kalglo Electronics Co. Inc. . . . 42
Kemp Software . . . 42
Letcher Offshore Design . . . 31
Linear Aesthetic Systems . . . 91
Marinchip Systems, Ltd. . . . 15
Micro-Ed, Inc. . . . 83
The Micro House . . . 31
Pablo Diablo . . . 85
Paul Yates Computer Services . . . 81
Pike Creek Computer Co. Inc. . . . 85
RCL Computers . . . 88
RKS Enterprises, Inc. . . . 44
Small Business/Professional Computer Services . . . 14
Software Exchange Group . . . 61
E. D. Schwarz, M. D. . . . 34
Sunshine Software . . . 91
TAM'S Inc. . . . 14
Texas Instruments, Inc. . . . 2
TI More, Inc. . . . 45
W. R. Wilson Co. . . . 50
99'er Bookstore . . . 66, 67, 68, 69
99'er Magazine . . . 86, 93
99'er-Ware . . . 3, 42, 95, 96