Guest Commentary: Managing Technostress

COMPUTE!

$2.50
May, 1982
Issue 24
Vol. 4, No. 5
63379

The Journal For Progressive Computing™

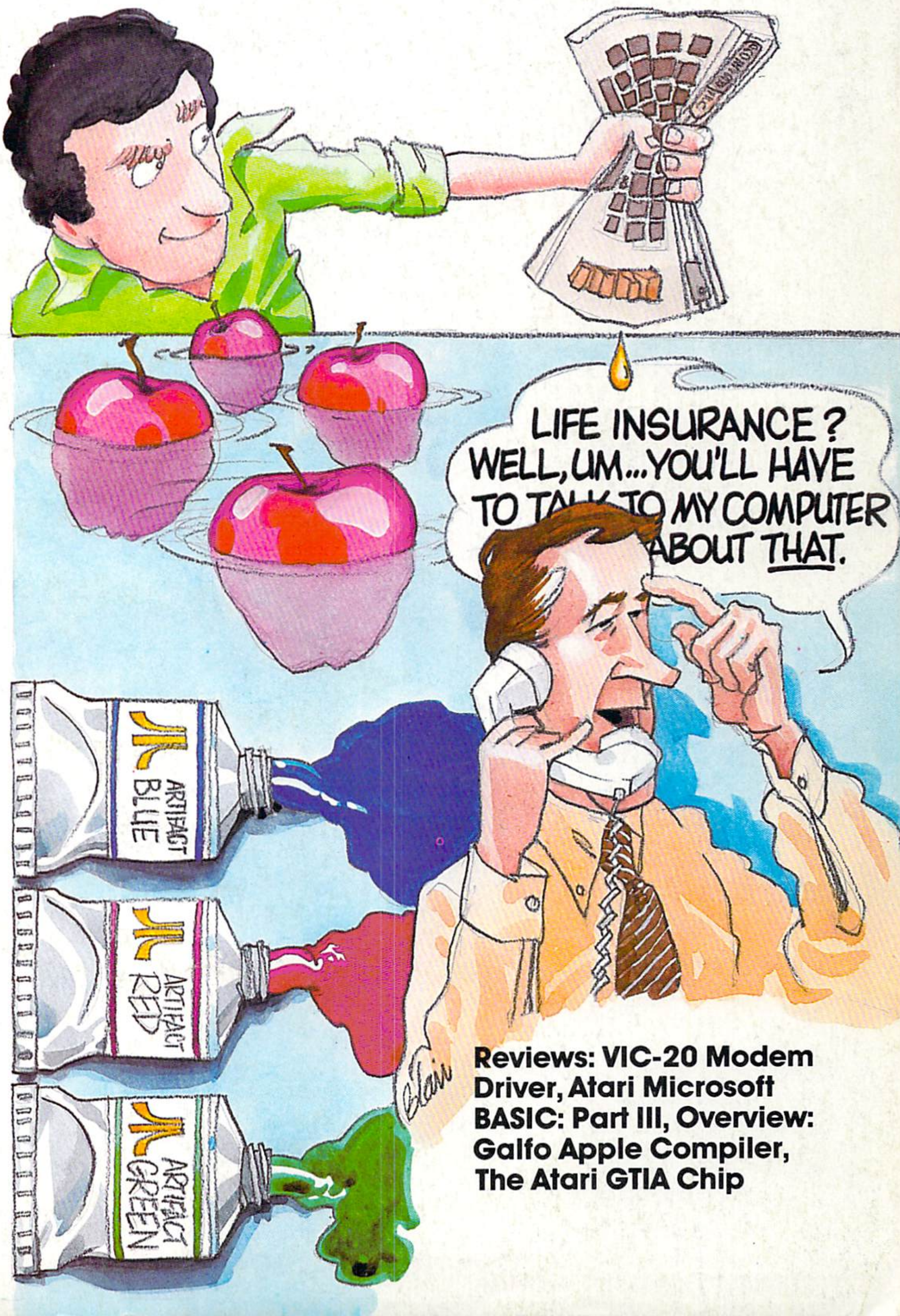
Putting The Squeeze On Your VIC-20:
Getting The Most Out Of 5000 Bytes...

Modifying Apple's Floating Point BASIC

Life Insurance Estimator:
An Applications Program

Extra Colors For Atari Through Artifacting

Screen Input For Commodore PET/CBM



Reviews: VIC-20 Modem Driver, Atari Microsoft BASIC: Part III, Overview: Galfo Apple Compiler, The Atari GTIA Chip

IF YOU'RE WAITING FOR THE PRICE OF WORD PROCESSORS TO FALL WITHIN REASON,

Everyone expected it would happen sooner or later... with WordPro PLUS™ it already has! Now all the marvelous benefits of expensive and advanced word processing systems are available on Commodore computers, America's largest selling computer line. WordPro PLUS, when combined with the new 80 column CBM 8032, creates a word processing system comparable to virtually any other top quality word processor available—but at savings of thousands of dollars!

New, low cost computer technology is now available at a fraction of what you would expect to pay. This technology allowed Commodore to introduce the new and revolutionary CBM 8032 Computer.

WordPro PLUS turns this new CBM 8032 Computer into a sophisticated, time saving word processing tool. With WordPro PLUS, documents are displayed on the computer's screen. Editing and last minute revisions are simple and easy. No more lengthy re-typing sessions. Letters and documents are easily re-called from memory storage for editing or printing with final drafts printed perfectly at over five hundred words per minute!

Our nationwide team of professional dealers will show you how your office will benefit by using WordPro PLUS. At a price far less than you realize.

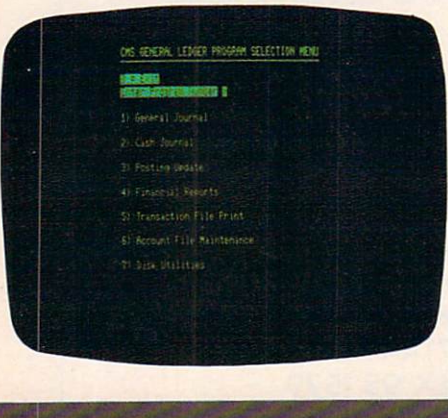
Invest in your office's future...
Invest in WordPro PLUS...
Call us today for the name of the WordPro PLUS dealer nearest you.

Professional Software Inc.
166 Crescent Road
Needham, MA 02194
(617) 444-5224
TELEX: 95 1579

CMS Softwa General Accou

For the Commodore 8000

Now there is an interactive General Accounting System designed especially for the first time user. The user is led through each function by highlighted prompts explaining the required input at each point. And all input requests are prompted with complete verification. Plus, a step-by-step instruction manual guides the user through the program. The user prompts and the detailed manual make it virtually impossible for the first time user to get lost or to accidentally crash.



General Ledger

Cash Disbursements and Cash Receipts Journal as well as a General Journal. Maintains account balances for Present Month, Quarter to Date and Year to Date. Accepts postings from Accounts Receivable, Accounts Payable, Payroll and other programs. Also provides user customized financial statements. Up to 1000 accounts and up to 99 departments.

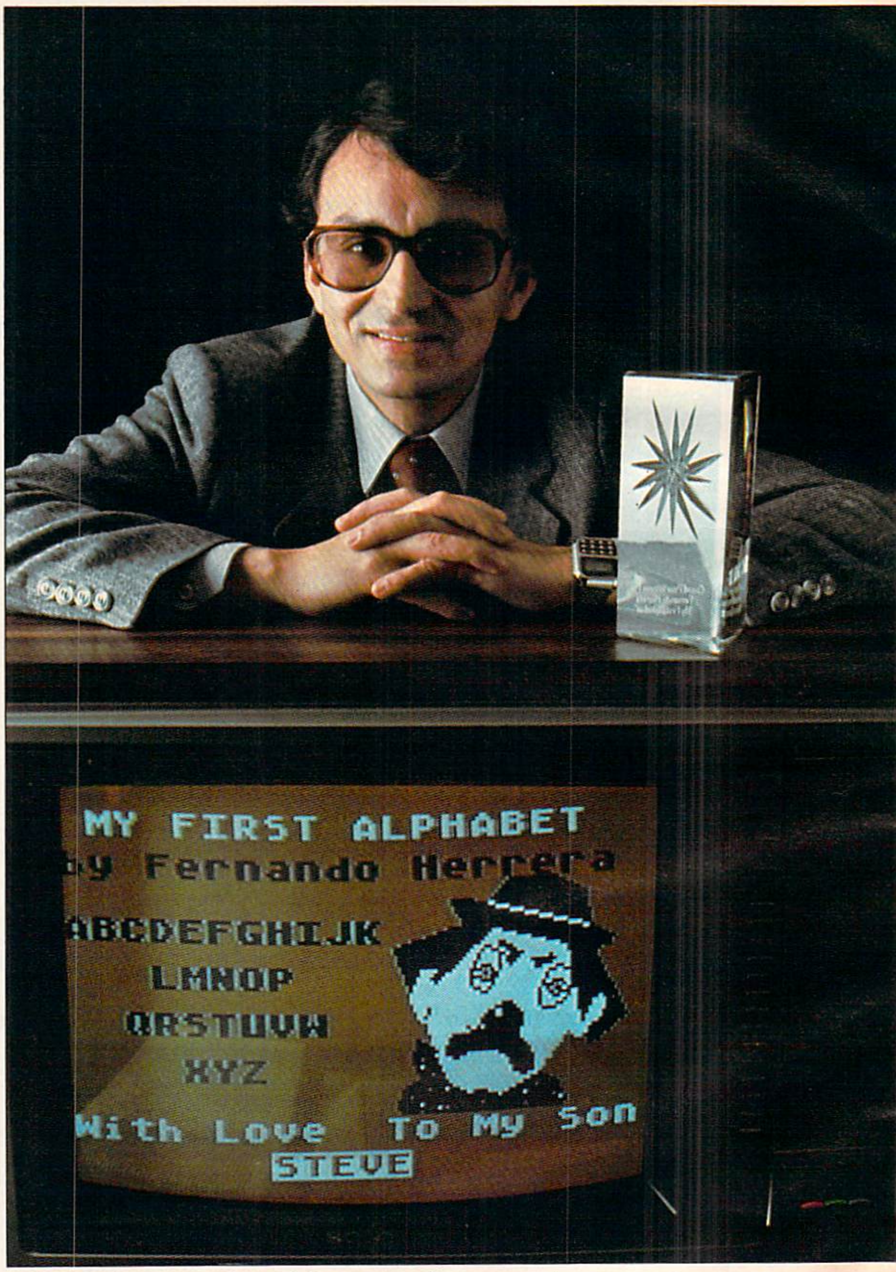
Table of Contents

May, 1982 Vol. 4, No. 5

AP = Apple, **AT** = Atari, **P** = PET/CBM, **V** = VIC-20, **O** = OSI, **C** = Radio Shack Color Computer, * = All or several of the above.

COMPUTE! The Journal for Progressive Computing (USPS: 537250) is published 12 times each year by Small System Services, Inc., P.O. Box 5406, Greensboro, NC 27403 USA. Phone: (919) 275-9809. Editorial Offices are located at 625 Fulton Street, Greensboro, NC 27403. Domestic Subscriptions: 12 issues, $20.00. Send subscription orders or change of address (P.O. form 3579) to Circulation Dept., **COMPUTE!** Magazine, 515 Abbott Drive, Broomall, PA 19008. Controlled circulation postage paid at Greensboro, NC 27403 and additional mailing offices.
ISSN 0194-357X.

AN ATARI 800™ HOME COMPUTER AND A FATHER'S LOVE COMBINED TO HELP CHILDREN EVERYWHERE.



Fernando Herrera became the first grand prize winner of the ATARI Software Acquisition Program (ASAP) competition because he believed in computers, his son and himself.

The story of Herrera's success began with his son's sight problems. Young Steve Herrera had been born with severe cataracts in both eyes and, naturally, his father was concerned. Herrera reasoned that the boy's learning abilities could be seriously affected by growing up in a world he could not see.

Having just purchased an ATARI 800 Home Computer, it occured to Herrera that this could be the perfect tool for testing Steve's vision. So he wrote a program simply displaying the letter "E" in various sizes.

Success! It turned out that 2-year-old Steve could see even the smaller "E's" without special lenses. Herrera was first relieved, and then intrigued when he discovered that not only could his son see the "E's," but he would happily play with the computer-generated letters for hours. So Herrera added a picture of an elephant to go with the "E," and then more letters and pictures. Thus, "My First Alphabet" was born, a unique teaching program for children two-years and older consisting of 36 high resolution pictures of letters and numbers.

Herrera submitted the program to the ATARI Program Exchange, where it became an instant best-seller. ATARI was so impressed with the outstanding design, suitability and graphic appeal of "My First Alphabet," that the program is being incorporated into the ATARI line of software.

In addition to his grand prize winnings of $25,000 in cash and an ATARI STAR trophy, Herrera also automatically receives royalties from sales of his program through the ATARI Program Exchange.

But Fernando Herrera wasn't the only software "star" that ATARI discovered. Three other ATARI STARS were awarded at the ASAP awards ceremony for software submitted to the ATARI Program Exchange and judged by ATARI to be particularly unique and outstanding.

Ron and Lynn Marcuse of Freehold, New Jersey, teamed up to write three winning entries in the Business and Professional category for home computers: "Data Management System," "The Diskette Librarian" and "The Weekly Planner."

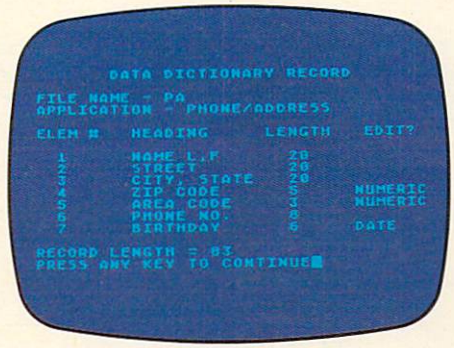
Sheldon Leeman of Oak Park, Michigan, captured an ATARI STAR for his exceptionally well-engineered "INSTEDIT" character set editor.

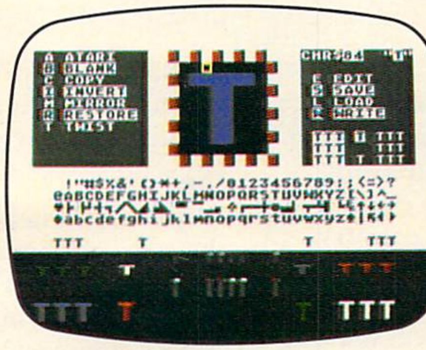
Greg Christensen of Anaheim, California, became our youngest ATARI STAR winner at the age of 17. Christensen designed the clever "Caverns of Mars" game program, which also will be incorporated into the ATARI product line. Greg designed the program in 1½ months after owning his ATARI Home Computer for less than a year.

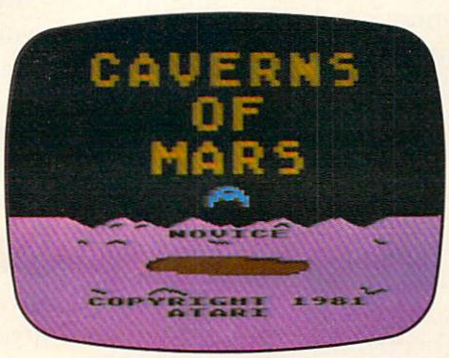
Every three months, ATARI awards ATARI STARS to the writers of software programs submitted to the ATARI Software Acquisition Program and judged first, second and third place in the following categories: Consumer (including entertainment, personal interest and development); Education; Business and Professional programs for the home (personal finance and record keeping); and System Software.

Quarterly prizes consist of selected ATARI products worth up to $3,000, as well as an ATARI STAR, plus royalties from program sales through the ATARI Program Exchange. The annual grand prize is the coveted ATARI STAR trophy and $25,000 in cash.

To be eligible, your software idea must be accepted by the ATARI Software Acquisition Program. Your program can have a broad application or serve a very specific purpose.







After submittal, consultation from ATARI is available if you need personal assistance with sound, graphics, or other technical aspects of your program.

To make your job easier, ATARI provides some 20 software development tools through the ATARI Program Exchange. A list and description of the various system software is published quarterly in the ATARI Program Exchange Catalog. These tools enable you to utilize all the ATARI resources and software, including the six ATARI programming languages.

Fernando Herrera had a great idea that made him a star. ATARI would like to give you the same opportunity.

ATARI®

A Warner Communications Company

Enter the ATARI ASAP competition and you could win $25,000 in cash, royalties, some great prizes and an ATARI STAR.

SUBMIT TO: ATARI® Software Acquisition Program
Dept. C4R, P.O. Box 427
155 Moffett Park Dr., B-1
Sunnyvale, CA 94086

OR CALL: 800-538-1862; in California, 800-672-1850.

I'm reaching for the stars. Please send me an entry form today.

Name____________________

Address____________________

City____________________

State/Zip____________________

ATARI HOME COMPUTERS

We've Brought The Computer Age Home™

The Editor's notes...

Robert C. Lock
Publisher/Editor

The Apple Plan

Many of you may not be aware that Apple, Inc., is currently pushing for a revision of the tax law that will allow them to give an Apple II system (with appropriately revised tax deductions) to every elementary and secondary school in the country. *The Wall Street Journal* estimates this number to be in excess of 80,000 schools, and that's a great many Apple II's. We applaud this move, but feel some additional perspective might be in order. Apple, Inc., is quoted as wishing to make this massive donation because they're concerned with the future ... exposing kids to computers, etc. We certainly share this concern and wouldn't want to be accused of looking a gift computer in the keyboard, as it were.

A Historical Perspective

Apple has been an industry leader in the placement of computer systems in school systems. They and many Apple dealers have been at the forefront with aggressive bidding and state contract winning ways for several years. Their principal competition during the last year or so has been Radio Shack, Commodore, and Atari, with Atari coming on quite strong. A case in point? The Minnesota Educational Consortium has been a quantity purchaser (and not so coincidentally, a significant developer of educational software) from and for Apple for the last three years. We reported in this column last November that Atari, Inc. had been added to that contract with fully configured Atari 400 systems (including black and white monitor, BASIC, joystick and disk drive) for less than $600. MEEC had already ordered 1000 +, and the Dade County School System in Florida was right behind with a similar contract.

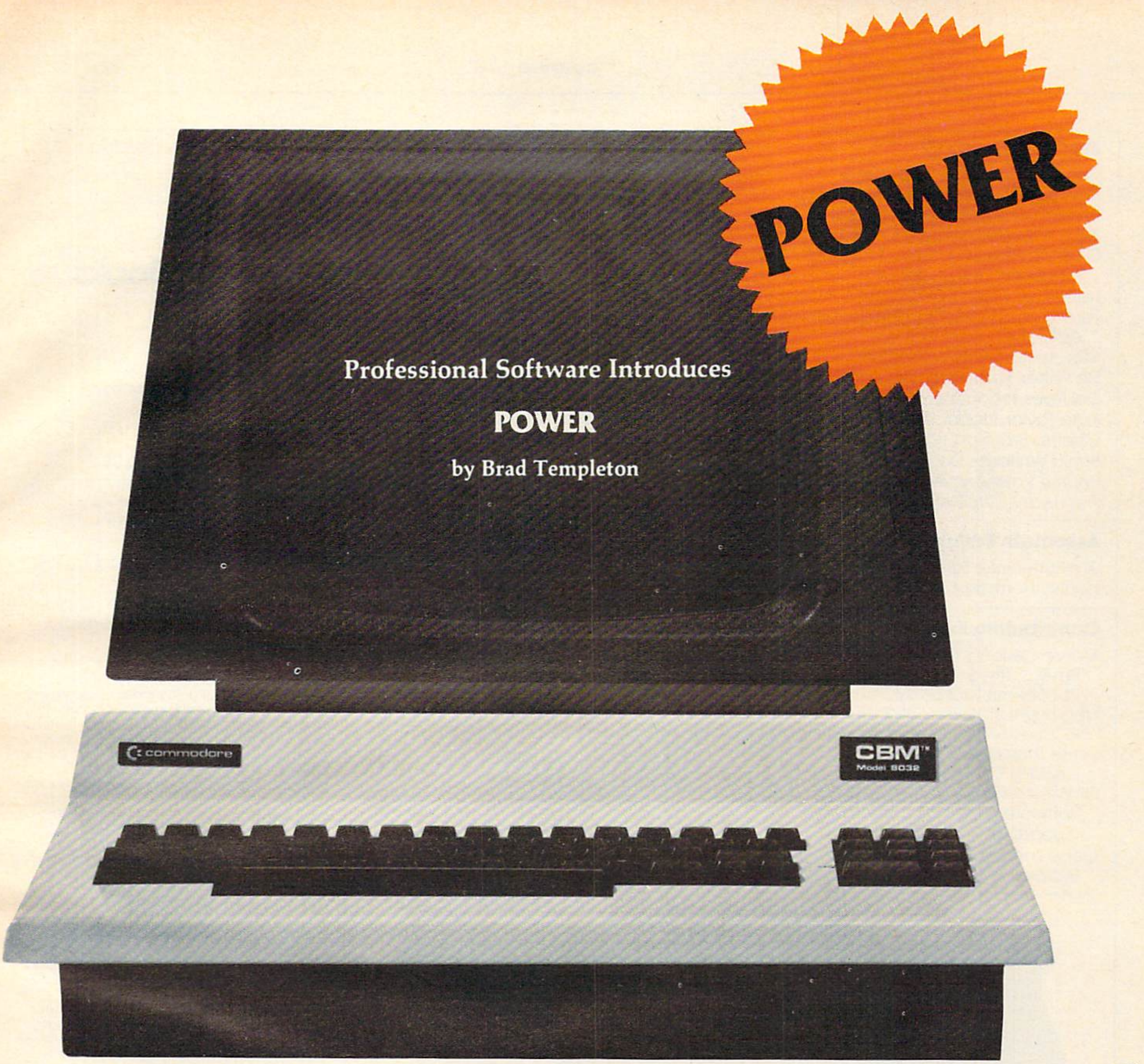
In the last ten months we've seen increasingly powerful and user friendly computer systems (the Atari 400, the VIC-20, the Radio Shack Color Computer) become available, fully configured, in the $500-$800 price range. We suspect that Apple is running into a rapidly diminishing market share at the elementary and secondary level.

The Future

We see nothing but good news on the horizon for the cost-conscious educators. With the coming Super VIC (the VIC-64 from Commodore, due to be introduced this summer at less than $600) and Atari's rumored low-mid-end entry sometime this summer, the value of the computing power dollar for educators will only increase. Apple has apparently chosen to ignore the low-end market in their product line. With this statement we're certainly not implying that they don't want the business of consumers or educators. We simply mean they've chosen for the moment not to enter the $500-$800 consumer computer market. On the high end they're battling with such industry names as Xerox, IBM, and Hewlett-Packard. And, quite frankly, they have no product on the low end. We suspect the secondary education market, long a potentially substantial market for Apple and others, is taking a hard look at the $2000 or more Apple system. After all, when your goal is allowing computer access to your sixth graders, how many VIC-20, or Atari-400, or Radio Shack Color Computer systems can you buy for the cost of a similarly configured Apple II system?

Of Tax Breaks and Competition

Press reports indicate no special treatment for Apple, Inc., in the proposed revisions to the corporate tax law. At that point, every other vendor has the same options open to them. In that light, we support Apple's aggressive offer, and the revision of the tax law, as suggested. After all, manufacturers can already donate hardware to universities, why not to secondary schools? We do feel, however, that while Apple's move deserves plaudits, the spectre of competition should be maintained to help balance our mutual perspective. ©



ADD POWER™ TO YOUR COMMODORE COMPUTER

$89.95

POWER produces a dramatic improvement in the ease of editing BASIC on Commodore's computers. POWER is a programmer's utility package (in a 4K ROM) that contains a series of new commands and utilities which are added to the Screen Editor and the BASIC Interpreter. Designed for the CBM BASIC user, POWER contains special editing, programming, and software debugging tools not found in any other microcomputer BASIC. POWER is easy to use and is sold complete with a full operator's manual written by Jim Butterfield.

POWER's special keyboard 'instant action' features and additional commands make up for, and go beyond the limitations of CBM BASIC. The added features include auto line numbering, tracing, single stepping through programs, line renumbering, and definition of keys as BASIC keywords. POWER even includes new "stick-on" keycap labels. The cursor movement keys are enhanced by the addition of auto-repeat and text searching functions are added to help ease program modification. Cursor UP and cursor DOWN produce **previous** and next lines of source code. COMPLETE BASIC program listings in memory can be displayed on the screen and scrolled in either direction. POWER is a must for every serious CBM user.

Call us today, for the name of the Professional Software dealer nearest you.

Professional Software Inc.
166 Crescent Road
Needham, MA 02194
Tel: (617) 444-5224 Telex #951579

TM POWER is a Registered Trademark of Professional Software, Inc.

Staff of COMPUTE!

Robert C. Lock, Publisher/Editor
Kathleen Martinek, Managing Editor
Richard Mansfield, Assistant Editor
Alice S. Wolfe, Publisher's Assistant
Georgia Papadopoulos, Art Director/ Production Manager
Harry Blair, Advertising Sales Manager
Terry Cash, Typesetting/Production Assistant
Margret Jackson, Typesetting
De Potter, Production Assistant
Dai Rees, Production Assistant
Kate Taylor, Production Assistant
Charles Brannon, Editorial Assistant
Sonja Whitesell, Dealer Coordinator
Bonnie Valentino, Accounting Coordinator
Fran Lyons, Circulation Coordinator

Associate Editors

Jim Butterfield, Toronto, Canada
Harvey Herman, Greensboro, NC

Contributing Editors

Marvin DeJong, Dept. of Mathematics - Physics, The School of the Ozarks Pt. Lookout, MO 65726
Fred D'Ignazio, c/o COMPUTE! P.O. Box 5406, Greensboro, NC 27403
David Thornburg, P.O. Box 1317, Los Altos, CA 94022
Bill Wilkinson, Optimized Systems Software, 10379-C Lansdale, Ave. Cupertino, CA 95014
Gene Zumchak, 1700 Niagara St., Buffalo, NY 14207

Subscription Information

COMPUTE! Circulation Dept.
515 Abbott Drive
Broomall, PA 19008 USA

TOLL FREE
Subscription
Order Line
800-345-8112
In PA 800-662-2444

COMPUTE! Subscription Rates

(12 Issue Year):

US	(one yr.)	$20
	(two yrs.)	$36
	(three yrs.)	$54
Canada and Foreign Surface Mail		$25
Air		
Europe, US Possessions		$38
Middle East, Central America and North Africa		$48
South America, South Africa, Far East and Australia		$88

Small System Services, Inc. publishes:

COMPUTE!
The Journal For Progressive Computing

COMPUTE! Books

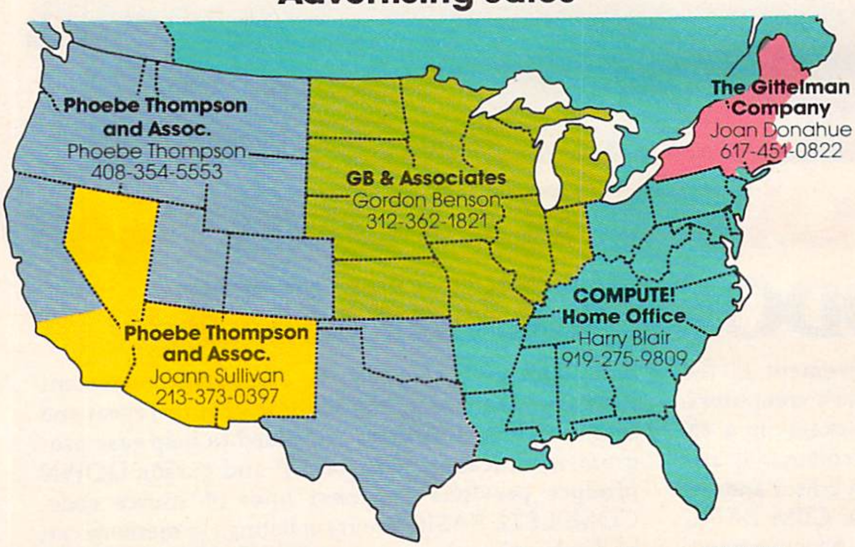
Corporate office:
625 Fulton Street,
Greensboro, NC 27403 USA

Mailing address: COMPUTE!
Post Office Box 5406
Greensboro, NC 27403 USA

Telephone: 919-275-9809

Robert C. Lock, President
W. Jerry Day, Vice-President, Comptroller
Kathleen Martinek, Assistant To The President
Ellen Day, Accounting Manager
Carol Holmquist Lock, Research Assistant

Advertising Sales



Phoebe Thompson and Associates
101 Church Street, Suite 13
Los Gatos, CA 95030
PHOEBE THOMPSON

Phoebe Thompson and Associates
P.O. Box 7000-462
Redondo Beach, CA 90277
JOANN SULLIVAN

GB & Associates
P.O. Box 335
Libertyville, IL 60048
GORDON BENSON

COMPUTE! Home Office
625 Fulton Street
Greensboro, NC 27403
HARRY BLAIR

The Gittelman Company
Statler Office Building
Suite 518
20 Providence Street
Boston, MA 02116
JOAN DONAHUE

In British Columbia, Canada call 408-354-5553.

Elsewhere in Canada or outside North America call 919-275-9809.

In Alaska or Hawaii call 408-354-5553.

Address all advertising materials to:
Alice S. Wolfe
Advertising Production Coordinator
COMPUTE! Magazine
625 Fulton Street
Greensboro, NC 27403 USA

Coming In The June Issue Of COMPUTE!

Income Property Report

A Self-Modifying P/M Graphics Utility

Apple Manager

VIC/PET BASIC Program Transfers

Outpost – A Strategy Game

JINSAM™

OFFICIALLY APPROVED BY COMMODORE

Data Manager selected by NASA, Kennedy Space Center

SAVE TIME! SAVE MONEY!
JINSAM saves taxpayers 100's of thousands of $

Coming Soon!
JINSAM Executive for IBM P.C.
See your dealer

"Much more powerful than you can imagine!"
Robert Baker
Kilobaud Microcomputing

JINSAM is an integrated system. It makes it easy to use your information to its fullest. No more will hundreds of valuable hours be spent searching or analyzing needed information nor re-entering information for various reports.

JINSAM transforms your desk-top computer into the "state of the art" data processing machine with features and accessories found nowhere, even at 10 times the price. NASA, Kennedy Space Center selected JINSAM 8.0 and saved approximately $95,000 over other software/hardware costs. Riley County, Kansas also selected JINSAM 8.0 and saved approximately $90,000 over other software/hardware costs.

JINSAM is designed for you. It is forgiving. It has help commands for every option, available at the touch of a button. The amount of information you store, its structure and/or your hardware can change but your data won't have to be re-entered. Recovery utilites are included even for catastrophes, security passwords are built in for privacy, simple editing and entry includes auto recall, and deleting records is easy and the space is reclaimed. JINSAM includes TWO FREE accessories for reports and labels. You have unlimited report formats with summing and lined up decimals and the label printer prints up to 5 across – any size and even allows single envelopes or index cards.

JINSAM 1.0 allows fast and easy file handling, manipulation and report generation for any CBM computer with CBM 2040 disk drive. It features a menu for ease, has encrypted passwords, 3 deep sorts, .5 to 3 second recall.

JINSAM 4.0 for CBM 4000 series adds JINSORT, user accessible machine sort of 1000 records in 15 seconds; compaction/expansion of information, automatic list maintainance, unlimited number of fields, unlimited record length and much more.

JINSAM 8.0 for CBM 8000 series has all 4.0 features plus unlimited sort, horizontal format, and search by key or record number.

JINSAM 8.2, NEW FOR '82 expands 8.0 capabilities by adding information search by word, key or record number and machine language print, format and manipulation routines.

- ★ CUSTOM DATA FILES
- ★ CUSTOM REPORTS/LABELS
- ★ KEYED RANDOM ACCESS
- ★ FAST/EASY/MENU DRIVEN
- ★ MULTIPLE SEARCH KEYS
- ★ PRIVACY ACCESS CODES
- ★ WILD CARD SEARCH

JINSAM EXECUTIVE version (soon to be released) is our most powerful professional system for the CBM 8000 and 9000 series. Executive will have 8.2 extended features plus allow multiple users with in-use lockout protection, executive command files, automatic math relations, join, merge or link files, greatly increased record capacity and machine information search by word, as well as by key or record number and many, many more features.

There are currently 7 more interfacing modules – and more under development, including independent interfaces between JINSAM and business packages for your G/L, A/R, A/P needs. We announce the availability of modules and enhancements in JINSAM's quarterly newsletter.

WORDPROPACK - Intelligent interface for WordPro 3, 3+, 4, 4+, creates lists of information from JINSAM files. It allows up to 10 conditions based on each item of information. Produce individualized letters, report cards, special reports, checks, invoices, etc.

MULTI-LABEL - Prints multiple labels per record with up to 2 lines for messages and consecutive numbering. Produce inventory, caution labels, bulk mail labels, etc.

MATHPACK - global calculator/editor +, −, *, /, by another field or constant; null (remove contents) of a field or replace contents of a field with any word, number or phrase. Sum multiple fields in each record or running sum of single field in all records. Extract information or effect permanent change. Replace in the same field or place in a waiting field.

DESCRIPTIVE STATPACK - Determine MEAN, MEDIAN, MODE, STANDARD DEVIATION, VARIANCE, RANGE. Generate HISTOGRAMS from 1 to 25 steps, and produce Z-SCORE reports.

ADVANCED STATPACK - (You must also acquire DESCRIPTIVE STATPACK) Generate one, two or three way CROSSTABS (number of occurance) CHI SQUARE, LINEAR REGRESSION, with graphic representation and prediction, LINEAR CORRELATION and SIMPLE ANALYSIS OF VARIANCE.

CALCPACK - 2 way interface to VisiCalc or any user program. It lets you use VisiCalc for complex manipulation, editing, placing results in JINSAM for sorting, storing or moving data to WordPro as well as giving the ability for exchange with your own applications.

INTERAC - Interface which can read VisiCalc files, WordPro files and almost any sequential files to build JINSAM databases automatically. For example: You could "download" information on holerith cards to sequential files and INTERAC would place them into JINSAM files.

All accessories are accessed thru the JINSAM menu and require security password to gain entrance.

JINSAM gives you FREEDOM OF CHOICE. Start with JINSAM 1.0 and upgrade hardware and data at any time. Choose from accessories at any time. The JINSAM Newsletter brings the latest updates, user input and uses and keeps an eye on the future.

JINSAM stands alone by placing "a lot of potential computing power in one integrated program package" (Fred Klein, Ferson div. of Bausch & Lomb). "The JINSAM package is justification for buying a system no matter what the hardware, be it Vector or Commodore or whatever the system" (Larry Colvin, Micro Computer Systems). It is a "grandfather" in this young field. JINSAM EXECUTIVE will be the third generation in development. All JINSAM systems are sophisticated and flexible yet easy to use. JINSAM is saving its users valuable time and money in government, educational and research institutions, business and industry worldwide.

JINSAM is a Commodore approved product. See your local dealer for a demonstration.

JINSAM Data Manager

.... Additional Information and nearest dealer
.... Newsletter Subscription ($5 US/$8 Worldwide)
.... JINSAM Demo Disk ($15, plus shipping & tax)
.... User's Guide ($40.00, plus shipping & tax)

Please send to:

Name ______________________

Position ______________________

Company ______________________

Address ______________________

City, State, Zip ______________________

Phone (   ) ______________________

Computer, Disk ______________________

Application ______________________

JINI MICRO-SYSTEMS, Inc.
Box 274C • Kingsbridge Station
Riverdale, New York 10463
(212) 796-6200

Dealer Inquiry Welcome

Ask The Readers

Robert Lock, Richard Mansfield
And Readers

Please address any questions or answers to: Ask The Readers, **COMPUTE!** *Magazine, P.O. Box 5406, Greensboro, NC 27403.*

Answers

Mr. Rovero, whose article, "A User-defined Character Editor," appeared in **COMPUTE!**, February, 1982, #21, has updated the program to run on all PET/CBM machines. He offers to make copies for interested readers:

"I have since modified the program to work with all ROMs 1.0 through 4.0, 16K-32K (8K with REMARKS deleted), 40 or 80 column screens and with either tape or disk (Update and 4.0 ROMs). Readers may obtain a copy of the original program or the 'All ROM' version on tape or 4040 format disk by providing the magnetic media, self addressed mailer, and $3.00"

P. J. Rovero
Navy Oceanography Command Center
Comnavmarianas Box 2
FPO San Francisco 96630

"I am writing about the Commodore disk drive, upgraded 2040 or the 4040. It seems like when I had a program running relative files, the disk would write to two or more files at the same time. I would write information to file 'A' and then later pull up file 'L' and it would contain the same information as does file 'A.' I went over and over my programs, and could not find anything that looked out of order.

Last week, a customer of mine who uses a custom program I wrote for him called and said his accounts were all messed up. He had the same problem of writing to two different accounts at the same time.

I took his disks and ran a printout of the directory, Track 18, Sectors 0-19 and I found the problem. It seems that the directory has been messed up and is pointing to the same data for both accounts.

If you look at bits 4 and 5 of each directory entry they will tell you the track and sector that each entry's data is stored at. Bit 4 points to the Track, and bit 5 points to the Sector. If for example, the bits 4 and 5 look like this 0E 01 then you should look at Track 14 Sector 1 for your data. If bit 4 and 5 of the directory entries are the same, then, when you write to any one of these files, they will all look as if they have been written to. The problem is that the directory tells them all to look at the same data.

I just got a call from my dealer, who I convinced to call Commodore. That he did, and the answer he got was that they had a list of bugs on the 2040 and 4040 operating system and it seems this is one of them. He said if you have too many files on a disk that it may start to overwrite the directory. He didn't give any idea of what was too many....

I am going to rewrite my customer's program and all his data disks for 101 files maximum (down from 141) and see if this won't help. In the meantime maybe this information will help someone else who is having similar problems.

Ronald L. Straley

"In **COMPUTE!**, *February, 1982, #21, Michael A. Ivins wanted a program for the Atari that would dump text screens to a printer or file. He hoped for DATA statement creation for use in POSITION and PRINT statements in an adventure game.*

Computer's Voice is currently marketing a program called Menumakr that will complete this application. This program currently allows the user to type any screen in GRAPHICS 0 with full key editing. The program then dumps the screen to a tape or disk file making POSITION and PRINT statements. By including a different character set with the program, screen creation with that new set is very easy. Cost is $14.95 for the 8K tape version and $19.95 for the 16K disk version. We are currently working on a GRAPHICS 1 and 2 version as well.

Dave Pettit, President
Computer's Voice
2370 Ella Dr.
Flint, MI 48504

We received several replies to Don Dudley's question about problems with his VIC-20 tape drive. Here are two, but they both wisely suggest that you might want to leave adjustments to the experts.

"In reply to Don Dudley's question regarding his cassette player. The most likely cause of his problem is that the read/write head is out of alignment. I suggest using a plastic screwdriver, to keep from magnetizing the head, this can be purchased from any electronics supply store for less than a buck. With the player running, and the computer trying to read it, you will see a small screw on either the right or left side of the head. It is necessary to turn this screw, clockwise or counter-clockwise, until the computer begins to read the tape. Be careful not to remove the screw during the process, it's a real pain to get it back in again!

If you feel at all uncomfortable with the idea, any electronics repair store can align it in just a few moments, and usually for under $25." Mike Lipay

Hayes Stack™

Microcomputer Component Systems

Right for the time. Finally someone invented an RS-232C compatible calendar/clock system, complete with 6-digit display... and selling for only $249. Hayes did it!

Introducing the Hayes Stack Chronograph, the newest addition to the Hayes Stack microcomputer component series. It allows your computer to accurately record all of your system activities by date and time... down to the second. Thanks to a battery back-up system, you never have to reset the time when your computer is off, and it will keep on ticking even when there's a power failure. A write-protect switch prevents accidental changing of day, date or time.

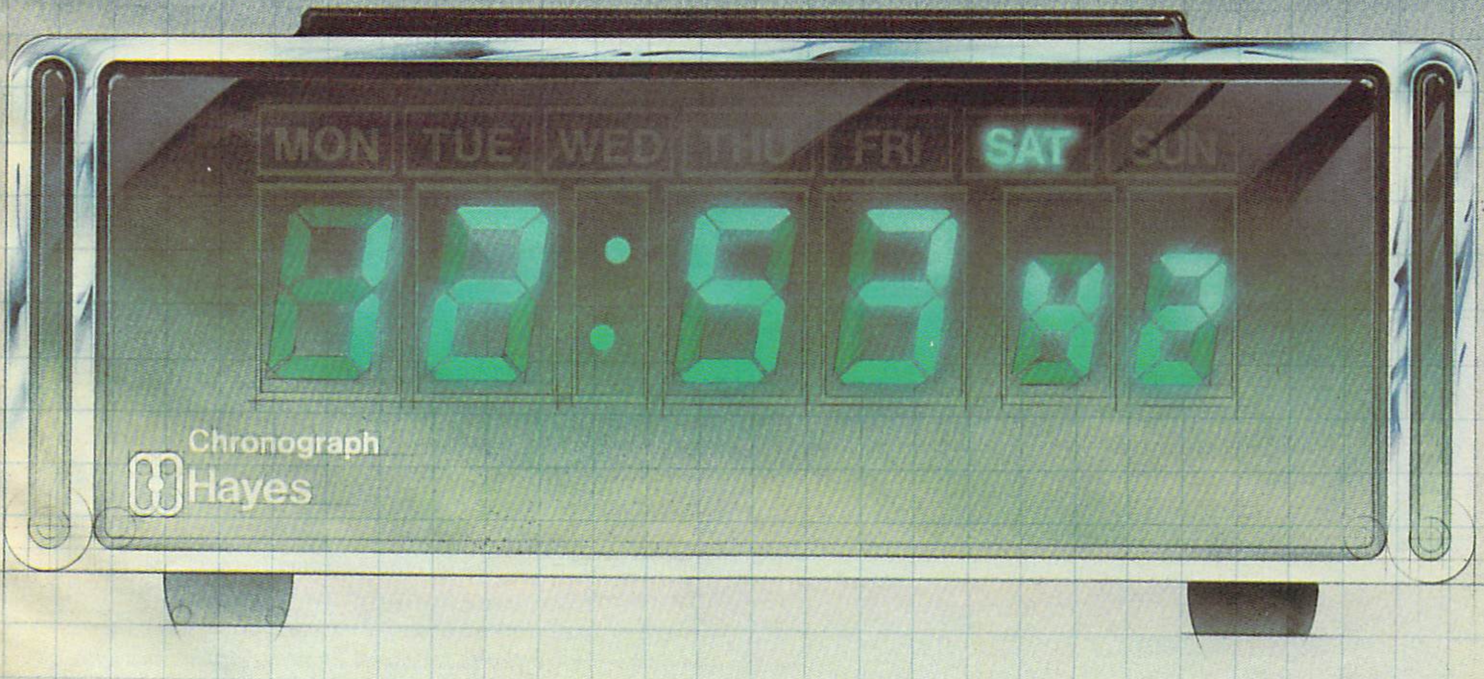
Right for the job. The Hayes Stack Chronograph is ideal for any home or business application requiring accurate timekeeping. Use it for timing everything from lights, burglar alarms, or sprinkler systems ...to sending mail electronically (with the Hayes Stack auto-dial Smartmodem and your computer) ...logging and recording reports or time-sharing access time... and batching all your messages to send at night, when rates are lowest. Chronograph helps do it all.

And Chronograph stacks up. Keep your computer system up-to-date with the Hayes stackables, including the RS-232C compatible Smartmodem, the most sophisticated 300-baud originate/answer modem you can buy. And yet, it's probably the easiest to use too.

The Hayes Stack Chronograph and Smartmodem are available wherever fine computer products are sold. It's time. And it's now.

Hayes™

The Hayes Stack™ Chronograph. There's no better time.



Hayes Microcomputer Products, Inc. 5835 Peachtree Corners East, Norcross, Georgia 30092 404/449-8791

"In response to Don Dudley's letter (March '82) about his VIC recorder that only loads when upside down, I had a similar problem with mine. A week after I bought it, I tried loading tapes made by some friends, with little success. The dealer exchanged it, since it wouldn't load one of his demo tapes.

The replacement still wouldn't load one friend's tape until by chance I pushed down a bit on the recorder's door, effectively changing the azimuth relationship of the tape and the playback head. Another friend had a similar problem until he bent the tension spring in the back of the cassette compartment forward far enough to increase the tape to head pressure. If neither of these methods appeal to you, a trip to the dealer for a head alignment may be required. Good luck!" John Williamson

---

The example program which illustrated the technique described in this letter was left out when it was first published in this department in **COMPUTE!**, March, 1982, #22. We reprint it here in its entirety.

"For several years I have been dealing with the crash of the INPUT statement in our 2001 and 4016 Commodore Computers when the RETURN key is depressed with no input! I have read of several fairly short routines that overcome this problem and have developed my own favorite, as have most users. (Our students develop software for the Elementary School – kids, and their teachers, will *crash programs.)*

Anyway – this evening, quite by accident, I stumbled upon a quick way to protect the RETURN key on INPUT! At least I have never heard of it. [Lines 70, 80, 160, 170, and 180 will do.] *If you type in the enclosed program and RUN same, you will notice that the INPUT at line 70 (numeric) and line 160 (string) are* protected under input/return *(and, of course RUN/STOP)!!*

When you look at the listing, I'm sure your reaction will be the same as mine was when I looked at the listing and said to myself and all within earshot, "What's going on here? I tried this years ago! It didn't work then, so why is it working now?"

The key lies in the length of the input prompt – exactly 38 characters. *This configuration puts the INPUT question mark at the end of the input line and the cursor all by itself on the left of the next line. Bingo – RETURN = null. The conditional picks up the pieces and we're in business."* John Taylor

```
10 REM >> DIE THROW <<
20 :
30 PRINT [CLEAR]
40 :
50 REM >> HOW MANY THROWS ? <<
60 :
70 INPUT"HOW MANY TIMES SHALL THE DIE
   BE THROWN";N
80 IF N<1 OR N>50 THENPRINT"CAN'T DO
   THAT":FOR X=1 TO 2000:NEXT:GOTO 30
90 PRINT
100 :
110 :
120 REM > BODY OF PROGRAM OMITTED <
130 :
140 :
150 REM >> AGAIN ? <<
160 INPUT"ANOTHER ROUND? JUST GIVE A YES
    OR NO";Q$
170 IF LEFT$(Q$,1)="Y"THEN 30
180 IF Q$=""OR LEFT$(Q$,1)<>"N" THENPRINT
    "[3 UP]":GOTO160
READY.
```

Mr. Ferguson is the author of "Large Alphabet For The VIC," **COMPUTE!**, March, 1982, #22, pg. 104.

*"Today I have discovered why I never could understand how to create more than 64 characters: your article in the October '81 [**COMPUTE!** #17] issue was in error.*

*Not your fault, however. [Your information, on] page 32 of the October **COMPUTE!** is obviously from Commodore's ... new VIC manual. The VIC people are wrong.*

On pages 83-84 of their manual it gives the values to be POKEd into 36869. But the formula given on page 215 of the manual (which you printed in October) does not yield these values. It should be POKE 36869,PEEK (36869) AND 240 OR X.

I had had trouble last fall getting the erroneous Boolean formula to work, but I assumed I was doing something wrong. But with the correct values printed on pgs. 83-84, I realized the formula was wrong."
Doug Ferguson

Questions

"I would appreciate it if your readers could give me the publisher or bookstore where I could obtain the book Computers In Medicine *by Derek Enlander. It was recommended reading in a recent course I took, but I do not have the publisher's name."* L. Thomas

---

"I have a few questions regarding that pernicious malady known as "Atari lock-up." First of all, does this happen with other brands of micros? Secondly, though it is generally attributed to "over-editing," why *does it occur at all? And finally, has anyone else suffered a similar occurrence which I shall dub "two-line lock-up"? In this frustrating instance (which always seems to occur when typing "L." for a listing when the cursor is on the bottom row of the display area in text mode), the cursor advances one row after return, then does* nothing! *The screen may be cleared or reset, but any command issued will have the same two-line response followed by zilch, cipher, naught, nothing. Of course the program so intensly struggled with is in another dimension at this point and cannot be retrieved (as far as I know); the unit must be powered down to recover. Unfortunately, the programmer has a more difficult time. Comments, suggestions and, of course, remedies appreciated."* Greg Kopp

NASIR GEBELLI PRESENTS:
PATHFINDER
PATHFINDER
Never before released, a super arcade-adventure game for Atari . . . alone in a vast transparent maze inhabited by radioactive monsters and nuclear waste, you must blast the waste and free the maze of mutant monsters. By Solitaire Group, requires 24K Atari with disk drive.
MATCH RACER
MATCH RACER
Brand new for the Atari, this one or two player game tests your driving skills on a race track filled with treacherous obstacles. See how far you can get before your luck and skill run out! By Solitaire Group, requires 16K Atari with disk drive.
ANDROMEDA
ANDROMEDA
Introducing a spectacular, original arcade-adventure for the Atari . . . a battle in inner space . . . you must invade a living organism and destroy its tissues before its antibodies destroy you. By Solitaire Group, requires 16K Atari with disk drive.
HOLLYWOOD
GEBELLI SOFTWARE INC
Gebelli Software Inc.,
1771 Tribute Rd., Suite A, Sacramento, CA
© 1981 Gebelli Software, Inc. Atari is a registered trademark of Atari, Inc.

I am an Apple II owner. I have had my system for three years. I am very interested in the Apple's high-res graphics capabilities. I have been unable to locate the machine code addresses of the Apple's high-res subroutines. Any suggestions on where to find this information would be greatly appreciated. I am looking for highly detailed technical information on the Apple II high resolution graphic mode." Scott Ayers

"I hope that either you or your readers can help me. I have an original 8K Commodore PET, which I recently converted to 32K, along with the upgraded (rev. 3.0) ROMs. I've converted all my programs to work on the new system except for two, Swordquest and Escape From the Death Planet, with sound (both written by P. O'Donovan for Fantasy Games Software; originally in Madison, Wisconsin). The problem with trying to convert them is that both have lines at the beginning which don't list, that contain both BASIC commands and ML subroutines. I would appreciate it if anyone can send me fixes. I hate to lose these two excellent programs." Rudolph F. Lauer

"One time, I used Atari's DOS, and I had two copies of Lotto, a game I have made, with the same filename. I could only get the first one. Furthermore, I don't know what to change in my program. How do I get rid of the first LOTTO without killing the second one?"
Brent Edwards

RENAME does not check to see if the new name you give is the name of an existing file on the disk, so you can end up with two files with the same name. This can be very hard to deal with – any access to the first file also references the duplicate. If you try to delete or rename one of the files, you delete or rename the other as well. To fix such problems, you need to disable the feature of the DOS which causes it to access both names (which is useful for allowing multiple RENAMEs). After booting DOS with the BASIC cartridge, enter:

```
POKE 3118,0
```

You can type DOS to go to the disk menu and any RENAMEs or DELETEs will only affect the first file of the duplicate pair.

Note that you should not re-save this modified DOS to disk, since it will not be able to perform multiple RENAMEs. You can also restore the DOS with POKE 3118,184. (This information was found in the *OS/A + Users Manual,* published by Optimized Systems Software, Inc.) ©

Computers And Society

David D. Thornburg
Innovision
Los Altos, CA

And Now Let's Hear It For The Cookie Monster...

Omens are funny things. I was sitting on a flight to New York next to a person who was telling me of her concern with the Game Arcades. It seems that a midwestern community close to her home had recently banned these places in response to parent's beliefs that such activities were contributing to their children's truancy rate. Her belief was that, if anything, school use of computer games should increase, not decrease. From her perspective, the computer would free the teacher to give more individual attention to the students. I told her that I found her ideas to be similar to my own, and was amused that this conversation took place on a trip during which I would be seeing more than 50 games written or acquired by Children's Television Workshop (CTW) for use in their computer building at the Seasame Place park.

I haven't been to Sesame Place, but I am told that it is a wonderful place for parents and children to enjoy themselves in an environment that includes elements of a museum, recreation area, and computer game arcade.

Since I had provided CTW with some minor assistance on the technical aspects of the computer activity at Sesame Place, I was most pleased to be given an opportunity to see the programs which were captivating the many thousands of children who visited there. Two events suggested that my experience would be worth sharing with you. First, at least two more Sesame Place parks are planned, greatly increasing the number of families who can visit this facility. Second, I had read that certain of the CTW programs were going to be released for the Apple computer, and that versions for other computers might follow in the not too distant future.

Before describing the CTW effort, it might be beneficial to explain that major efforts in the generation of educational software for personal computers have been tried before. Among the more successful activities, one must list the Minnesota Educational Computer Consortium (MECC). This branch of Minnesota's state government provides teachers all over the state with access to centralized computers, as well as providing special purchase contracts for Apple and Atari computers for classroom use.

Many of the MECC programs (such as the simulation, Lemonade) have become quite popular. Through a variety of marketing arrangements, MECC software is now becoming available for the Apple and Atari computers on a nationwide basis. MECC is one of the pioneers in the educational use of computers, and has done a fine job overall. The concentration of talent and effort required for the success of a venture the size of MECC is sufficiently rare that they have appeared to be a beacon in a vast sea of mediocre educational software.

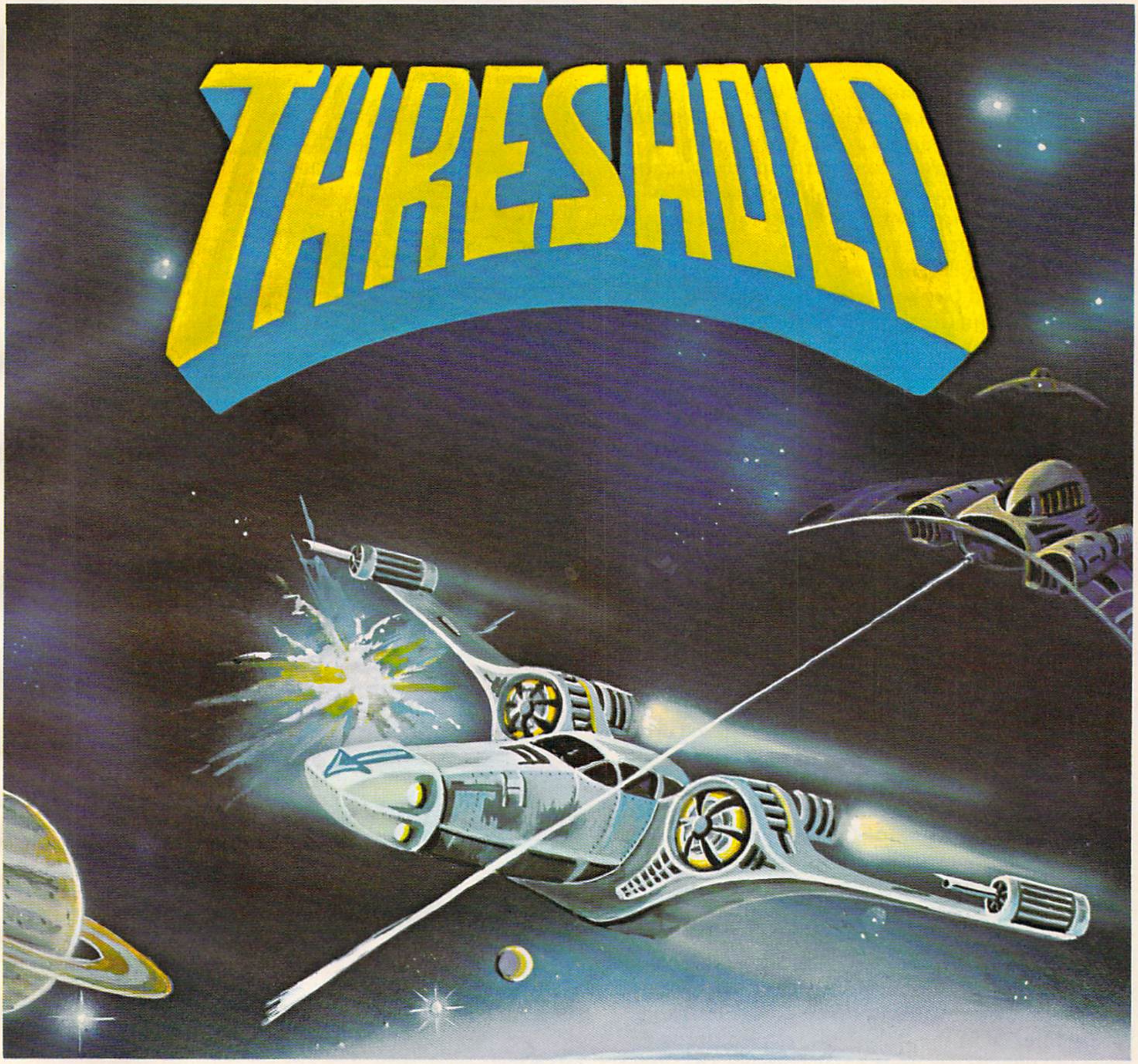
What excites me is the fact that by introducing educational software products of their own, CTW will further legitimatize the business of high quality educational software packages and thus, encourage the development of more packages of this type by the many hundreds of excellent programmers located in their homes all over the world.

In order to be useful, multi-program educational software offerings should include a distribution of educational goals and skill level requirements. In addition, these programs should be interconnected with a well defined common theme.

The common theme for the CTW software is very simple. The activities must be both educational and entertaining.

As at other high quality software houses, the software teams at CTW include designers, artists, teachers, code writers, etc. Once a program is designed in storyboard form it is then implemented for classroom testing. Based on the results of this testing, the program is then modified and tested again. By the time the program is approved for use, it has undergone a massive amount of testing, evaluation, and modification. The effect of this effort is obvious when these programs are run. There is a level of quality and polish which sets these programs apart from most of the other educational software I have seen.

While MECC software is available for a wide range of grade levels, the present CTW programs are geared primarily for the youngsters who would watch Sesame Street on television. There are games for pattern matching, pattern recognition, eye-hand coordination, counting, etc. One of my favorite programs requires that you identify a picture by looking through a keyhole that can be moved around the surface of the object you are trying to identify. The keyhole only shows a part of the underlying picture, so the user has to reconstruct



ON-LINE SYSTEMS introduces arcade gaming as an art form. THRESHOLD, by WARREN SCHWADER and KEN WILLIAMS, features fast smooth animation, HI-RES graphics, and more challenge than you'll find in any other arcade game on the market.

THRESHOLD is an arcade game with alien attackers galore. In fact, there are more monsters out there than we expect you'll ever see. THRESHOLD was designed to be an arcade game that you won't get bored of, and that means a constantly changing game with a graduated skill level, but more than that, THRESHOLD means constant fun.

THRESHOLD runs on any 48K Apple II or II Plus DOS 3.2 or 3.3. Available now for $39.95 on disk from your local computer store or you may order directly from . . . . .

Now Available For ATARI 400/800 with 40K and a Disk Drive

ON-LINE systems

36575 Mudge Ranch Road - Coarsegold, CA 93614 - 209-683-6858

ADD $1.00 FOR DIRECT ORDERS

Visa, Master Charge, C.O.D. or Checks Accepted the overall image from memory. When I played this game, the object was a pig. I typed *ANIMAL* as my response and was rewarded for my correct answer. This illustrates another important feature of the CTW software. Acceptable answers are carefully chosen to allow some variation in response. Correct (but generalized) answers are accepted as are some misspelled answers. For example, I consistently spelled Grover as *GROWVER* with no complaints from the system (although the computer did show me the proper spelling of Grover's name each time as a reminder).

CTW and MECC are not the only organizations devoted to the creation and distribution of high quality educational software. I know of several teams of brilliant people who are devoting their efforts to this area.

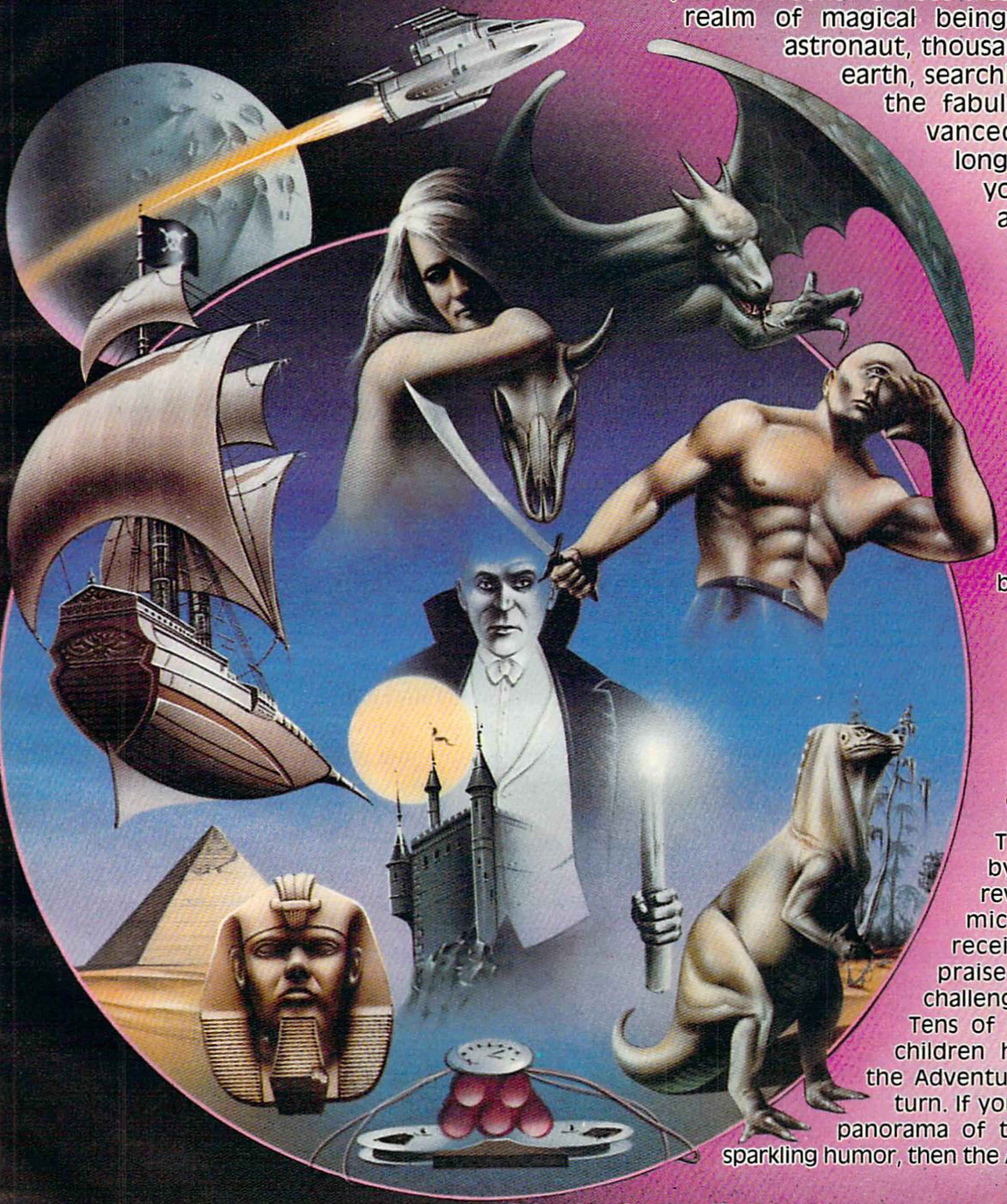
One excellent team is located at Automated Simulations – a software house known primarily for its adventure games. While these games (created by the EPYX division) are their mainstay, releases from the Mind Toys division suggest that Automated Simulations has a promising future in educational game software.

Two recent releases of theirs, Jabbertalky and Ricochet, deserve comment. Both of these games are very high in educational value. While Jabbertalky is a word and sentence game and Ricochet is a game of logical thinking, they each share a quality that I have not seen before in personal computer games – dynamic handicapping. When two players are competing with each other, the computer is paying close attention to the skill levels displayed by each player. As each new round is started, the challenge level for each player is adjusted accordingly. The result of this is that a novice can compete against an expert without being devastated and losing interest in the game. The expert doesn't have to "play down" to the newcomer – the game will take care of the discrepancy itself. Now parents and their children can play the same logic game and find themselves equally challenged. Even if this was the only thing worth mentioning about these games (and there is much more to recommend them), I would still suggest that you see them.

The design team at Automated Simulations includes Bernie de Koven – the author of "The Well Played Game" – a book I reviewed last year.

As I look toward the future, I see a new class of educational software being developed which bridges the gap between the traditional CAI and drill and practice programs and the arcade games. The creation and distribution of this new software should do much to secure an appropriate place for the personal computer in education. ©

WHO ARE YOU THIS TIME?



Are you the seeker of lost treasures in an enchanted realm of magical beings? Perhaps you're an astronaut, thousands of light years from earth, searching the galaxy's rim for the fabulous treasures and advanced technologies of a long-dead civilization. Maybe you're the plunderer of ancient pyramids in a maddeningly dangerous land of crumbling ruins and trackless desert wastes.

Imagine.

You wish to soar to other worlds, to behold wonders never beheld by mortal eyes, to dream unrestrained to the furthest limits of your imagination.

Imagine now that you can.

The ADVENTURE SERIES by Scott Adams has been reviewed by **every** major microcomputing magazine, receiving only the highest praise for its mind-puzzling challenge & refreshing originality. Tens of thousands of adults and children have matched wits with the Adventure Master. Now, it's your turn. If you crave the challenge and panorama of the exotic, touched with sparkling humor, then the Adventure Series is for you.

With the incredible Adventure Series by Scott Adams,
YOU ARE WHOMEVER YOU WANT TO BE, WHENEVER YOU WANT TO BE!

	SINGLE TAPE	12 PACK TAPE	SINGLE DISK	TRIPLE DISK	12 PACK DISK
APPLE	$19.95 24K	$129.95† 24K	$29.95* 48K	$69.95* 48K	—
ATARI	$19.95 24K	$129.95† 24K	—	$39.95 40K	$129.95† 48K
CP/M (Z-80)	—	—	—	—	$129.95†
TRS-80 MOD 1&3	$19.95 16K	$129.95† 16K	—	$39.95 32K	$129.95† 32K
TRS-80 MOD 2	—	—	—	—	$129.95

PRICES SUBJECT TO CHANGE WITHOUT NOTICE

Adventure Hint Book (All 12)	$ 7.95
Adventure Hint Sheet (Single)	1.00
Commodore VIC (Cartridge)	39.95
T.I. 99/4 (Tape or Disk)	29.95

Requires Adventure Command Module

*Scott Adams' Graphic Adventure Series Apple Disks-Full Color Hi-Res Graphics.
†12 Adventure Hint Book Incl.

SHIPPING & HANDLING ARE EXTRA

Adventure INTERNATIONAL
A DIVISION OF SCOTT ADAMS, INC.

BOX 3435, LONGWOOD FL 32750
(305) 862-6917 (QUESTIONS)
ORDER FROM YOUR FAVORITE DEALER or CALL TOLL FREE (800) 327-7172
(ORDERS ONLY PLEASE A.I. #119)

Introducing computers into the office can create problems of employee adjustment to the new technology. A word processor, for example, can seem uncomfortably fast, complex, and altogether too sensitive to employees who are used to typewriters. Here, Craig Brod, President of Technostress International, explains some techniques which can speed the acceptance, the diffusion, of computers into the office environment.

Guest Commentary:

Managing Technostress: Accelerating Diffusion Of New Technology

Craig Brod
Berkeley, CA

The process by which innovation spreads is a diffuse one, and the rate at which the new replaces the old within companies exercises a major influence on costs. The introduction of a computer into an office environment will produce disruption. Delayed schedules, decreased performance and productivity will add to the cost of implementing the computer.

Additionally, there will be a delay between when the computer accounts for ten percent of the office productivity and the date when the proportion reaches 90 percent. The faster the computer is adopted, the more quickly the average costs of a company will approximate those of the new technology. Also, the smaller the gap will be between best and only average productivity. The rate of diffusion of the computer depends on personnel who must feel comfortable using it.

Accelerated diffusion of new technology necessitates a high level match between operator and machine. This match is an arrival point: altered tasks, roles, and machine packing must be learned. A secretary who trades her typewriter for a computer word processor has to learn to increase her concentration on work (due to the sensitivity of the machine), her hand-eye coordination (due to the speed of information flow), and her ability to respond to signs instead of symbols (due to the shorthand language of the computer). All of this learning, in turn, depends on flexible mental functions.

Even under optimal working conditions – that is, conditions that lack union-management conflicts, major compensation issues, and layoffs – it is difficult for employees to learn so many new skills well. In fact, employees often lack the necessary skills to make a rapid and performance-oriented adjustment to new technology; neither their capabilities nor those of the computer are tapped. Here we often find *technostress.*

Limited Diffusion

Technostress is the condition resulting from the inability of an individual or organization to adapt to the introduction and operation of new technology. It has a negative impact on human performance by: (a) shifting attention from work-congruent stress to internal states of distress, (b) reducing the ability to process information accurately, (c) slowing the response time to computer generated demands, and (d) breaking up natural work-rest pauses that characterize normal work patterns.

Technostress, which begins as reduced performance, results in behavior that slows the rate of diffusion. Let's highlight three of the most pertinent manifestations of technostress.

First, patterns of use. Shortly after the introduction of a computer, for example, a few employees, for reasons we don't yet understand, will excell at using it. This manifests itself as repeated use and little unnecessary movement around the office. Many employees, however, show initial excitement, genuine experimentation with new skills, and then, due to technostress, are unable to make a proper adjustment. What follows is withdrawal from using the computer: increased time on non-machine tasks, hand calculated data, and social activities away from the machine. The result is a centralized pattern of use of the computer.

Second, high error rates. Indicators of high error rates begin to surface early. Employees often forget or violate new procedures. This ostensibly occurs because new procedures require new learning. However, closer scrutiny reveals a more precise explanation. Employees who learn quickly, and are intellectually astute, for example, have higher order needs to emit novel responses in relation to work rules and procedures they perceive to be simple and repetitive.

Therefore, these employees, in order not to be bored, attempt to invent new rules and shortcut new procedures. It is not uncommon to see an adept secretary, who has only recently begun to operate a computer, skip over sections of the training manual and try to learn procedures on her own. Also, in a short time, she will type in information that is extraneous to the program.

Overload

On the other hand, a large number of employees experience overload when learning to operate a computer. This results from a combination of factors: the high workloads which continue while learning of new procedures is taking place, poor problem-solving skills, self-doubt concerning one's ability to master new tasks, and, often, jealousy of those in the surrounding environment who are learning quickly.

Overload takes the form of forgetting new procedures and applying old solutions to new problems. It is common to see an overloaded secretary, turned computer operator, revert to past conditioning by typing on the computer as though it were a standard typewriter. When the computer will not accept what is being typed, the operator continues to replicate these trials, while insisting that he or she is doing everything according to code. This usually ends with the secretary blaming the machine or the computer manufacturer for personal failures.

One measure of inappropriate learning is inaccurate input of data. Once inaccurate data is entered into a computer system, for example, it sets up a technostress-error cycle (see Figure 1 below).

Figure 1 indicates that errors occur in a closed system. Unlike a typist, whose spelling mistakes on an interoffice transmittal may go to one or two other people, and elicit a comment of sloppiness, an operator who enters inaccurate data into a computer system has a direct impact on many people who depend on accurate information. In addition, errors are visible to anyone within the information network, and the operator is held accountable for errors that may have dramatic consequences. A common result is that operators of computer systems often suffer from what I call the "tyranny of errors": the feeling that they lack the skills to control errors frustrates them, and this frustration, in turn, is converted into the feeling that the machine is too sensitive to use for anything except simple problems and assignments.

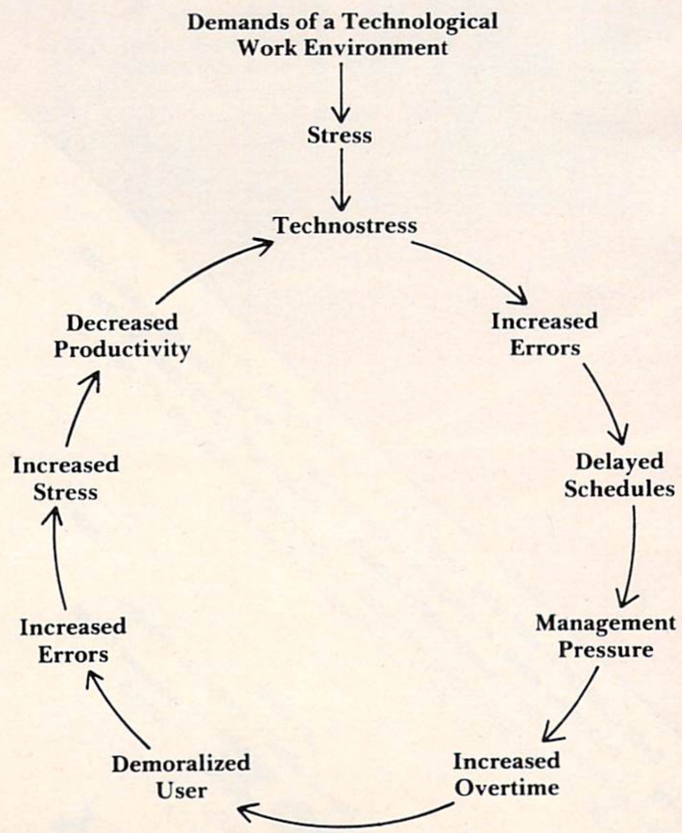


Figure 1: Technostress Error Cycle

Third, sabotage. This occurs in a variety of ways. A computer operator who continually bangs a knee against the computer, causing it to lose its memory, complains that the computer was designed poorly. A programmer, impatient with the response time of a computer terminal, will sometimes hit the terminal or bang down on the keys. Recently, I was called in on a consulting assignment by an operations officer who wanted to prevent the firing of a programmer who had a history of hitting visual display terminals. The man, who suffered from technostress, needed to be trained to work more effectively with new technology.

Three Phases Which Improve Diffusion

Diffusion of new technology can be accelerated by shaping human performance. Technostress training is designed to reduce technostress and build a better match between employees and new technology. There are three phases to the training.

Phase I – Education. The educational phase is designed to provide the employee with an explanatory scheme for understanding the nature of technostress and his or her responses to it. The most important aspect of this phase is the conceptual framework: technostress is made plausible to the employee and its acceptance naturally leads to the practice of specific coping techniques.

For example, a trainer working with a group of employees identifies the demands of the new technology: specific communication patterns, short-cycle repetitive work, machine dependent time, altered task-related thinking functions, increased hand-eye coordination, and altered kinesthetic responses to the environment. The responses to these demands are individually elicited from members of the group. The point here is to begin to build a prospective set of skills which can be used to reduce the technostress.

Not only do employees appreciate learning about the changes taking place, but they also im-

mediately begin to sense increased control over task-specific behaviors. In addition, appreciation of the complexity of the changes reinforces the learning of new skills.

After identifying the demands and individual responses, the trainer points out the elements of a technostress response: heightened arousal – increased heart rate, sweaty palms, rapid breathing, bodily tension or a speedy, buzzed feeling; blocked learning – past conditioning, avoidance thoughts, images, and self-statements (e.g., old solutions to new problems, rejection of new procedures, a sense of helplessness, failure thoughts; distorted information processing function (IPF) of which four modes are identified – inventive or novel behavior, mediated or rule behavior, attentive or emergency behavior, conditioned or stimulus-response behavior.

Under technostress, the modes are used inappropriately to perform a variety of tasks. The trainer then indicates that training will be directed toward: (a) controlling physiological arousal, (b) substituting coping statements for negative learning statements, (c) using imaging to accomplish work tasks more effectively, and (d) employing accurate information processing.

The educational phase concludes with a discussion centering on employees viewing technostress as a series of phases, rather than as one massive reaction. To master technostress, four phases are suggested: preparing for new demands, confronting or handling new demands, possibly being overwhelmed by new demands, and finally, reinforcing oneself for having coped.

The initial educational phase provides employees with a cognitive framework to better grasp work demands and the appropriate responses to them. It secures the transition into the second, the rehearsal phase of the training.

Rehearsal

Phase II – Rehearsal. The second phase of technostress training is designed to provide employees with a variety of coping techniques to use at each of the various phases of the coping process. The coping techniques include both direct action and cognitive coping modes. Direct action modes include: identifying task-related stressors, arranging an action plan, regulating work pauses, and physical relaxation.

The cognitive coping modes, which are most important because they are especially adversely affected under conditions of high stress, are treated in three ways. First, viewing cognitive processes as a series of self-statements that the employee said to himself. Thus, appraisal of learning blocks and self-perception are translated into specific self-statements. The modification of the employee's internal dialogue – "I can't learn these procedures" – is accomplished by having him become aware of and monitor the negative self-defeating self-statements he emits under conditions of high work-congruent stress.

In collaboration with the trainer, the employees are able to generate sets of coping self-statements that encourage them to: (1) assess their technostress level; (2) control negative thoughts and self-statements; (3) acknowledge and use positively the arousal they are experiencing; (4) cope with the overwhelmed feeling they might experience; (5) reinforce themselves for having coped.

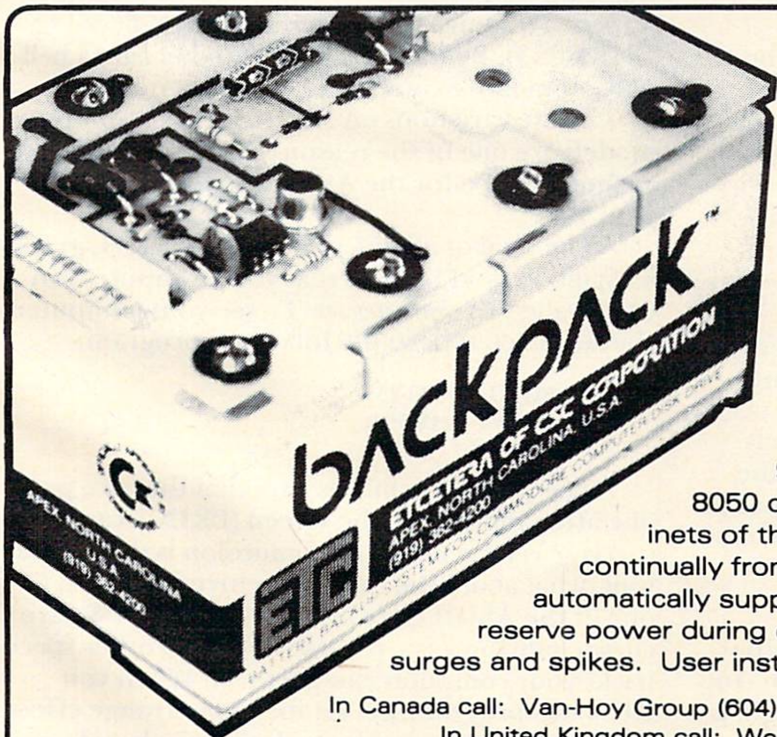
Second there is technostress reduction emphasizing high-intensity imaging. Employees are instructed to take the task-related stressors they have identified and visualize how they would alter them under conditions of technostress. For example, a programmer, who identified irregular computer response times as a source of technostress, was asked to visualize an increase in his workload, and a decrease in supervisory support, then to visualize alternatives to his technostress. The alternatives, arrangement of contingency work, pacing, of morning activities, and paused relaxation, are rewarded by the trainer.

A third aspect of rehearsal is practice and familiarity with identified information processing functions. This requires two steps. Employee cognitive styles can be assessed on a test. This test briefly identifies the way employees organize information. Then, employees are instructed in how to use their cognitive style to make the accomplishment of tasks easier and more efficient. Next, employees are instructed in how to identify and use IPF's to solve task-related problems. IPF's can be learned and practiced in structured role-play and workbook assignments. Special attention is given to repetitive association between past conditioning and methods of returning to the proper mode of information processing.

The Final Phase: Monitored Work

Phase III – Network Training. Once the employees become proficient in employing behavioral and cognitive coping skills, the trainer suggests that the employees should test out and practice their coping skills by actually employing them under day-to-day stressful conditions. At this point, a number of employees are selected as network mentors. They are given additional training in problem-solving methods, technostress, and leadership skills. These mentors are able to monitor the environment for signs of technostress and to act as a valuable resource to other employees. After some months, the mentors can fill out a brief questionnaire to assess the degree to which the knowledge and skills learned in the education and rehearsal phase have been generalized to the high technology environment.

Technostress training involves discussing the nature of technostress, and individual responses, rehearsing coping skills, and testing these skills under actual work conditions. Network mentors should be educated to monitor the impact of the training. Preliminary results suggest that such a skills-oriented, technostress training procedure is successful in accelerating the diffusion of new technology into a working environment. ©



The Beginner's Page

The ASCII Code

Richard Mansfield
Assistant Editor

It's easy to see how a typewriter puts a letter of the alphabet onto a piece of paper. You press the "F" key and the paper is struck by an inked ribbon, pushed against the paper by a small metal image of F. But what happens when you hit the F on a computer? It puts the number 70 into one of its memory cells.

What does 70 have to do with *F*? To answer that, we'll need to know what the ASCII Code is and learn the meanings of two BASIC commands: CHR$ and ASC.

Type in this short program:

```
5 DIM A$(1) : REM THIS LINE IS ONLY
  NECESSARY ON ATARI
10 INPUT A$
20 PRINT ASC(A$)
30 GOTO 10
```

When you RUN this, you can type letters on your keyboard and see them translated into numbers. (Hit the RETURN key after each one.) Try "F" and you'll get 70. What you are seeing is called the ASCII Code. Computers only store *numbers* in their memory cells. In fact, they can only store the number one and the number zero. (For a more detailed explanation of how computers remember things, see "The Beginner's Page," **COMPUTE!** March, 1982, #22.) The computer can store words and symbols or pictures or anything else in patterns of these ones and zeros.

To store the letters of the alphabet, symbols like the percent sign, punctuation marks – all the keys on your keyboard – the computer uses a special code, the American Standard Code for Information Interchange, ASCII.

When Seven Is Not Seven

If you are RUNning the program above, type the number seven on your keyboard. It's not seven! In the ASCII code, it's 55. The number six, though, is 54, so the scheme is not entirely random. Why didn't they just use the number seven to stand for number seven in this code?

There are reasons for everything. If you learn to program in *machine language,* as opposed to BASIC, you'll work with hexadecimal numbers. In hex, the ASCII code for zero is 30 and seven is 37. In hex, it makes a bit more sense.

But back to ASCII. ASC, of course, is short for ASCII and you can find out what the ASCII equivalent of a single character is by typing PRINT ASC("F") or by asking for the ASC of a string variable (as we do with A$ in the program above).

You can go the other way with CHR$. This is BASIC's "character string" command. Where ASC translates a character into ASCII, CHR$ translates ASCII back into a character. So, you give CHR$ a number between 0 and 255 and it will give you a character. Here's a short program to see how CHR$ works:

```
10 INPUT X
20 PRINT CHR$(X)
30 GOTO 10
```

Each manufacturer has deviated somewhat from standard ASCII. For example, the Atari uses a code called ATASCII which is very similar to ASCII, but there are some differences. The creators of the ASCII standard had decided that the number seven should not print any character. Instead, seven is supposed to ring a bell, or a buzzer, or whatever sound your computer can make that can be used like a bell on standard typewriters. But on the Atari, if you type PRINT CHR$(7), the computer puts a graphics symbol, a large backslash, on screen. To sound the buzzer, use PRINT CHR$(253). By the way, CHR$(7) *does* ring a bell on Commodore computers with built in sound.

These variations on ASCII between computer models are one of the reasons that you cannot take a game on tape for the Apple and LOAD it into your PET.

The total of all 255 possible characters, graphics symbols, and buzzers that your computer can use is called its *character set.* To see your computer's character set, type in the following program:

```
10 FOR I=0 TO 255
20 PRINT CHR$(I);
30 NEXT I
```

The semicolon makes sure that they are put one after another on the screen (PRINT causes a carriage return unless the semicolon is there). But wait, what about the carriage return itself? Isn't it one of the ASCII code numbers? You bet. So are cursor-moving keys, reverse field, and other special tricks your computer can perform. When you RUN this program, get set for some strange effects. As the program runs through the numbers from zero to 255, it will encounter the clear-the-screen character too.

There are a number of uses for ASC and CHR$. With CHR$, you can send characters to your computer that cannot be typed in from the

keyboard. There might be no key on your keyboard which rings the bell. Use CHR$. This can be done directly from the keyboard or as part of a BASIC program.

Another common problem is trying to print quotes. You can't just type PRINT "THE "BEST" COMPUTER" because the set of inside quotes around the word *best* will confuse the computer. Try it. It will think you are printing the words *the* and *computer* with a numeric variable (*best*) between them. It will print a zero since the variable *best* has no value. To achieve the result you want, type PRINT "THE " CHR$(34) "BEST" CHR$(34) "COMPUTER".

If there is a printer attached and "listening" (responding) to your computer, you can make it do a carriage return by typing PRINT CHR$(13). Or, if the printer has a bell, try PRINT CHR$(7). Most printers accept the standard ASCII code and their instruction booklets will usually explain what numbers to send to perform backspace, underlining, etc. Remember, in this case it doesn't matter what codes your *computer* is using. When you send a letter to the printer, the *printer's* code (probably standard ASCII) will determine what gets put on paper. ©

SYSRES™

THE ULTIMATE RESIDENT PROGRAM MANIPULATION SYSTEM FOR PET™/CBM™ MICROCOMPUTERS

SYSRES™ EXTENDED DOS SUPPORT COMMANDS

@ (type "N" keyboard)
← (type "B" keyboard)
! (original keyboard)
> (for 'wedge' users)

These commands may be used interchangably, to perform the following dos support functions.

Disk	Printer	Tape	Directory	Modes	Command	Function
x				3	@	Display disk status / send command
x					@N	Format (header) a new diskette
x					@I	Force initialize diskette
x					@V	Validate diskette (collect)
x					@D	Duplicate diskette
x			x	4	@C	Copy or concatenate disk file(s)*
x					@R	Rename file
x			x	3	@S	Scratch file(s)*
x	x				@$	List directory**
x					@U:	Reset disk drive
x	x	x	x	6	@L	List disk file or BASIC program**

Note: Some of the disk utility command set may also be used, if an appropriate direct access channel has been opened.

* Standard command with added options.
** Added disk command.

SYSRES™ EXTENDED EDITOR COMMANDS

Disk	Printer	Tape	Directory	Modes	Command	Function
x			x	4	/	Quick load from disk
x			x	4	↑	Quick load from disk with auto run
x			x	2	APPEND	Append from disk to end of current program
				4	AUTO	Auto line number (allows header)
x			x	3	BLOAD	Load machine language (binary) file
x			x	3	BRUN	Load and execute machine language program
	x			776	CHANGE	Change pattern to another pattern
				2	CLOSE	Close one or all files
				1	CMD	Set output to file (does not send "READY.")
				4	DELETE	Delete a range of lines from program
	x			1	DUMP	Dump all scalar variables to screen or file
x			x	2	EXEC	Execute a file as keyboard commands
	x			240	FIND	Find occurances of a pattern
x		x	x	3	GET	Read a sequential file into editor
				7	KEY	Define a key as a special function
				1	KEYS	Turn key functions on
				1	KILL	Disable SYSRES™
				1	KILL*	Disable SYSRES™ and unreserve memory
	x			10	LIST	Improved BASIC LIST command
x		x	x	3	LOAD	Defaults to disk drive
x			x	2	MERGE	Merge from disk into current program
	x			1	MON	Break to current machine language monitor
				1	OLD	Restore program after "NEW"
x	x	x	x	24	PUT	Send program to disk as text file
				6	RENUMBER	Renumber all or part of program
				2	RUN	Run current program, ignores screen garbage
x		x	x	3	SAVE	Defaults to disk drive, allows replace
x		x		1	SETD	Set disk device #, allows multiple drives
	x			4	SETP	Set printer channel, format mode, paging
	x			4	TRACE	Select 1 of 3 trace/step modes and speed
x		x	x	3	VERIFY	Compare current program against disk/tape
				1	WHY	Print position of last error
				1	WHY?	List line of break or error
x	x				*	Send output to printer
	x			1	#	Display current version of SYSRES™

JUST A FEW OF THE FEATURES OF SYSRES™

* Fast up/down scrolling which works!
* Advanced repeat-key routine!
* Re-define any or all keys as any keyword (full or short form) or as any string up to 255 charactors long!
* Auto line numbering which can feed a string of up to 127 charactors as well!
* Extended DOS support (requires DOS 2A or greater)!
* Never enter another file name! All file commands work from the directory!
* Supports multiple disk drives!
* List BASIC programs, sequential and relative files without loading them into memory!
* TRUE PROGRAM MERGE (overlay). Supports subroutine libraries!
* Load and run machine language programs with parameter passing!
* Supports multiple printers!
* Automatic printer output with paging plus formatted listings with full ASCII code conversion including cursor control and special charactors for non–CBM™ printers!
* Edit text files and assembler source code without leaving BASIC!
* Renumber part of a program or even change the order of lines!
* Over 700 FIND/CHANGE commands including variable names ("A$" will not match "BA$"), pattern matching with "wild–cards", and even commands to remove spaces and REM's!
* Three TRACE modes including trace variables!
* Does not affect BASIC program operation!
* One AUTO–BOOT DISKETTE works for ALL PET™ or CBM™ computers (BASIC 2.0 or greater with at least 16k of RAM.). SYSRES™ requires NO ROM SPACE or extra boards, so you can take it with you if you want to use another computer. It may be put above the screen if you have RAM there. It boots automatically without disturbing any program in RAM!
* Diskette and Extensive Manual – only $75
Documented Source Code (YES!) – $150
Custom versions are available for unique systems!

Dealer enquiries are welcome.

VISA MasterCard

CANSOFT

CANSOFT DATA INC.

MICROCOMPUTER SERVICES

PROGRAMMING * CONSULTING * DESIGNING * EVALUATING

BOX 76752, STN.S, VANCOUVER, B.C. V5R 5S7 TEL:(604)437-6667

This program will analyze your life insurance needs and give you a guide to the coverage necessary to adequately provide for your family. The program will run on VIC-20, Apple, Atari, PET, and OSI computers.

Life Insurance Estimator

David E. Pitts
Houston, TX

Calculating the life insurance needs of a family can be a time consuming procedure since the entire financial picture, the age of all family members, and future plans must be considered. Because of this, most families probably don't have a clear understanding of their life insurance needs. The program which is described here will help make this planning a little easier since it removes the burden of using discount tables, annuity tables, and year-by-year survivor benefit tables.

The program calculates a balance sheet for the family which gives the family assets and the family liabilities (see the example run). The user inputs the assets: life insurance, real estate, securities, cash on hand and also inputs a liability: current uninsured debts. All other liabilities are calculated by the program based upon the monthly goals for the insurance plan, the social security benefits (or other benefit plan), the age of and number of children, the age of the spouse, the current tax bracket and the interest rate expected on annuities and funds set up from the insurance proceeds. The default tax bracket used is 27.3% and the default savings rate is 5.5%, yielding a 4% annual return.

The monthly income goals are by phase: 1) spouse with children at home, 2) education fund, 3) spouse between child rearing and retirement, and 4) spouse's retirement. Inflation is not directly accounted for in the calculations. However, in order to offset inflation, the fund for the spouse's retirement is not discounted for the number of years in the future when retirement will occur. The user may want to adjust the other income goals upward as a function of expected inflation and the number of years in the future when the fund will be required.

The use of this program should not be considered the only approach to understanding a family's life insurance needs; advice from life insurance agents, accountants, lawyers, etc. may be more appropriate for a particular situation. Careful evaluation of the program outputs is advised based upon all family needs, both those entered into the program and those which the program may not allow.

The program is based upon the Consumer's Union's book 1972 *Revised Edition of the Consumers Union Report on Life Insurance, A Guide to Planning and Buying the Protection You Need* (Grossman Publishers, New York, 1973).

Social security benefits for children are in force until they are 18 at which time they stop drawing benefits unless they are full time students. A full time student can often receive higher benefits, thereby causing the family's maximum allowable benefits to be exceeded. The effect of this is to reduce the amount available to the family. If the family has another benefit plan which provides for the spouse (independent of the children being in college after 18) then the maximum permitted monthly benefit should be set to a large number in order to adjust for this limiting factor. Social Security doesn't allow for benefits between child rearing and retirement so, if your benefit plan allows such benefits, or if your spouse plans to work during this period, then you should reduce the monthly goal for that period appropriately.

The year by year monthly income from the retirement plan (e.g. Social Security) for the years minor children are in the home can be verified by printing the variable C(I) for I = 1 to 18.

Organization Of The Program

Statement	Description
30	Rounds to next highest 1000 dollars.
40	Function for positioning data in balance table.
60-190	Inputs income goals and benefits.
210-240	Calculates income table components for widow and children C(I) and students S(I).
250-290	Sums the benefits and checks for total not to exceed maximum allowable benefits.
300-340	Inputs Tax bracket and Interest rate to be applied immediate annuities for funds needed for future income. Interest income is reduced by Tax Bracket percentage.
360-390	Reduces 18 year income table to several periods, each of which has constant monthly income benefits.
400-420	Calculates family income fund.
440-450	Calculates education fund assuming 40 month eligibility.
470	Retirement fund calculation.
490-500	Widow's income calculation between child rearing and retirement.
520-730	Balance sheet calculations and printout.
780	Print routine for proper spacing of variables in balance sheet table.

Sams helps you cut deeper into the workings of your Apple with two new additions to our crop of Apple® Computer Books.

APPLE® FORTRAN by George Blackwood is a beginner's guide to using FORTRAN, a powerful, science-oriented language, on your APPLE® computer. NO. 21911, $12.95

CIRCUIT DESIGN PROGRAMS FOR THE APPLE II® by Howard M. Berlin lists several BASIC-language programs which simplify the design and analysis of commonly encountered electronic circuits. NO. 21863, $15.95

The Sams Apple® collection also includes: APPLE® INTERFACING with tested designs for interfacing circuits and their associated BASIC software, allowing you to connect your APPLE® computer to the real world. No. 21862, $10.95

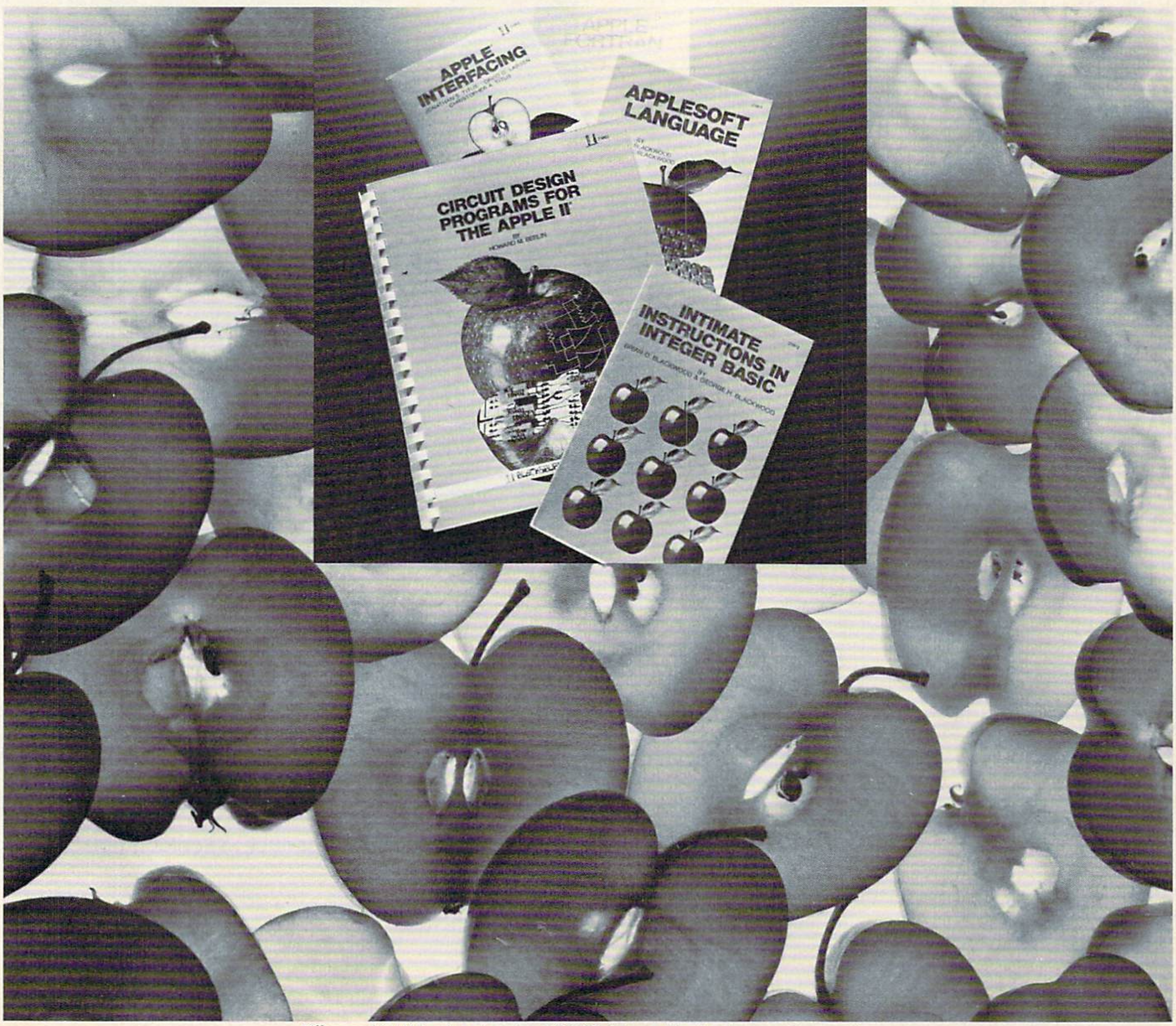
APPLESOFT LANGUAGE, NO. 21811, $10.95 is a beginner's guide to APPLESOFT, the native language of the popular APPLE II Plus® microcomputer. INTIMATE INSTRUCTIONS IN INTEGER BASIC introduces you to the fundamental language of the standard APPLE II®. NO. 21812, $7.95

Sams continues to slice through the Apple®, giving you a good look at some of the very best pieces of this powerful machine.

To order these Sams Books or to get the name of your Sams retailer, call 800-428-3696 toll free, or 317-298-5566 and reference AD180.

SAMS BOOKS ™
Howard W. Sams & Co., Inc.
4300 West 62nd Street, P.O. Box 7092
Indianapolis, IN 46206

NEW SLICES FOR YOUR APPLE®



Offer good in USA only and expires 9/30/82. Prices subject to change without notice.
Visit Sams at the NCC and Comdex. Booth Nos. 8223 and 1710.

Program Run

```
              LIFE INSURANCE PLAN

--------------------------------------------

ENTER # OF CHILDREN UNDER 18? 3

MONTHLY INCOME GOAL FOR SPOUSE & CHILDREN?
750

MONTHLY INCOME GOAL FOR SPOUSE'S RETIREMENT?
500

MONTHLY INCOME GOAL FOR SPOUSE BETWEEN CHILD
REARING AND RETIREMENT,IF NO CHILDREN <18,
THIS APPLIES NOW TO RETIREMENT? 250

LUMP SUM EDUCATION FUND FOR EACH CHILD? 12000


--------------------------------------------
START WITH OLDEST CHILD


AGE OF CHILD #  1 ? 12
AGE OF CHILD #  2 ? 10
AGE OF CHILD #  3 ? 3
--------------------------------------------
ENTER SOCIAL SECURITY (OR OTHER PROGRAM)
MONTHLY SURVIVOR BENEFITS:

FOR A SPOUSE & 2 OR MORE CHILDREN? 402
FOR A SPOUSE & 1 CHILD? 326
FOR A STUDENT 18-21? 163
FOR A FAMILY'S MAXIMUM PERMISSABLE BENEFIT?
402
FOR SPOUSE STARTING RETIREMENT AT AGE 62? 179
--------------------------------------------
IF INTEREST RATE REQUESTED ISN'T KNOWN ENTER
'0'

CURRENT TAX BRACKET IN %? 30

SAVINGS ACCOUNT INTEREST RATE FOR SURVIVORS
ANNUITY? 10
```

Using This Program On PET, Apple VIC-20 And Atari

•Program 1 is a completely general Microsoft BASIC program and will run without modification on any PET/CBM, Apple, OSI, etc. However, it is written for a 64-character display. The only problem is with the "balance sheet"; it will not look as neat and formatted as the author intended. Two solutions are easy to implement. 1) Have the balance sheet printed out on an 80-column printer, 2) make minor changes to the program. Solution 2) requires you to divide the number in TAB statements by two (e.g. PRINT TAB(35) becomes PRINT TAB(17)) and change line 780 to read:

```
780 V=FNP1 (V):PRINTTAB(V/2);:RETURN
```

• VIC owners should substitute the lines in Program 2 for those in Program 1. Although the printout will not look as the author intended, due to the 22 character display, you can easily read the information. You will also probably want to delete line 580 (by entering the number 580 and pressing RETURN, or just not typing it in.)

• Atari owners should type the lines given in Program 3 in place of the corresponding lines in Program 1 to convert the program to Atari BASIC and correct for a 40-column display. You may wish to make other cosmetic changes, such as using lowercase for prompts, or using a formatting routine for the balance sheet.

```
CURRENT AGE OF SPOUSE? 33
----------------------------
ENTER FAMILY ASSETS

LIFE INSURANCE? 25000

CASH ON HAND? 10000

REAL ESTATE EQUITY? 7000

SECURITIES? 0

OTHER ASSETS? 0
----------------------------
UNINSURED DEBTS, OTHER THAN
HOME MORTGAGE? 0
----------------------------
```

```
                          BALANCE SHEET

FAMILY ASSETS                     FAMILY LIABILITIES

LIFE INSURANCE    25000           FAMILY INCOME FUND   50000

REAL ESTATE        7000           EDUCATION FUND        9000

SECURITIES            0           RETIREMENT FUND      41000

CASH ON HAND      10000           UNINSURED DEBTS          0

OTHER ASSETS          0           SPOUSE INCOME FUND   10000

------------------------------------------------------------
TOTAL                 42000                           110000
------------------------------------------------------------


          LIFE INSURANCE NEEDED = $ 68000
```

DYNACOMP

Quality software for:*

ATARI — **TRS-80 (Level II)****
PET/CBM — **NORTH STAR*****
APPLE II Plus — **CP/M Disks/Diskettes**
OSBORNE-1 — (See Availability box MBASIC/CBASIC)
ALTAIR**** — **SUPERBRAIN*******

CARD GAMES

NEW **BRIDGE MASTER (North Star only)** — **Price: $21.95 Diskette**
If you liked DYNACOMP'S BRIDGE 2.0, you will absolutely love BRIDGE MASTER. BRIDGE MASTER is a comprehensive bridge program designed to provide hours of challenging competition. Bidding features include the Blackwood convention, Stayman convention, pre-emptive openings, and recognition of demand bids and jump-shift responses. After playing a specific hand, you may replay the same hand, with the option of switching cards with your computer opponents. This feature allows you to compare your bidding and playing skills to BRIDGE MASTER. Bonuses for game contracts and slams are awarded as in duplicate bridge. Doubled contracts are scored based upon a computer assigned vulnerability. A score card is displayed at the conclusion of each hand. The score card displays a summary of total hands played, total points scored, number of contracts made and set, and #'s bids made. BRIDGE MASTER is clearly the best computer bridge program available.
DYNACOMP'S previous BRIDGE 2.0 customers may upgrade to BRIDGE MASTER for a nominal charge of $5.00 plus postage and handling (see ordering information box).

NEW **BACCARAT (Atari only)** — **Price: $18.95 Cassette/$22.95 Diskette**
This is the European card game which is the favorite of the Monte Carlo jet set. Imagine yourself at the gaming table with 007 to your left and Goldfinger to your right. Learn and play BACCARAT at your leisure on the Atari. Contains full high resolution color graphics and matching sound. Runs in 16K. Requires one joystick.

NEW **GIN RUMMY (Apple diskette only)** — **Price: $22.95 Diskette**
This is the best micro computer implementation of GIN RUMMY existing. The computer plays exceptionally well, and the HIRES graphics are superb. What else can be said?

POKER PARTY (Available for all computers) — **Price: $17.95 Cassette/$21.95 Diskette**
POKER PARTY is a draw poker simulation based on the book, POKER, by Oswald Jacoby. This is the most comprehensive version available for microcomputers. The party consists of yourself and six other (computer) players. Each of these players (you will get to know them) has a different personality in the form of a varying propensity to bluff or fold under pressure. Practice with POKER PARTY before going to that expensive game tonight! Apple cassette and diskette versions require a 32 K (or larger) Apple II.

NEW **GO FISH (Available for all computers)** — **Price: $14.95 Cassette/$18.95 Diskette**
GO FISH is a classic children's card game. The opponent is a friendly computer with user inputs that are simple enough for small children to easily master. The Apple and Atari versions employ high resolution graphics for the display of hands. A must for children!

NEW **BLACKJACK COACH (32K TRS-80 only)** — **Price $29.95 Cassette/$33.95 Diskette**
BLAGKJACK COACH is both a game and an educational tool. With this program you may quantitatively test standard and special playing and betting methods, including the several card count schemes. You can simply play, play with the computer as a coach, or statistically test your method under long run automatic play. All the standard player choices are included: insurance, splitting pairs, double downs and surrender (optional). The computer analyzes the technique and provides detailed summary reports which statistically pinpoint the strengths and weaknesses of your play. Don't risk your money at the tables until you have practiced with BLACKJACK COACH!

THOUGHT PROVOKERS

MANAGEMENT SIMULATOR (Atari, North Star, OSBORNE and CP/M only) — **Price: $19.95 Cassette $23.95 Diskette**
This program is both an excellent teaching tool as well as a stimulating intellectual game. Based upon similar games played at graduate business schools, each player or team controls a company which manufacturers three products. Each player attempts to outperform his competitors by setting selling prices, production volumes, marketing and design expenditures etc. The most successful firm is the one with the highest stock price when the simulation ends.

FLIGHT SIMULATOR (Available for all computers) — **Price: $17.95 Cassette/$21.95 Diskette**
A realistic and extensive mathematical simulation of take-off, flight and landing. The program utilizes aerodynamic equations and the characteristics of a real airfoil. You can practice instrument approaches and navigation using radials and compass headings. The more advanced flyer can also perform loops, half-rolls and similar aerobatic maneuvers. Although this program does not employ graphics, it is exciting and very addictive. See the software review in COMPUTRONICS. Runs in 16K Atari.

VALDEZ (Available for all computers) — **Price: $15.95 Cassette/$19.95 Diskette**
VALDEZ is a computer simulation of supertanker navigation in the Prince William Sound/Valdez Narrows region of Alaska. Included in this simulation is a realistic and extensive 256 × 256 element map, portions of which may be viewed using the ship's alphanumeric radar display. The motion of the ship itself is accurately modelled mathematically. The simulation also contains a model for the tidal patterns in the region, as well as other traffic (outgoing tankers and drifting icebergs). Chart your course from the Gulf of Alaska to Valdez Harbor! See the software reviews in 80 Software Critique and Personal Computing.

BACKGAMMON 2.0 (Atari, North Star, OSBORNE and CP/M only) — **Price: $14.95 Cassette/$18.95 Diskette**
This program tests your backgammon skills and will also improve your game. A human can compete against a computer or against another human. The computer can even play against itself. Either the human or the computer can double or generate dice rolls. Board positions can be created or saved for replay. BACKGAMMON 2.0 plays in accordance with the official rules of backgammon and is sure to provide many fascinating sessions of backgammon play.

CHESS MASTER (North Star and TRS-80 only) — **Price: $19.95 Cassette/$23.95 Diskette**
This complete and very powerful program provides five levels of play. It includes castling, en passant captures and the promotion of pawns. Additionally, the board may be preset before the start of play, permitting the examination of "book" plays. To maximize execution speed, the program is written in assembly language (by SOFTWARE SPECIALISTS of California). Full graphics are employed in the TRS-80 version, and two widths of alphanumeric display are provided to accommodate North Star users. See review in onComputing.

FOREST FIRE! (Atari only) — **Price: $14.95 Cassette/$18.95 Diskette**
Using excellent graphics and sound effects, this simulation puts you in the middle of a forest fire. Your job is to direct operations to put out the fire while compensating for changes in wind, weather and terrain. Not protecting valuable structures can result in startling penalties. Life-like variables are provided to make FOREST FIRE! very suspenseful and challenging. No two games have the same setting and there are 3 levels of difficulty.

BLACK HOLE (Apple only) — **Price: $14.95 Cassette/$18.95 Diskette**
This is an exciting graphical simulation of the problems involved in closely observing a black hole with a space probe. The object is to enter and maintain, for a prescribed time, an orbit close to a small black hole. This is to be achieved without coming so near the anomaly that the tidal stress destroys the probe. Control of the craft is realistically simulated using side jets for rotation and main thrusters for acceleration. This program employs Hi-Res graphics and is educational as well as challenging.

NEW **SPACE EVACUATION! (Apple, Atari and TRS-80 only)** — **Price: $15.95 Cassette/$19.95 Diskette**
Can you colonize the galaxy and evacuate the Earth before the sun explodes? Your computer becomes the ship's computer as you explore the universe to relocate millions of people. This simulation is particularly interesting as it combines many of the exciting elements of classic space games with the mystery challenge of ADVENTURE.

MONARCH (Atari only) — **Price: $11.95 Cassette/$15.95 Diskette**
MONARCH is a fascinating economic simulation requiring you to survive an 8-year term as your nation's leader. You determine the amount of acreage devoted to industrial and agricultural use, how much food to distribute to the populace and how much should be spent on pollution control. You will find that all decisions involve a compromise and that it is not easy to make everyone happy. Runs in 16K Atari.

CHOMPELO (Atari only) — **Price: $11.95 Cassette/$15.95 Diskette**
CHOMPELO is really two challenging games in one. One is similar to NIM; you must bite off part of a cookie, but avoid taking the poisoned portion. The other game is the popular board game REVERSI. It fully uses the Atari's graphics capability, and is hard to beat. This package will run on a 16K system.

AVAILABILITY

DYNACOMP software is supplied with complete documentation containing clear explanations and examples. Unless otherwise specified, all programs will run within 16K program memory space (ATARI requires 24K). Except where noted, programs are available on ATARI, PET, TRS-80 (Level II) and Apple (Applesoft) cassette and diskette as well as North Star single density (double density compatible) diskette. Additionally, most programs can be obtained on standard (IBM 3740 single density/double density compatible format) 8" CP/M floppy disks for systems running under MBASIC or CBASIC (for example, Altos, Xerox 820 and many others). 5¼" CP/M diskettes are available for the North Star and Osborne computer systems.

**ATARI, PET/CBM, NORTH STAR, CP/M, IBM, OSBORNE, SUPERBRAIN and XEROX are registered tradenames and/or trade marks.*

***Except where noted, all TRS-80 Model I software is available on cassette (only) for the TRS-80 Model III. Exceptions: VALDEZ, CRIBBAGE, GRAFIX, CHESSMASTER. TRS-80 diskettes are not supplied with either DOS or BASIC.*

****For most North Star disk-based systems. DYNACOMP presently does not support the new North Star Advantage.*

*****For Altair systems having Microsoft BASIC.*

******For SUPERBRAIN systems running under MBASIC or CBASIC (state which).*

DYNACOMP OFFERS THE FOLLOWING

- Widest variety
- Guaranteed quality
- Fastest delivery
- Friendly service
- Free catalog
- Toll free order phone

AND MORE...

STARTREK 3.2 (Available for all computers) — **Price: $11.95 Cassette/$15.95 Diskette**
This is the classic Startrek simulation, but with several new features. For example, the Klingons now shoot at the Enterprise without warning while also attacking starbases in other quadrants. The Klingons also attack with both light and heavy cruisers and move when shot at! The situation is hectic when the Enterprise is besieged by three heavy cruisers and a starbase S.O.S. is received! The Klingons get even! See the software reviews in A.N.A.L.O.G., 80 Software Critique and Game Merchandising.

NEW **LIL' MEN FROM MARS (Atari only)** — **Price: $19.95 Cassette/$23.95 Diskette**
Defend yourself! The little men from Mars are out to get you if you don't get them first. This is a hilarious high resolution animated graphics (arcade) game which exercises much of the Atari's power. Requires one joystick.

SPACE TILT (Apple and Atari only) — **Price: $10.95 Cassette/$14.95 Diskette**
Use the game paddles to tilt the plane of the TV screen to "roll" a ball into a hole in the screen. Sound simple? Not when the hole gets smaller and smaller! A built-in timer allows you to measure your skill against others in this habit-forming action game.

NEW **ESCAPE FROM VOLANTIUM (Atari only)** — **Price: $15.95 Cassette/$19.95 Diskette**
Bring the action and excitement of an arcade into your home with ESCAPE FROM VOLANTIUM! To escape you must maneuver your space ship around obstacles and laser blast the dragon (without being eaten). If he is killed with a direct shot (not just a leg lopped off), a door opens to the outside. However, the door does not stay open indefinitely. If you fail to escape in time, the door closes and a new dragon appears. Sometimes you can smash through the door by repeatedly chipping away at it. Other times it is impervious. At the higher levels of play more obstacles and dragons appear, adding to the excitement. Uses high resolution graphics and sound. Runs in 16K.

ALPHA FIGHTER (Atari only) — **Price: $12.95 Cassette/$16.95 Diskette**
Two excellent graphics and action programs in one! ALPHA FIGHTER requires you to destroy the alien starships passing through your sector of the galaxy. ALPHA BASE is in the path of an alien UFO invasion; let five UFO's get by and the game ends. Both games require the joystick and get progressively more difficult the higher you score! ALPHA FIGHTER will run on 16K systems.

THE RINGS OF THE EMPIRE (Atari only) — **Price: $14.95 Cassette/$18.95 Diskette**
The empire has developed a new battle station protected by rotating rings of energy. Each time you blast through the rings and destroy the station, the empire develops a new station with more protective rings. This exciting game runs on 16K systems, employs extensive graphics and sound and can be played by one or two players.

INTRUDER ALERT (Atari only) — **Price: $14.95 Cassette/$18.95 Diskette**
This is a fast paced graphics game which places you in the middle of the "Dreadstar" having just stolen its plans. The droids have been alerted and are directed to destroy you at all costs. You must find and enter your ship to escape with the plans. Five levels of difficulty are provided. INTRUDER ALERT requires a joystick and will run on 16K systems.

NEW **MIDWAY (Atari only)** — **Price: $14.95 Cassette/$18.95 Diskette**
MIDWAY is an exciting extension of the game of Battleship. It mixes the challenges of strategy and chance. Your opponent can be another human or the computer. Color graphics and sound are both included. Runs in 16K.

TRIPLE BLOCKADE (Atari only) — **Price: $14.95 Cassette/$18.95 Diskette**
TRIPLE BLOCKADE is a two-to-three player graphics and sound action game. It is based on the classic video arcade game which millions have enjoyed. Using the Atari joysticks, the object is to direct your blockading line around the screen without running into your opponent(s). Although the concept is simple, the combined graphics and sound effect lead to "high anxiety".

GAMES PACK I (Available for all computers) — **Price: $10.95 Cassette/$14.95 Diskette**
GAMES PACK I contains the classic computer games of BLACKJACK, LUNAR LANDER, CRAPS, HORSERACE, SWITCH and more. These games have been combined into one large program for ease in loading. They are individually accessed by a convenient menu. This collection is worth the price just for the DYNACOMP version of BLACKJACK.

GAMES PACK II (Available for all computers) — **Price: $10.95 Cassette/$14.95 Diskette**
GAMES PACK II includes the games CRAZY EIGHTS, JOTTO, ACEY-DUCEY, LIFE, WUMPUS and others. As with GAMES PACK I, all the games are loaded as one program and are called from a menu. You will particularly enjoy DYNACOMP's version of CRAZY EIGHTS.

Why pay $7.95 or more per program when you can buy a DYNACOMP collection for just $10.95?

MOON PROBE (Atari and North Star only) — **Price: $11.95 Cassette/$15.95 Diskette**
This is an extremely challenging "lunar lander" program. The user must drop from orbit to land at a predetermined target on the moon's surface. You control the thrust and orientation of your craft plus direct the rate of descent and approach angle. Runs in 16K Atari.

SPACE TRAP (Atari only, 16K) — **Price: $14.95 Cassette/$18.95 Diskette**
This galactic "shoot'em up" arcade game places you near a black hole. You control your spacecraft using the joystick and attempt to blast as many of the alien ships as possible before the black hole closes about you.

NEW **CHIRP INVADERS (PET/CBM only)** — **Price: $14.95 Cassette/$18.95 Diskette**
CHIRP INVADERS is an addictive game using action graphics. A Federation space station must be reached before the Chirps conquer the Earth. Stationary obstacles, moving meteors, and the attacking Chirps must all be avoided for a successful journey. Good luck.

NEW **SUPER SUB CHASE (Atari only)** — **Price: $19.95 Cassette/$23.95 Diskette**
SUPER SUB CHASE simulates a search and destroy mission. Set your course and keep an eye on the sonar readings as you hunt for the hidden submarine. Set the depth charge explosion depth and watch them sink towards the sub. This is an addictive game which takes advantage of the Atari's graphics and sound capabilities. One or two players. Joystick(s) required.

ADVENTURE

CRANSTON MANOR ADVENTURE (North Star and CP/M only) — **Price: $19.95 Diskette**
At last! A comprehensive Adventure game for North Star and CP/M systems. CRANSTON MANOR ADVENTURE takes you into mysterious CRANSTON MANOR where you attempt to gather fabulous treasures. Lurking in the manor are wild animals and robots who will not give up the treasures without a fight. The number of rooms is greater and the associated descriptions are much more elaborate than the current popular series of Adventure programs, making this game the top in its class. Play can be stopped at any time and the status stored on diskette. Not available in 5¼" CP/M format.

GUMBALL RALLY ADVENTURE (North Star only, 48K) — **Price: $21.95 Diskette**
Take part in this outlaw race from the east coast to the west coast. The goal is to find your way to the finish line while maintaining the highest possible speed. You may choose one of five cars available at the garage. The choice will affect your speed and range. Remember to take spare parts and don't get caught speeding!

NEW **UNCLE HARRY'S WILL (North Star only, 40K)** — **Price: $24.95 Diskette**
Uncle Harry has died and has left you everything. However, he has neglected to mention where everything is! Instead, his will consists of a poem which contains clues. You will have to travel all over the United States both by car and on foot to solve the puzzle, and there are over 300 locations to probe. Be careful and watch out for red herrings!

SPEECH SYNTHESIS

DYNACOMP is now distributing the new and revolutionary TYPE-'N-TALK^{TM} (TNT) speech synthesizer from Votrax. Simply connect TNT to your computer's serial interface, enter text from the keyboard and hear the words spoken. TNT is the easiest-to-program speech synthesizer on the market. It uses the least amount of memory and provides the most flexible vocabulary available anywhere!

TYPE-'N'-TALK List price $375. DYNACOMP'S price $319.95 plus $5.00 for shipping and handling.

TALK TO ME (T'N'T Atari only, 24K) — **Price: $14.95 Cassette/$18.95 Diskette**
This program presents a superb tutorial on speech synthesis using the Atari 800 and TYPE 'N TALK^{TM}. TALK TO ME will illustrate normal word generation as well as phoneme generation. The documentation includes many helpful programming tips. TALK TO ME has been demonstrated on network (CBS) TV!

MISCELLANEOUS

CRYSTALS (Atari only) — **Price: $ 9.95 Cassette/$13.95 Diskette**
A unique algorithm randomly produces fascinating graphics displays accompanied with tones which vary as the patterns are built. No two patterns are the same, and the combined effect of the sound and graphics are mesmerizing. CRYSTALS has been used in local stores to demonstrate the sound and color features of the Atari. Runs in 16K Atari.

NORTH STAR SOFTWARE EXCHANGE (NSSE) LIBRARY
DYNACOMP now distributes the 23 volume NSSE library. These diskettes each contain many programs and offer an outstanding value for the purchase price. They should be part of every North Star user's collection. Call or write DYNACOMP for details regarding the contents of the NSSE collection.
Price: $9.95 each/$7.95 each (4 or more)
The complete collection may be purchased for $149.95

NEW **5¼" DISKETTES (soft sectored/ten sectored)** — **Price: $39.95/20 Diskettes**
As you might imagine, DYNACOMP purchases diskettes in large quantities and at wholesale prices. We want to pass the savings along to you!

Program 1: Microsoft Version

```
10 REM LIFE INSURANCE PLAN BY DAVID E. PITTS
20 REM 16011 STONEHAVEN DR. HOUSTON TX 77059
30 DEFFNRD(A)=INT((A+999)/1000)*1000
40 DEFFNP1(A)=L-LEN(STR$(A))+1
50 DIMS(18),MI(18),Y(18),C(19):E=0:Y=0
60 PRINTTAB(20);"LIFE INSURANCE PLAN":PRINT:PRINT:GOSUB740
70 PRINT:INPUT"ENTER # OF CHILDREN UNDER 18";N:IFN=0THEN90
80 PRINT:INPUT"MONTHLY INCOME GOAL FOR SPOUSE & CHILDREN";SC
90 PRINT:INPUT"MONTHLY INCOME GOAL FOR SPOUSE'S RETIREMENT";WR
100 PRINT:PRINT"MONTHLY INCOME GOAL FOR SPOUSE BETWEEN CHILD REARING"
110 PRINT"AND RETIREMENT,IF NO CHILDREN <18,THIS APPLIES NOW TO
    RETIREMENT";
120 INPUTMB:IFN=0THENGOSUB760:GOTO190
130 PRINT:INPUT"LUMP SUM EDUCATION FUND FOR EACH CHILD";E
140 PRINT:PRINT:GOSUB740:PRINT"START WITH OLDEST CHILD":PRINT:PRINT
150 FORI=1TON:PRINT"AGE OF CHILD # ";I;:INPUTY(I):Y(I)=18-Y(I):NEXT
160 GOSUB760:INPUT"FOR A SPOUSE & 2 OR MORE CHILDREN";B2
170 INPUT"FOR A SPOUSE & 1 CHILD";B1:INPUT"FOR A STUDENT 18-21";BS
180 INPUT"FOR A FAMILY'S MAXIMUM PERMISSABLE BENEFIT";BF
190 INPUT"FOR SPOUSE STARTING RETIREMENT AT AGE 62";B6:IFN=0THEN300
200 REM CALC OF YEAR BY YEAR MONTHLY BENEFITS WHILE CHILDREN <18
210 FORI=1TOY(N):IFI<=Y(N-1)THENC(I)=C(I)+B2
220 IFI>Y(N-1)ANDI<=Y(N)THENC(I)=C(I)+B1
230 FORJ=1TON:IFI>Y(J)ANDI<=Y(J)+4THENS(I)=S(I)+BS
240 NEXT:NEXT
250 FORI=1TO18:IFS(I)>BFTHENS(I)=BF
260 IFS(I)>0THENS(I)=BF-S(I)
270 IFS(I)=0THENS(I)=C(I)
280 IFS(I)<C(I)THENC(I)=S(I)
290 NEXT
300 GOSUB740:PRINT"IF INTEREST RATE REQUESTED ISN'T KNOWN ENTER '0'"
310 PRINT:INPUT"CURRENT TAX BRACKET IN %";T:T=T/100:PRINT
320 IFT=0THENT=.273
330 INPUT"SAVINGS ACCOUNT INTEREST RATE FOR SURVIVORS ANNUITY";R:R=R/100
340 IFR=0THENR=.055
350 REM BEGINNING OF INCOME FUND COMPUTATIONS
360 R=R*(1-T):MI(1)=C(1):K=1:S(0)=0:X=1:IFN=0THENE=0:GOTO450
370 FORI=1TO19:IFC(I)=MI(K)THEN390
380 S(K)=I-X:X=I:K=K+1:MI(K)=C(I)
390 NEXT:X=0:Y=0
400 FORI=1TOK-1:A=12*(1-(1+R)^-S(I))/R:D=1/(1+R)^X:X=X+S(I)
410 Z=SC-MI(I):IFZ<0THENZ=0
420 MI(I)=Z*A:MI(I)=FNRD(MI(I))*D:Y=Y+MI(I):NEXT:Y=FNRD(Y)
430 REM BEGINNING OF EDUCATION FUND CALCULATIONS
440 ED=E-BS*40:E=0:FORI=1TON:E=E+ED/(1+R)^Y(I):NEXT:E=FNRD(E)
450 INPUT"CURRENT AGE OF SPOUSE";AG:RE=WR-B6:AG=62-AG:IFAG<0THENAG=0
460 REM BEGINNING OF RETIREMENT CALCULATIONS
470 RE=RE*12*(1-(1+R)^-20)/R:RE=FNRD(RE)
480 REM INCOME BETWEEN CHILD-REARING AND RETIREMENT
490 AG=AG-Y(N):IFAG<0THENAG=0
500 WI=MB*12*(1-(1+R)^-AG)/(R*(1+R)^Y(N)):WI=FNRD(WI):GOSUB740
510 REM BEGIN BALANCE SHEET CALCULATIONS
520 PRINT"ENTER FAMILY ASSETS":PRINT:INPUT"LIFE INSURANCE";S:PRINT
530 INPUT"CASH ON HAND";X:Z=X+S:PRINT:INPUT"REAL ESTATE EQUITY";J:Z=J+Z
540 PRINT
```

COMPUTE! Back Issues

Here are some of the applications, tutorials, and games from available back issues of **COMPUTE!**. Each issue contains much, much more than there's space here to list, but here are some highlights:

January, 1981: Load PET Programs Into The Apple II, Player-Missile Graphics for Atari, The Atari DOS, The Kernel of the OSI Operating System, Fixing LOADing Problems on the PET, Spooling with the PET Disk, Expanding KIM.

February, 1981: Simulating PRINT USING, Using the Atari as a Terminal for Telecommunications, Attach a Printer to the Atari, Double Density Graphing on C1P, Commodore Disk Systems, PET Crash Prevention, A 25¢ Apple II Clock.

March, 1981: Machine Language Programming for Beginners, Getting the Most from your PET Cassette Deck, Apple and PASCAL, Flipping your Apple Disk, Designing your own Atari Character Sets, Renumber for Atari, An Atari Disassembler, Six-gun Shootout Game for OSI C1P, PET Machine Language Graphics.

April, 1981: How to be a VIC Expert, Resolving the Applesoft and Hires Graphics Memory Conflicts, Atari SuperCube, String Arrays in Atari, Memory Partition in PET, Pet Relative Files, Working with BASIC 4.0, Commodore File I/O, ROM Expansion for Commodore PET.

May, 1981: Named GOSUB/GOTO in Applesoft, Generating Lower Case Text on Apple II, Copy Atari Screens to the Printer, Disk Directory Printer for Atari, Realtime Clock on Atari, PET BASIC Delete Utility, PET Calculated Bar Graphs, Running 40 Column Programs on a CBM 8032.

June, 1981: Computer Using Educators (CUE) on Software Pricing, Apple II Hires Character Generator, Ever- expanding Apple Power, Color Burst for Atari, Mixing Atari Graphics Modes 0 and 8, Relocating PET BASIC Programs, An Assembler In BASIC for PET, QuadraPET: Multitasking?

July, 1981: Home Heating and Cooling, Animating Integer BASIC Lores Graphics, The Apple Hires Shape Writer, Adding a Voice Track to Atari Programs, Machine Language Atari Joystick Driver, Four Screen Utilities for the PET, Saving Machine Language Programs on PET Tape Headers, Commodore ROM Systems, The Voracious Butterfly on OSI.

August, 1981: Minimize Code and Maximize Speed, Apple Disk Motor Control, A Cassette Tape Monitor for the Apple, Easy Reading of the Atari Joystick, Blockade Game for the Atari, Atari Sound Utility, The CBM "Fat 40," Keyword for PET, CBM/PET Loading, Chaining, and Overlaying.

September, 1981: The Column Calculator, What is a Modem and Why Do I Need One?, PET, Apple, Atari: On Speaking Terms, A Tape "EXEC" for Applesoft, A Self-altering Program for Apple II, Positioning P/M Graphics and Regular Graphics in Memory, An Atari BASIC Sort, Shoot, an Arcade Game for Atari, Exploring OSI's Video Routine, PET Tape Append and Renumber, All About LOADing PET Cassettes.

October, 1981: Automatic DATA Statements for CBM and Atari, VIC News, Undeletable Lines on Apple, PET, VIC, Budgeting on the Apple, Switching Cleanly from Text to Graphics on Apple, Atari Cassette Boot-tapes, Atari Variable Name Utility, Atari Program Library, Train your PET to Run VIC Programs, Interface a BSR Remote Control System to PET, A General Purpose BCD to Binary Routine, Converting to Fat-40 PET.

November, 1981: SuperPet: A Preview, Japanese Micros: A First Look, Introduction to Binary Numbers, An Apple Primer, Page Flipper for Apple, An Atari Database System, A Program for Writing Programs on the Atari, Atari Textplot, OSI Relocation, The PET Speaks, Inversion Partitioning, A Personal News Service on PET, Bits, Bytes, and Basic Boole.

December, 1981: Saving Fuel $$ (Multiple Computers: versions for Apple, PET, and Atari), Unscramble Game (multiple computers), Maze Generator (multiple computers), Animating Applesoft Graphics, A Simple Printer Interface for the Apple II, A Simple Atari Wordprocessor, Adding High Speed Vertical Positioning to Atari P/M Graphics, OSI Supercursor, A Look At SuperPET, Supermon for PET/CBM, PET Mine Maze Game.

January, 1982: Invest (multiple computers), Developing a Business Algorithm (multiple computers), Apple Addresses, Lowercase with Unmodified Apple, Cryptogram Game for Atari, Superfont: Design Special Character Sets on Atari, PET Repairs for the Amateur, Micromon for PET, Self-modifying Programs in PET BASIC, Tinymon: a VIC Monitor, Vic Color Tips, VIC Memory Map, ZAP: A VIC Game.

February, 1982: Insurance Inventory (multiple computers), Musical Transposition (multiple computers), Multitasking Emulator (multiple computers), Disassemble Apple Programs from BASIC, Plotting Polar Graphs on Apple, Atari P/M Graphics Made Easy, Atari PILOT, Put A Rainbow in your Atari, Marquee for PET, PET Disk Disassembler, VIC Paddles and Keyboard, VIC Timekeeping.

March, 1982: Word Hunt Game (multiple computers), Infinite Precision Multiply (multiple computers), Atari Concentration Game, VIC Starfight Game, CBM BASIC 4.0 To Upgrade Conversion Kit, Apple Addresses, VIC Maps, EPROM Reliability, Atari Ghost Programming, Atari Machine Language Sort, Random Music Composition on PET, Comment Your Apple II Catalog.

April, 1982: Track Down Those Memory Bugs (multiple computers), Shooting Stars Game (multiple computers), Intelligent Input Subroutines (multiple computers), Ultracube for Atari, Customizing Apple's Copy Program, Using PET/CBM In The High School Physics Lab, Grading Exams on a Microcomputer (multiple computers), Atari Mailing List, Renumber VIC Programs The Easy Way, Browsing the VIC Chip, Disk Checkout for PET/CBM.

Back issues are $3.00 each or six for $15.00. Price includes freight in the US. Outside the US add $1.00 per magazine ordered for surface postage. $3.00 per magazine for air mail postage. All back issues subject to availability.

In the Continental US call
TOLL FREE 800-334-0868
(In NC Call 919-275-9809)

Or write to **COMPUTE!** Back Issues, P.O. Box 5406, Greensboro, NC 27403 USA. Prepayment required in US funds. MasterCard, Visa and American Express accepted. North Carolina Residents add 4% sales tax.

```
550 INPUT"SECURITIES";H:Z=H+Z:PRINT:INPUT"OTHER ASSETS";O:Z=O+Z:GOSUB740
560 INPUT"UNINSURED DEBTS, OTHER THAN HOME MORTGAGE";Q:PRINT:GOSUB740
570 PRINT:PRINTTAB(22);"BALANCE SHEET":PRINT
580 PRINTTAB(5);"FAMILY ASSETS";TAB(35);"FAMILY LIABILITIES"
590 PRINT:PRINTTAB(5);"LIFE INSURANCE";:L=26:V=S:GOSUB780:PRINTS;
600 PRINTTAB(35);"FAMILY INCOME FUND";:L=59:V=Y:GOSUB780:PRINTY
610 PRINT:PRINTTAB(5);"REAL ESTATE";:L=26:V=J:GOSUB780:PRINTJ;
620 PRINTTAB(35);"EDUCATION FUND";:L=59:V=E:GOSUB780:PRINTE
630 PRINT:PRINTTAB(5);"SECURITIES";:L=26:V=H:GOSUB780:PRINTH;
640 PRINTTAB(35);"RETIREMENT FUND";:L=59:V=RE:GOSUB780:PRINTRE
650 PRINT:PRINTTAB(5);"CASH ON HAND";:L=26:V=X:GOSUB780:PRINTX;
660 PRINTTAB(35)"UNINSURED DEBTS";:L=59:V=Q:GOSUB780:PRINTQ
670 PRINT:PRINTTAB(5);"OTHER ASSETS";:L=26:V=O:GOSUB780:PRINTO;
680 PRINTTAB(35);"SPOUSE INCOME FUND";:L=59:V=WI:GOSUB780:PRINTWI
690 X=Y+E+RE+WI+Q
700 PRINT:GOSUB740:PRINT"TOTAL";:L=26:V=Z:GOSUB780:PRINTZ;
710 L=59:V=X:GOSUB780:PRINTX
720 GOSUB740:PRINT:PRINTTAB(15);"LIFE INSURANCE NEEDED = $";X-Z
730 END
740 PRINT"------------------------------------------------------------"
750 RETURN
760 GOSUB740:PRINT"ENTER SOCIAL SECURITY (OR OTHER PROGRAM) MONTHLY"
770 PRINT"SURVIVOR BENEFITS:":PRINT:RETURN
780 V=FNP1(V):PRINTTAB(V);:RETURN
790 REM S( )=INCOME WHILE STUDENT 18-21
800 REM C(I)=MONTHLY INCOME FROM RETIREMENT PLAN FOR YEAR I
810 REM      FOR WIDOW AND CHILDREN UNDER 18.
820 REM Y(I)=YEARS UNTIL CHILD I IS 18
830 REM N=NUMBER OF CHILDREN UNDER 18
840 REM B2=SURVIVOR BENEFITS FOR SPOUSE AND 2 OR MORE CHILDREN
850 REM B1= SURVIVOR BENEFITS FOR SPOUSE AND 1 CHILD
860 REM BS= SURVIVOR BENEFITS FOR STUDENT 18-21
870 REM BF=FAMILY'S MAX BENEFIT
880 REM B6=SPOUSE'S RETIREMENT BENEFIT AT AGE 62
890 REM SC=MONTHLY GOAL FOR SPOUSE & CHILDREN
900 REM WR=MONTHLY GOAL FOR SPOUSE (E.G. WIDOW) RETIREMENT
910 REM MB=MONTHLY GOAL FOR SPOUSE BETWEEN CHILDREN AND RETIREMENT
920 REM E=LUMP SUM EDUCATIN FUND FOR EACH CHILD
930 REM MI( )=WORKING VARIABLE FOR INCOME CALCULATIONS
940 REM A=MULTIPLIER FOR ANNUITY FOR MONTHLY INCOME FOR SURVIVORS
950 REM D=DISCOUNT MULTIPLIER FOR MONEY NEEDED IN FUTURE
960 REM Y=FAMILY INCOME FUND
970 REM ED=TOTAL BENEFITS PAID TO EACH STUDENT
980 REM E=LUMP SUM EDUCATION FUND
990 REM RE=SPOUSE'S RETIREMENT FUND
1000 REM AG=# YEARS TO 62 FOR SPOUSE
1010 REM WI= WIDOW'S (SPOUSE'S) INCOME BETWEEN CHILDREN AND RETIREMENT
1020 REM Z=TOTAL OF FAMILY ASSETS
1030 REM K=#OF HOMOGENOUS PERIODS IN YEAR BY YEAR TABLE OF INCOME
```

Program 2: VIC-20 Version

```
570 PRINT:PRINT"    BALANCE SHEET":
    PRINT
600 PRINT"FAMILY INCOME":PRINT"FUND
    ";TAB(17);Y
610 PRINT"REAL ESTATE";TAB(17);J
620 PRINT"EDUCATION FUND";TAB(17);E
630 PRINT"SECURITIES";TAB(17);H
640 PRINT"RETIREMENT FUND";TAB(17);
    RE
650 PRINT"CASH ON HAND";TAB(17);X
```

```
660 PRINT"UNINSURED DEBTS";TAB(17);
  Q
670 PRINT"OTHER ASSETS";TAB(17);O
680 PRINT"SPOUSE INCOME":PRINT"FUND
  ";TAB(17);WI
700 PRINT"TOTAL";Z
710 PRINT"       ";X
720 GOSUB740:PRINT:PRINT"LIFE INSUR
  ANCE":PRINT"NEEDED =$";X-Z

740 PRINT"---------------------"
```

Program 3: Atari Version

```
30 FNRD=2000
60 POKE 85,10:? "LIFE INSURANCE PLAN":?
:? :GOSUB 740
70 ? :? "ENTER # OF CHILDREN UNDER 18";:
INPUT N:IF N=0 THEN 90
80 ? :? "MONTHLY INCOME GOAL FOR SPOUSE
AND     CHILDREN";:INPUT SC
90 ? :? "MONTHLY INCOME GOAL FOR SPOUSE'
S":? "RETIREMENT";:INPUT WR
100 ? :? "MONTHLY INCOME GOAL FOR SPOUSE
":? "BETWEEN CHILD REARING"
110 ? "AND RETIREMENTM IF NO CHILDREN <1
8":? "THIS APPLIES NOW TO RETIREMENT";
130 ? :? "LUMP SUM EDUCATION FUND FOR EA
CH CHILD";:INPUT E
150 FOR I=1 TO N:? "AGE OF CHILD # ";I;:
INPUT TEMP:Y(I)=18-TEMP:NEXT I
160 GOSUB 760:? "FOR A SPOUSE & 2 OR MOR
E CHILDREN";:INPUT B2
170 ? "FOR A SPOUSE & 1 CHILD";:INPUT B1
:? "FOR A STUDENT 18-21";:INPUT BS
180 ? "FOR A FAMILY'S MAXIMUM PERMISSABL
E BENEFIT";:INPUT BF
190 ? "FOR SPOUSE STARTING RETIREMENT AT
 AGE 62";:INPUT B6:IF N=0 THEN 300
240 NEXT J:NEXT I
290 NEXT I
310 ? :? "CURRENT TAX BRACKENT IN %";:IN
PUT T:T=T/100:PRINT
330 ? "SAVINGS ACCOUNT INTEREST RATE FOR
":? "SURVIVORS ANNUITY";:INPUT R:R=R/100

390 NEXT I:X=0:Y=0
420 MI(I)=Z*A:TEMP=MI(I):GOSUB FNRD:MI(I
)=TEMP*D:Y=Y+MI(I):NEXT I:TEMP=Y:GOSUB F
NRD:Y=TEMP
440 ED=E-BS*40:E=0:FOR I=1 TO N:E=E+ED/(
1+R)^Y(I):NEXT I:TEMP=E:GOSUB FNRD:E=TEM
P
450 ? "CURRENT AGE OF SPOUSE";:INPUT AG:
RE=WR-B6:AG=62-AG:IF AG<0 THEN AG=0
470 RE=RE*12*(1-(1+R)^-20)/R:TEMP=RE:GOS
UB FNRD:RE=TEMP
500 WI=MB*12*(1-(1+R)^-AG)/(R*(1+R)^Y(N)
):TEMP=WI:GOSUB FNRD:WI=TEMP:GOSUB 740
520 ? "ENTER FAMILY ASSETS":? :? "LIFE I
NSURANCE";:INPUT S:?
530 ? "CASH ON HAND";X:Z=X+S:? :? "REAL
ESTATE EQUITY";:INPUT J:Z=J+Z
550 ? "SECURITIES";:INPUT H:Z=H+Z:? :? "
OTHER ASSETS";:INPUT O:Z=O+Z:GOSUB 740
560 ? "UNINSURED DEBTS, OTHER THAN":? "H
OME MORTGAGE";:INPUT Q:? :GOSUB 740
570 ? :POKE 85,11:? "BALANCE SHEET":?
580 ? "FAMILY ASSETS";:POKE 85,17:? "FAM
ILY LIABILITIES"
590 ? :? "LIFE INSURANCE ";S
600 POKE 85,17:? "FAMILY INCOME FUND ";Y

610 ? :? "REAL ESTATE ";J;
620 POKE 85,17:? "EDUCATION FUND ";E
630 ? :? "SECURITIES ";H;
640 POKE 85,17:? "RETIREMENT FUND ";RE
650 ? :? "CASH ON HAND ";X;
660 POKE 85,17:? "UNINSURED DEBTS ";Q
670 ? :? "OTHER ASSETS ";O;
680 POKE 85,17:? "SPOUSE INCOME FUND";:? WI
700 ? :GOSUB 740:? "TOTAL ";Z
710 ? X
720 GOSUB 740:? :POKE 85,7:? "LIFE INSUR
ANCE NEEDED =$";X-Z
740 ? "--------------------------------
---"
760 GOSUB 740:? "ENTER SOCIAL SECURITY":
? "(OR OTHER PROGRAM) MONTHLY"
770 ? "SURVIVOR BENEFITS":? :RETURN
2000 TEMP=INT((TEMP+999)/1000)*1000:RETU
RN
```

©

Some Speculations On The Well-Programmed Game

Myron Miller
Indianola, PA

I would like to expand on David D. Thornburg's excellent commentary "Computers And Society: Some Speculations On The Well-Played Game..." (**COMPUTE!**, July, 1981, #14, pp. 12-16). Mr. Thornburg dealt with the educational value of playing games. I would like to consider the educational value of programming games. A computer is a versatile educational device.

Computer literacy is an awareness of the use of computers, though not necessarily a formal education in computer science. It will become an extremely desirable secondary skill as inexpensive microcomputer technology continues to invade non-computer related career fields. Programming is the essence of computer literacy.

Why Games?

A program is the application of machine logic, usually through a high level language (BASIC, PASCAL, etc.), to solve a problem or perform a certain task. Games, while perhaps not the most important task, are a very effective and versatile medium for learning programming. Games offer some advantages that may be lacking in the more *serious* forms of programming. Games can be simple, or they can be rather complex. You can always think up a game that can match your level of programming ability. However, an effective checkbook balance program may be beyond your capabilities as a beginner and yet, later, prove too simple. Games can be complex enough to use all of the power in your computer. The average home user may have very little use for trigonometric and other higher math functions in normal home applications, but these functions can often be utilized in screen plots for games. The same is true for machine language and graphics.

Games do not require expensive peripherals such as disk drives or printers. Games have a clear goal, they exercise your creativity, and they can be very motivating especially for children. A seventh grader may not be enthusiastic about writing a financial program, but blasting Klingons might prove interesting. The quality of the game itself is secondary, but it must be well-programmed. If the game is a real bore, who cares? What did you *learn* from the program?

Let's take a look at one way to program and, at the same time, develop a simple game program. A good program is nurtured – it is thought out and planned well in advance of actually typing the program into the machine's memory. Good programming procedure might be divided into five distinct steps. You do not need your computer until step five.

Step I. Creation

The first step is creating an application or game. This can be difficult because it requires creativity and vision. You might think of this step in terms of "I would like the computer to do the following: ____________." (You fill in the blank.) This is the brainstorming step. Let your imagination run free. Program creation should not be dominated with questions such as: "Will this work?", or "How can I do this?" Rather, think in terms of unrestricted possibilities. Don't allow concern for the implementation of the program to stifle a valuable creative effort. If something is a bit farfetched, it can be pared down later. But if it is cut at its inception, a valuable idea may be lost forever.

Example 1 shows the creation of the "High/Low Computer Game." This is a simple game, but it will demonstrate the principles that we are discussing. Note that the objectives of the program are very general. This step is a creative effort. Thus, at this point, we do not want specific details. We have room to expand the program or limit it. Note that there is not a hint as to *how* the computer will accomplish the task – the end result of program creation is a rough draft of *what* the program will do.

Example 1. Program Creation of the High/Low Computer Game.

I would like the computer to do the following: "Think" of a number, and ask the user to guess what the number is. If the user's guess is correct, the computer will indicate so, and "think" of another number. If the user's guess is too high or too low, the computer will indicate so, and the user will continue to guess until the correct number is obtained.

NEECO

WHY BUY FROM THE BEST?

Service... Support...
Software...

commodore

16K B (16K RAM-40 Column) - Lim. Qty	$ 995
32K B (32K RAM-40 Clm.) - Lim. Qty	$1295
4016 (16K RAM 4.0 Basic-40 Clm.)	$ 995
4032 (32K RAM 4.0 Basic-40 Clm.)	$1295
8032 (32K RAM 4.0 Basic-80 Clm.)	$1495
8050 Dual Disk (1 Meg Storage)	$1795
4040 Dual Disk (343K Storage)	$1295
8010 IEEE Modem	$ 280
C2N Cassette Drive	$ 75
CBM - IEEE Interface Cable	$ 40
IEEE - IEEE Interface Cable	$ 50
VIC 20 Home/Personal Computer	$ 295

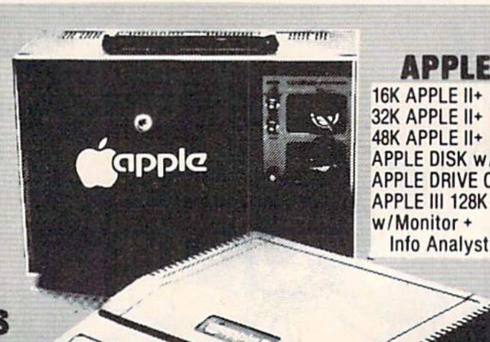
CALL NEECO FOR ANY OF YOUR COMMODORE COMPUTER NEEDS

NEC SPINWRITER PRINTERS

5530 (Parallel)	$3055
5510 (Serial)	$3055
5520 (KSR-Serial)	$3415
Tractor Option	$ 225

EPSON PRINTERS

MX-80 PRINTER	$ 645
MX-80 FT	$ 745
MX-100	$ 945
MX-70	$ 459
INTERFACE CARDS	
8141 (RS-232)	$ 75
8150 (2K Buffered RS-232)	$ 150
8161 (IEEE 488)	$ 55
8131 (Apple Card)	$ 85
8230 (Apple Card)	$ 25
8220 (TRS-80 Cable)	$ 35



APPLE

16K APPLE II+	$1330
32K APPLE II+	$1430
48K APPLE II+	$1530
APPLE DISK w/3.3 DOS	$ 650
APPLE DRIVE Only	$ 490
APPLE III 128K - In Stock! w/Monitor + Info Analystpak	$4740

DIABLO 630 PRINTER

DIABLO 630 - Serial - RS-232	$2710
Tractor Option	$ 250

AMDEK MONITORS

Video 100 12" B+W	$ 179
Video 300 12" Green	$ 249
Color I 13" Low Res	$ 449
Color II 13" High Res	$ 999

INTERTEC COMPUTERS

64K Superbrain (360 Disk Storage), CP/M™	$3495
64K QD Superbrain (700K Disk Storage), CP/M™	$3995

*CP/M is a registered trademark of Digital Research.

ATARI COMPUTERS

Atari 400 (16K RAM)	$ 399
Atari 800 (32K RAM) - good thru 8/31	$1080
Atari 410 RECORDER	$ 89.95
Atari 810 DISK DRIVE	$ 599.95

NEECO carries all available ATARI Software and Peripherals.

PROFESSIONAL SOFTWARE

WordPro 1 8K	$ 29.95
WordPro 3 (40 Clm.)16K	$ 199.95
WordPro 3+	$ 295
WordPro 4 (80 Clm.) 32K	$ 375
WordPro 4+	$ 450

JUST A SAMPLE OF THE MANY PRODUCTS WE CARRY, CALL US FOR OUR NEW 60-PAGE CATALOG.
WE WILL MATCH SOME ADVERTISED PRICES ON CERTAIN PRODUCTS LISTED UNDER SIMILAR "IN STOCK" CONDITIONS.

NEECO
679 HIGHLAND AVE.
NEEDHAM, MA 02194

(617) 449-1760
Telex: 951021

MON-FRI 9:00 - 5:00
MasterCharge and VISA Accepted

Step II. Definition

Program creation (Step I) was an exercise for the imagination, but program definition (Step II) is concerned with reality. Program definition will define *exactly what* the program will do – again, not *how* but *what*. Step II considers that there are real limitations in the abilities of computers and human beings. Program definition will convert the desired effects of the program creation into real and possible objectives. Programming, like a trip in a car, requires that you know where you are going first, then you can figure out how to get there. Occasionally, an improvement will arise during the later stages of the programming effort, and such changes should be incorporated into the program. This should, however, be an exception rather than a rule.

Example 2 shows the program definition of the High/Low Computer Game. Note that the number that the computer "thinks" has been limited to an integer (whole number) between 1 and 100. Also note that specific responses by the computer have replaced the general statements found in the program creation.

Example 2. Program Definition of the High/Low Computer Game

Objectives:

1. The program will 'think' of an integer between 1 and 100.

2. The program will display: "I am thinking of a number between 1 and 100." "Can you guess the number?" "Please enter your guess."

3. The user enters his guess on the keyboard.

4. If the guess is:

- **a.** Correct – the program will display: "Correct!" The program will return to the beginning, and will "think" of another integer....
- **b.** Too Low – the program will display: "Too Low!" "Please try again." The program will return to "Please enter your guess."...
- **c.** Too High – the program will display: "Too High!" Please try again." The program will return to "Please enter your guess"...

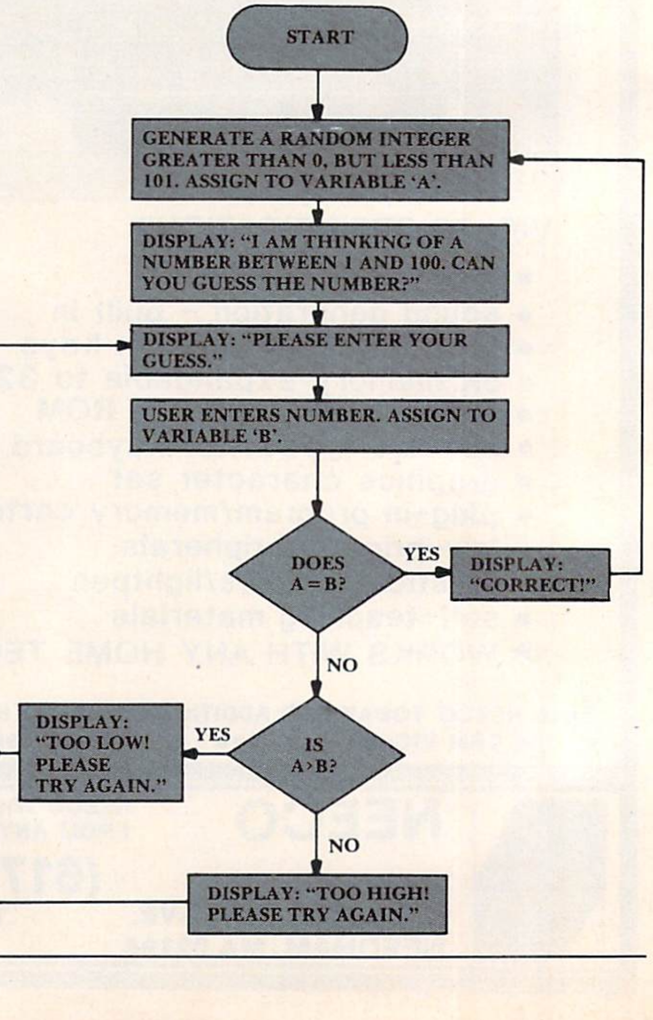
5. *b* and *c* will continue, guiding the user to the correct value, until the correct value is entered. Then *a* will display "Correct!," and repeat the program.

Step III. Solution

Thus far we have determined the "whats" of the program. Now we can develop the "hows." The objectives of the program definition (Step II) are actually a series of problems, and we must develop a series of solutions – a set of algorithms (a method, or sequence of operations) that will satisfy the objectives in Step II. We must also develop a "structure" for the program. We must look at the objectives in Step II and ask ourselves: "What needs to be done first?," "How do we solve this?," "What needs to be done second?," etc.... Then we must ask ourselves: "How do we flow from the first item to the second and to the third?',, etc....

I prefer the use of a flowchart for this step (see Example 3). Each box contains an operation or a group of related operations; the arrows indicate the flow from one operation to the next. The diamond shaped boxes are "decision making" or "test" operations. Don't go overboard when flowcharting – you may end up with a beautiful but useless piece of computer dogma. A flowchart is for human beings, not machines. Thus, it should be written in drab old English, not a computer language, for two reasons 1) a flowchart should be adaptable to any machine, and to almost any language; 2) it is very easy to get lost in a computer language. A good flowchart written in English can help you out of the woods, when you are debugging a program. If the flowchart is written in a computer language, you have two collections of gibberish.

Example 3. Program Solution in flowchart format for the High/Low Computer Game.



When you have completed your flowchart, you should pretend that you are a computer and "run" through your chart. Just follow the arrows! At decision-making operations (diamond shaped boxes), be sure to execute all of the possible conditions. Your flowchart "run" should satisfy the objectives determined in Step II. Look for illogical flow, or behavior that was not intended. Look for redundant operations that can be eliminated or combined into a subroutine. A flowchart is really a logic map. If your logic is correct, your flowchart "run" will accomplish the objectives stated in the program definition (Step II). If you find any errors, they should be corrected before proceeding.

Don't be afraid to make mistakes. If you are lucky you will make lots of them, and each one will burn a lesson into your memory.

In Example 3, we have a flowchart for the High/Low Computer Game. Note the use of English, and the use of yes/no to mark the flow of the conditions at the decision-making boxes. Try a flowchart "run"; you will find that all of the program objectives have been satisfied, and the logic is in order.

One last comment: if you don't like them, don't use them. You can outline the operations or implement some kind of numerical scheme. What is needed is a guide showing the algorithms and the logic flow. Flowcharting is one method, but not the only method, or necessarily the best method. The best method is whatever *you* find most convenient.

Step IV. Translation

We are at the step in which most beginners want to start coding the program into the resident language of the machine. Don't fire up your computer just yet, we will still be working with a pencil and paper. There are a number of reasons to write your program on paper, then code it into the machine. Primarily, it is easier to work with paper than a 25 or 16 line "window" provided by the screen format. Also, you learn by your mistakes; a written listing will record the mistake, and you can enter the correction on the listing. This way you will always have the error and the correction for future reference. Don't be afraid to make mistakes. If you are lucky you will make lots of them, and each one will burn a lesson into your memory.

Program translation is simply the process of converting the algorithms of the program solution (Step III) into the computer language used by your machine. If you did a good job in Step III, this step should be fairly straightforward. Naturally, you will have to be careful about the syntax (grammar of the language) and the rules for the machine to be used. Your flowchart or other solution medium should guide you from the beginning to the end of the program. When writing a listing, it is good practice to allow plenty of space in between each line for corrections or additional lines. Also, if you are not sure that a certain operation will work, test it on your computer. Whip up a "mini-program" that will use the operation in the same manner as desired in the listing. Test mini-programs can save you a lot of debugging (removal of errors) later on. When your listing is completed, again, pretend that you are a computer and execute a listing "run." Look for bad syntax, illogical operation, incorrect flow, and redundant steps. Your listing should comply with the objectives of Step II, and the flow in Step III.

Example 4 is the program translation of the High/Low Computer Game written in BASIC. The program is written for simplicity and clarity, rather than efficiency and design elegance. Note the frequent use of REM (remarks) statements. REM statements appear only in the listing, not during the execution of the program. They make a program easier to read and follow and, as such, should be used generously throughout the listing. REM statements do, however, consume memory space and increase program execution time. Thus, in

Example 4. Program Translation for the High/Low Computer Game.

Note: See if you can find the two bugs inserted into the program. (Program is written for a Commodore PET.)

```
 10 PRINT "HIGH/LOW COMPUTER GAME."
 20 LET A=INT(100*RND(1)+1)
 30 REM 'A' IS THE NUMBER THAT THE
    COMPUTER IS THINKING.
 40 PRINT "I AM THINKING OF A NUMBER
    BETWEEN 1 AND 100. CAN YOU GUESS THE
    NUMBER?"
 50 PRINT "PLEASE ENTER YOUR GUESS."
 60 INPUT B
 70 REM 'B' IS THE USER'S GUESS.
 80 IF A=B THEN GOTO 140
 90 REM TEST FOR CORRECT GUESS
100 IF A>B THEN GOTO 160
110 REM TEST FOR LOW GUESS
120 PRINT "TOO HIGH! PLEASE TRY AGAIN."
130 GOTO 20
140 PRINT "CORRECT!"
150 GOTO 20
160 PRINT "TOO LOW! PLEASE TRY AGAIN."
170 GOTO 20
```

programs where memory space or execution speed is critical, they should be avoided. Another point: the line numbers increase in increments of 10 (10, 20, 30...). This allows additional lines to be added in between the existing lines without having to renumber the remainder of the program.

Two bugs have been intentionally inserted into the listing in Example 4. Perform a listing run, and see if you can find the bugs. Hint: use the flowchart in Example 3.

Step V. Entering and Debugging

Plug in your computer and fire it up. When you are satisfied with your listing, type it into your machine's memory. I like to debug as I go, so about every 20 lines or so, I run the program. This will point out any syntax errors, and may indicate a logic error. It can save you a lot of debugging later by pointing out a mistake before it becomes compounded by additional future operations. *Caution!* Before you run a partial program, be sure that you do not have any functions that will cause the computer to crash (loss of control of the computer usually caused by the processor getting caught in an endless loop). Machine language is very susceptible to crashing and must be properly terminated before executing. If you have to shut down and power up again to recover control, you will lose your program! Therefore, get the program into mass storage (tape or disk) often – about every 20 lines. If you then have a crash or a power failure, you will lose only a small portion of the program and not the entire effort. Remember, Random Access Memory is volatile – the contents (your program) vanish if the power is interrupted.

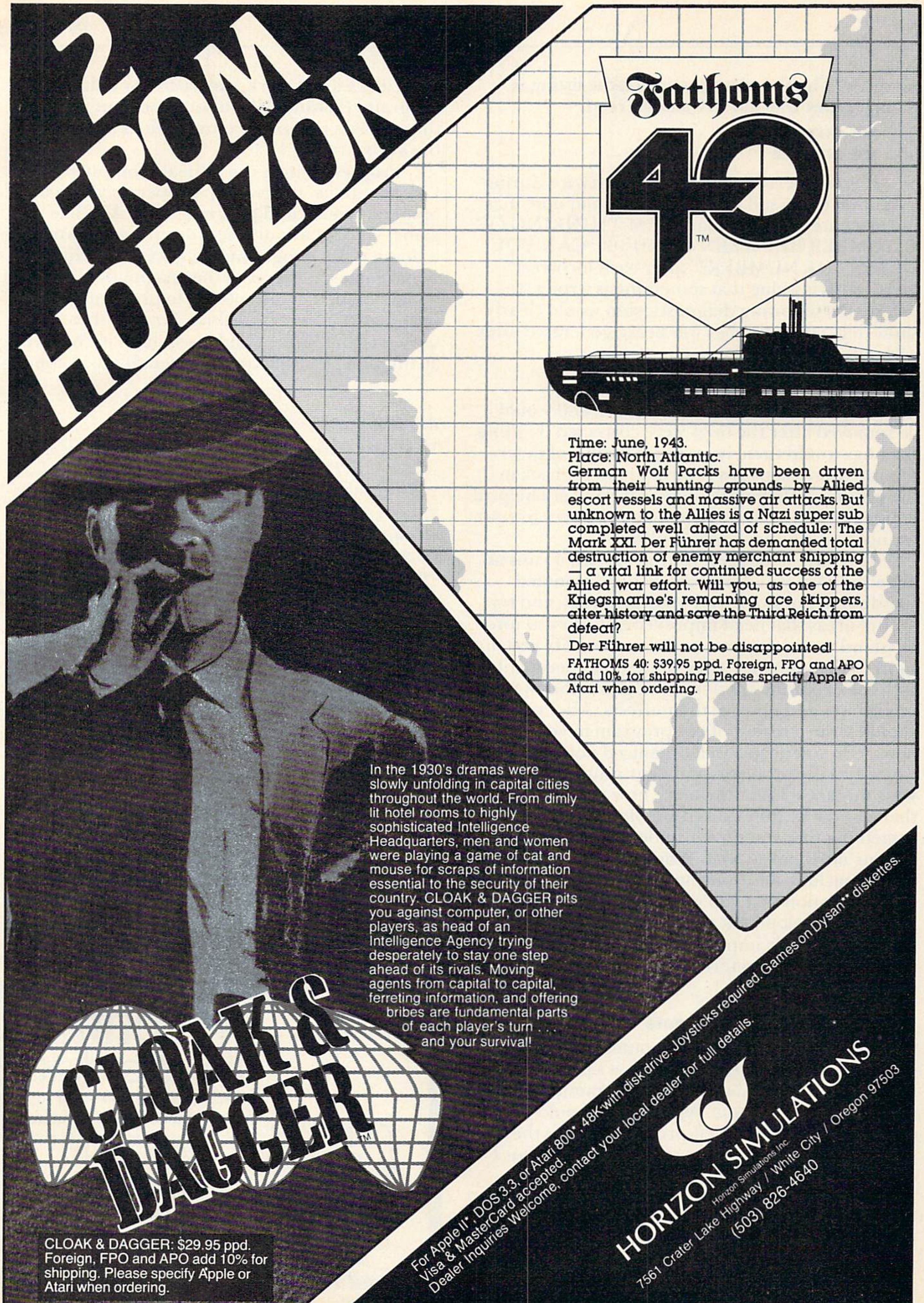
You have completed typing in your program, you are anxious to run it. You type in RUN and press return and then ... If you are lucky you will have a program rich in bugs. Lucky? Solving a tough bug is very educational, and the lessons taught by mistakes are longer lived than when everything goes smooth. Bugs are to programmers what storms are to sailors: you learn from them.

There are two general categories of bugs. The first is a syntax bug. You have violated a rule of the language or the specific machine. The program comes to a screeching halt and an error message is usually displayed. Quite simply, the machine does not understand what you have instructed it to do. Syntax bugs are usually easy to find and correct.

The second is a program logic bug. The program usually does not stop, nor are there any error messages. Rather, the program does unexpected things. The machine is content (that is, the syntax of the program is correct) with your instructions and merrily continues the execution no matter how weird things may get. But your instructions are not telling the computer to do what you want it to do. The problem may only be in your listing, or it may be both in the listing and the flowchart. A review of both items may point to the bug. Program logic bugs can be difficult to find and correct.

Because syntax bugs vary and are usually easy to locate, we will concentrate on how to find program logic bugs. The first thing to do is determine *how* the actual operation differs from the desired operation. Observe the screen carefully and watch for peculiar behavior. Now you have to find the location of the bug. If needed, insert diagnostic stop commands into the program, so that you can check out how far the program proceeds before malfunctioning. Take a look at the variables by inserting print commands. When you have the general area of the bug's location, then look for simple but subtle errors in the listing: failure to RETURN from a subroutine, nested FOR/NEXT statements using the same variable name, confused variable names, GOTO's going to the wrong place, IF/THEN statements blocking steps that must be done regardless. Most program logic errors result from very simple one-step errors. At times, it is hard to believe that a simple error can have such drastic results. Watch out for nested subroutines (one subroutine calling a second subroutine which in turn calls up a third ..., etc). They can multiply a bug immensely. If you still can't find the bug, make sure that the functions employed do what you think they do. Look at calculations. Did you consider the proper order of evaluation (e.g., 6*2+3=15, not 30)? Sometimes you may have to insert or delete steps just to see what happens. You may have to employ test mini-programs. Some bugs may show up only once in a while. You then have to isolate those conditions which cause it. One thing is certain, after you have solved a few tough bugs, you will know more about programming than the person, who by chance, gets things right the first time.

If you enter the program in Example 4 into your machine as it appears, you will find that you seldom will be able to guess the correct number. Your chances will always be 1 in 100. Also, the game will be very confusing: as you key in on what seems to be the correct guess, you will get conflicting results (e.g., 63 TOO LOW, 64 TOO HIGH). Comparison of the listing to the flowchart will identify the bugs. In the flowchart, the program returns to PLEASE ENTER YOUR GUESS after an incorrect guess has been processed. In the listing, the program returns to the beginning of the program and generates a new number. Thus, after each guess, the computer is thinking of a different number, but the user is trying to guess the original number. Thus, the user will very seldom guess


2 FROM HORIZON
Fathoms 40™
Time: June, 1943.
Place: North Atlantic.
German Wolf Packs have been driven from their hunting grounds by Allied escort vessels and massive air attacks. But unknown to the Allies is a Nazi super sub completed well ahead of schedule: The Mark XXI. Der Führer has demanded total destruction of enemy merchant shipping — a vital link for continued success of the Allied war effort. Will you, as one of the Kriegsmarine's remaining ace skippers, alter history and save the Third Reich from defeat?
Der Führer will not be disappointed!
FATHOMS 40: $39.95 ppd. Foreign, FPO and APO add 10% for shipping. Please specify Apple or Atari when ordering.
In the 1930's dramas were slowly unfolding in capital cities throughout the world. From dimly lit hotel rooms to highly sophisticated Intelligence Headquarters, men and women were playing a game of cat and mouse for scraps of information essential to the security of their country. CLOAK & DAGGER pits you against computer, or other players, as head of an Intelligence Agency trying desperately to stay one step ahead of its rivals. Moving agents from capital to capital, ferreting information, and offering bribes are fundamental parts of each player's turn . . . and your survival!
CLOAK & DAGGER™
CLOAK & DAGGER: $29.95 ppd. Foreign, FPO and APO add 10% for shipping. Please specify Apple or Atari when ordering.
For Apple II*, DOS 3.3, or Atari 800*, 48K with disk drive. Joysticks required. Games on Dysan** diskettes.
Visa & MasterCard accepted.
Dealer Inquiries Welcome, contact your local dealer for full details.
HORIZON SIMULATIONS
Horizon Simulations Inc.
7561 Crater Lake Highway / White City / Oregon 97503
(503) 826-4640

correctly. The flowchart is correct, the listing is wrong. Lines 130 and 170 should read:

```
130 GOTO 50
170 GOTO 50
```

If you observed the program behavior during execution (bugs still present), you would note that the machine would display "I AM THINKING OF A NUMBER BETWEEN 1 AND 100. CAN YOU GUESS THE NUMBER?" after each incorrect guess. This is a clue that something is wrong. Inserting the following diagnostic step would clearly show that a new number was being generated after each guess.

```
75 PRINT "A="A, "B="B
```

This step would display the variables, and would be removed after the bugs were corrected. Finding the bugs would then, be a simple matter of determining which steps are returning too far back in the program. Diagnostic steps can be a very helpful debugging aid, so be sure to look for places in your listing in which they can be utilized.

You have debugged your program; it runs as expected. Are you done? From an educational standpoint, no! Look over your program, and try to find areas that need improvement. Do you understand your program, or did you modify certain steps until, by chance, they worked? Experiment with the program: try doing things another way, insert bugs and observe their effects. Also, you should write a review of the program in your notebook. (Do you keep a well organized notebook?) Enter specific problems and your solutions. Note your mistakes. Enter the bugs, how you found them, and why they caused problems. If there is something that you do not understand, enter that in your notebook as well. When you find an explanation, include it in your notebook. Enter all of the documentation that you generated during the program development (i.e., Examples 1 through 4). Enter possible improvements. Your notebook should be the best reference you own; keep it up to date.

The Ultimate Computer Game

Obviously, the High/Low Computer Game is no edge-of-your-seat thriller. In fact, it's pretty boring. But, our intention was to learn programming, not create an exciting game. As such, we should be concerned with how "well-programmed" is the game? As it appears in this article, not very; but I chose brevity and simplicity over efficiency and function. If you have been into computing for a week, you are not going to write Lunar Lander, but you may be able to handle the High/Low Computer Game.

The entire programming procedure may seem like a lot of work, especially for a simple program. However, the most important point in this article is that programming does not start at the keyboard.

Programming is the ultimate computer game. Like any game, it has rules: proper program development, syntax, and program logic. Also like a game, you can win or lose: your program works, or it doesn't. But programming goes beyond conventional games; it exercises the imagination. Without imagination, there is nothing to program. With imagination, there is no limit. For me, there is a sense of achievement and fascination in converting an imagined idea into reality through programming.

REFERENCES

Thornburg, David D. "Computers And Society: Some Speculations On The Well-Played Game," **COMPUTE!**, *Vol. 3, No. 7 (July, 1981), 12-16.*

The idea of the High/Low Computer Game is not original, although the listing in Example 4 is my creation. I ran into this program while attending school, I don't know who came up with the idea originally, but I rather suspect it might be one of Ada's (the Countess of Lovelace) creations. ©

ARTWORX. IT'S A WHOLE NEW WORLD OF SOFTWARE.

Scene from BETA FIGHTER during creation using the DRAWPIC graphics editor.

HODGE PODGE: by Marsha Meredith (Atari and Apple)
NOW AVAILABLE FOR ATARI!!! This captivating program is a marvelous learning device for children from 18 months to 6 years. HODGE PODGE consists of many cartoons, animation and songs which appear when any key on the computer is depressed. A must for any family containing young children.
PRICE.................... **$19.95** diskette

☐ **PM EDITOR:** by Dennis Zander (Atari, 16K)
Create your own fast action graphics game for the Atari 400 or 800 using its player missile graphics features. By using player data stored as strings, players can be moved or changed (for animation) at machine language speed. All this is done with string variables (POS(Y)=SHIP4). This program is designed to permit creation of up to 4 players on the screen, store them as string data and then immediately try them out in the demo game included in the program. Instructions for use in your own game are included. PM EDITOR was used to create the animated characters in **ARTWORX** RINGS OF THE EMPIRE and ENCOUNTER AT QUESTAR IV.
PRICE .... **$29.95** cassette **$33.95** diskette

☐ **ROCKET RAIDERS** by Richard Petersen (Atari 24K)
Defend your asteroid base against pulsar bombs, rockets, lasers, and the dreaded "stealth saucer" as aliens attempt to penetrate your protective force field. Precise target sighting allows you to fire at the enemy using magnetic impulse missiles to help protect your colony and its vital structures.
PRICE .... **$19.95** cassette **$23.95** diskette

☐ **FOREST FIRE!:** by Richard Petersen (Atari,24K)
Using excellent color graphics, your Atari is turned into a fire scanner to help you direct operations to contain a forest fire. You must compensate for changes in wind, weather and terrain. Not protecting valuable property can result in startling penalties. Life-like variables make FOREST FIRE a very suspenseful and challenging simulation. **PRICE** .... **$16.95** cassette **$20.95** diskette

☐ **GIANT SLALOM:** by Dennis Zander (Atari, 16K)
Bring the Winter Olympics to your computer anytime of the year! Use the joystick to guide your skier's path down a giant slalom course consisting of open and closed gates. Choose from three levels of difficulty. Take practice runs or compete against from two to eight additional skiers.
PRICE .... **$15.95** cassette **$19.95** diskette

☐ **THE PREDICTOR** by Thomas Barker
(Apple, Atari, TRS-80, North Star and CP/M (M-BASIC)
This is a complete package that covers least squares fitting of parameters for two or more variables. THE PREDICTOR can be used for predicting sales and process behavior, trend analysis, model building and many other uses calling for multilinear regression techniques. Each option in the program is prompted with simple YES/NO commands making it **very** easy to use.
PRICE .... **$29.95** diskette

☐ **PILOT:** by Michael Piro (Atari, 16K)
Pilot your small airplane to a successful landing using both joysticks to control throttle and attack angle. PILOT produces a true perspective rendition of the runway, which is constantly changing. Select from two levels of pilot proficiency.
PRICE .... **$16.95** cassette **$20.95** diskette

☐ **TEACHER'S PET:** by Arthur Walsh (Atari, Apple, TRS-80, PET, North Star and CP/M (MBASIC) systems).
This is an introduction to computers as well as a learning tool for the young computerist (ages 3-7). The program provides counting practice, letter-word recognition and three levels of math skills.
PRICE .... **$14.95** cassette **$18.95** diskette

☐ **MAIL LIST 3.0:** (Atari, Apple and North Star)
The very popular MAIL LIST 2.2 has now been upgraded. Version 3.0 offers enhanced editing capabilities to complement the many other features which have made this program so popular. MAIL LIST is unique in its ability to store a maximum number of addresses on one diskette (typically between 1200 and 2500 names!). Entries can be retrieved by name, keyword(s) or by zip codes. They can be written to a printer or to another file for complete file management. The program produces 1, 2 or 3-up address labels and will sort by zip code (5 or 9 digits) or alphabetically (by last name). Files are easily merged and MAIL LIST will even find and delete duplicate entries! The address files created with MAIL LIST are completely compatible with **ARTWORX** FORM LETTER SYSTEM.
PRICE .... **$49.95** diskette

☐ **THE VAULTS OF ZURICH:** by Felix and Greg Herlihy (Atari, 24K, PET)
Zurich is the banking capital of the world. The rich and powerful deposit their wealth in its famed impregnable vaults. But you, as a master thief, have dared to undertake the boldest heist of the century. You will journey down a maze of corridors and vaults, eluding the most sophisticated security system in the world. Your goal is to reach the Chairman's Chamber to steal the most treasured possession of all: THE OPEC OIL DEEDS!
PRICE .... **$21.95** cassette **$25.95** diskette

☐ **BRIDGE 2.0** by Arthur Walsh (Atari (24K), Apple TRS-80, PET, North Star and CP/M (MBASIC) systems)
Rated #1 by Creative Computing, BRIDGE 2.0 is the only program that allows you to both bid for the contract and play out the hand (on defense or offense!). Interesting hands may be replayed using the "duplicate" bridge feature. This is certainly an ideal way to finally learn to play bridge or to get into a game when no other (human) players are available.
PRICE .... **$17.95** cassette **$21.95** diskette

☐ **ENCOUNTER AT QUESTAR IV:** by Douglas McFarland (Atari, 24K)
As helmsman of Rikar starship, you must defend Questar Sector IV from the dreaded Zentarians. Using your plasma beam, hyperspace engines and wits to avoid Zentarian mines and death phasers, you struggle to stay alive. This BASIC/Assembly level program has super sound, full player missile graphics and real time action.
PRICE .... **$21.95** cassette **$25.95** diskette

NEW PROGRAMS!

HAZARD RUN: by Dennis Zander (Atari, 16K)
The sheriff has spotted you and you must make the treacherous run through Crooked Canyon past Bryan's Pond to the jump at Hazard Creek and safety. You can even put the joystick-controlled GEE LEE car up on two wheels to make it through some tight spots. A lead foot is not always the answer as you dodge trees, rocks and chickens in this nerve-racking game. HAZARD RUN employs full use of player/missile graphics, re-defined characters and fine scrolling techniques to provide loads of fast action and visual excitement.
PRICE........ **$27.95** cassette **$31.95** diskette

BETA FIGHTER: by Douglas McFarland (Atari, 16K)
See who will be the ace gunner in this action game set on a spectacular Martian landscape. BETA FIGHTER can be played with one or two players and uses player/missile graphics and delightful sound effects.
PRICE........ **$16.95** cassette **$20.95** diskette

DRAWPIC: by Dennis Zander (Atari 16K)
DRAWPIC provides the user with an unbelievably easy way to create screens in graphics modes 3-7. Just sit back with your joystick and use POINT PLOT, DRAW LINE, RUBBER BAND fill and COLOR SET to create beautiful images on your Atari. Full or partial screen images are saved as string data in the program and can be instantly recalled and combined into new images using machine language subroutines. These graphic images can be easily incorporated into your own programs. The images of HODGE PODGE and the landscape of BETA FIGHTER were made using DRAWPIC.
PRICE .... **$29.95** cassette **$33.95** diskette

T: A TEXT DISPLAY DEVICE: by Joseph Wrobel (Atari 16K)
T: is an auto-loading, co-resident assembly language routine which greatly expands the display capabilities of the Atari. It allows you to freely intermix both text and graphics without the use of modified display lists, PEEKS or POKES. This is done by defining a new device ("T:"); printing to that device puts text onto the screen. The size of the text is determined by the graphics mode used.
PRICE........ **$17.95** cassette **$17.95** diskette

ORDERING INFORMATION

Call ARTWORX toll-free number to order direct:
800-828-6573
In New York, Alaska, Hawaii call:
(716) 425-2833

All orders are processed and shipped within 48 hours.

Shipping and handling charges:
Within North America: Add $2.00
Outside North America: Add 10% (Air Mail)
New York State residents add 7% sales tax

Quantity Discounts:
Deduct 10% when ordering 3 or more programs

Ask for ARTWORX at your local computer store.

Write for FREE Catalogue listing more information about these and other quality ARTWORX programs.

Artworx

150 North Main Street Fairport, NY 14450

This technique for the Apple, VIC-20, Atari, and PET is a new approach to an old problem: where to put machine language routines into a BASIC program. Adding small machine language subroutines can greatly speed up a BASIC program, but there are some difficulties when trying to SAVE or LOAD them as one piece. These programs solve this problem.

A New Technique For Mixing BASIC And Machine Language

Louis Sander
Pittsburgh, PA

If you've ever added a machine language subroutine to BASIC, you know how useful this can be in speeding up the program. Several ways of SAVEing the ML and BASIC programs together on one tape have been described, and each has drawbacks. Some require space-consuming DATA statements for the BASIC program. Others do not allow SAVEs from BASIC, or forbid you to ever change the BASIC program. Still others need direct mode POKEs each time the conjoined programs are SAVEd or RUN.

Some store the machine language in cassette buffers or at the top or bottom of memory, where you'd often prefer to keep other programs. None of this is very conducive to writing "fun" programs that you can easily RUN, improve, and modify, or give to friends.

Here is an easy and flexible technique for coupling BASIC programs and short machine language routines. This new technique eliminates all of the above-mentioned drawbacks. After a simple one-time setup, it allows routine SAVEs from BASIC. It does not restrict later changes to your programs, and does not require you to convert your machine language into DATA statements. The new method provides space for 249 bytes of machine language *inside* the BASIC program, and allows you to increase this amount at will. It *does* put two restrictions on the contents of your ML program, but these can be easily "programmed around."

By modifying links and line numbers, the technique establishes a very long dummy line as line 0 of your BASIC program. The line contains 249 dummy bytes which can be used to hold your ML program. You can set up the dummy line before your main program is in memory, or after, provided the main program contains no line numbers lower than 8. Here's how to do it:

1. Enter the following lines, exactly as shown:

Program 1. (PET/CBM)

```
0 REMXXX . . . (45 X's total) . . . XXX
1 REMXXX . . . (45 X's total) . . . XXX
2 REMXXX . . . (45 X's total) . . . XXX
3 REMXXX . . . (45 X's total) . . . XXX
4 REMXXX . . . (45 X's total) . . . XXX
5 IF PEEK (1279) THEN STOP
6 FOR I = 1030 TO 1278 : POKE I,88 : NEXT
7 POKE 1025,0 : POKE 1026,5 : LIST-7
```

The first five lines cannot contain embedded spaces, and each one must contain exactly 45 X's. Since all five are identical except for line numbers, you can type in the first one and use the screen editor to duplicate it.

2. RUN the program. This *must* be done for

Program 2. Apple Version

Since both the PET and the Apple store programs internally in the same way, Program 1 (for the PET) only has to be adjusted to account for memory differences. PET programs start at $0400, Apple programs at $0800, so an offset of 1024 must be added to memory references, and the "line link" bytes in line seven must be changed from $0500 to $0900. Free memory for your machine language program would begin at 2054 and end at 2302. This assumes, of course, Applesoft in ROM. If your machine stores programs in a different area of memory, the program will have to be changed to reflect this. Remember that lines 0-4 must contain exactly 45 X's.

```
0  REM XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
   XXXXXXXXXXXXXX
1  REM XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
   XXXXXXXXXXXXXX
2  REM XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
   XXXXXXXXXXXXXX
3  REM XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
   XXXXXXXXXXXXXX
4  REM XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
   XXXXXXXXXXXXXX
5  IF PEEK(2303) THEN STOP
6  FOR I = 2054 TO 2302 : POKE I, 88 :
   NEXT
7  POKE 2049, 0 : POKE 2050 , 9 : LIST
   - 7
```

VIC and PET are registered trademarks of Commodore Business Machines.

Enter The UMI Dimension of Mind

The Dimension of Mind is an extension of sight and sound. A dimension whose only boundary is imagination. These same characteristics are inherent in the extraordinary line of hardware and software products manufactured by UMI for Commodore's VIC, PET, and other micro-computers. UMI offers micro-computer users unique, innovative entertainment, educational, and professional programs on tape cassette or on UMI's own durable cartridges. UMI also provides memory expansion devices, communications programs, as well as programmer and hobbyist aids.

To begin your journey into The Dimension of Mind, please mail attached coupon to receive UMI's 1982 Catalog or call (714) 594-1351.

Please send me the
UMI 1982 CATALOG
UNITED MICROWARE IND.,INC.
3503 TEMPLE AVE., SUITE C
POMONA, CA 91768

NAME ____________
ADDRESS ____________
CITY ____________ STATE ____________
ZIP ____________ COMPANY ____________

umi

united microware industries, inc. • 3503 temple avenue • suite c • pomona, california • 91768 things to work out properly. If all is well, lines 0-7 will LIST automatically. Notice that line 0 has grown uncommonly large while lines 1 through 4 have disappeared. (If something else happened, you have entered things incorrectly. Reset your computer and start again.)

3. Delete lines 5, 6, and 7.

Your BASIC program now begins with a 255-byte dummy line. The 249 X's occupy memory locations 1030 through 1278, and you can replace them at any time with your machine language program, which you can insert by using POKEs or the monitor. With two exceptions, the BASIC program doesn't care what you put in these locations.

Program 3. Atari Version

The following program will link the first five lines of REM statements into one large REM statement. This will reserve 249 bytes of free memory within a BASIC program.

Each REM statement should have exactly 45 X's. The program will print the starting and ending addresses of the X's in memory. You can POKE in machine language or other data here, and when the program is SAVEd or LOADed, line zero will keep the data POKEd in.

An Atari BASIC program can start at any location in memory, so if you are not sure that the program will always be in the same place in memory, you should make your machine language relocatable.

```
0 REM XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXX
1 REM XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXX
2 REM XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXX
3 REM XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXX
4 REM XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXX
10 ADDR=PEEK(136)+256*PEEK(137)
20 POKE ADDR+2,255:POKE ADDR+3,255
30 FOR I=0 TO 3:FOR J=0 TO 5:POKE ADDR+5
1*I+50+J,88:NEXT J:NEXT I
40 ? "Reserved memory starts at ";ADDR+5
:? "Ends at ";ADDR+253
50 ? "249 BYTES":END


(Line 0 after concatenation:)

0 REM XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXX
```

Program 4. VIC-20 Version

VIC programs are stored from location (hexadecimal) $1000, rather than from $0400 as is the case of the PET/CBM, so an offset of (decimal) 3072 bytes must be added to the POKEs and PEEKs. Also, the "line-link" pointer at line seven must be changed to point at $1100 rather than $0500. Note that if you have the memory expansion module, your programs will be stored at $0400, just like PET/CBM programs, so you should use Program 1. Remember that lines 0-4 must contain exactly 45 X's.

```
0 REM XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 XXXXXXXXXXXX
1 REM XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 XXXXXXXXXXXX
2 REM XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 XXXXXXXXXXXX
3 REM XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 XXXXXXXXXXXX
4 REM XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 XXXXXXXXXXXX
5 IF PEEK(4351) THEN STOP
6 FOR I=4102 TO 4350 : POKE I,88 : NEXT
7 POKE 4097,0 : POKE 4098,17 : LIST -7
```

First, your ML program must not contain any 00's. This restriction exists because BASIC interprets 00's as line delimiters, even when they are part of a REM statement. The second restriction is that your machine language cannot include any CB's (203 decimal); CB is undefined within BASIC, and always gives a ?SYNTAX ERROR. Careful machine language programming can cope with these two restrictions in almost every case.

Once you've put your ML into the dummy line, it will list as gibberish, because PET is trying to interpret it for screen printing, and it contains some unprintable things.

After line 0 has been set up as described above, you can program in BASIC to your heart's content, and your ML won't be affected. When you SAVE or LOAD the BASIC program, the ML will go right along with it as part of line 0. You can add, delete, or change program lines at will, but don't try to edit line 0, because the screen editor will truncate it to 80 characters. Changes to line 0 *must* be made by POKEs or with the monitor.

The method described above will set aside one 249-byte block for machine language programs, but what if you need more? Program 5 will reserve up to 2739 bytes, on the PET/CBM which should be more than enough for any application. It will establish up to nine dummy lines, will tell which and how many memory locations are reserved for ML, and will erase itself when it's finished. It re-

quires that no line in your main program be numbered lower than 23.

These programs are modifying BASIC line links, and taking advantage of the fact that, while the computer's screen editor limits program lines to 80 characters, BASIC lets them be as long as 255.

Program 5: Reserving Up To 2739 Bytes On PET/CBMs

```
0 REM *** ML SPACEMAKER ***
1 REM
2 REM     LOUIS F. SANDER
3 REM     153 MAYER DRIVE
4 REM     PITTSBURGH, PA  15237
5 REM
6 REM *** DELETE ALL BUT LINES 14
   -22              BEFORE US
   ING THE PROGRAM.
7 REM
14 PRINT249*LN"BYTES SET UP FOR ML
   .  WANT MORE  Y{03 LEFT}";
   :INPUTA$:IFA$<>"Y"THEN21
15 PRINT"{CLEAR}{02 DOWN}":FORI=0T
   04:PRINTLN+I"REM";:FORJ=1T
   045:PRINT":";:NEXT:PRINT
16 NEXT:PRINT"LN="LN+1":GOTO17{HOM
   HOME}":POKE158,6:FORI=1TO6
   :POKE622+I,13:NEXT:END
17 A=255:NL=1280+A*(LN-1):IFPEEK(N
   L-1)THENPRINT"ERROR":STOP
18 FORI=NL-250TONL-2:POKEI,88:NEXT
   :POKENL-254,NL/256
19 POKENL-A,NL-256*INT(NL/256):PRI
   NT"{CLEAR}LOCATIONS AVAILA
   BLE FOR M.L.:{02 DOWN}"
20 FORI=1TOLN:PRINT"{UP}LINE"I-1":
   "A*I+775"TO"A*I+1023" = 24
   9 BYTES{DOWN}":NEXT:GOTO14

21 PRINT"{02 DOWN}":FORI=14TO22:PR
   INTI:NEXT
22 FORI=1TO11:PRINT"{UP}";:NEXT:PO
   KE158,9:FORI=1TO9:POKE622+
   I,13:NEXT:END
100 REM
101 REM * FOR ORIGINAL ROMS, SUBSTI
    TUTE           THESE FOR L
    INES 16 & 22:
102 REM
116 NEXT:PRINT"LN="LN+1":GOTO17{HOM
    HOME}":POKE525,6:FORI=1TO6
    :POKE526+I,13:NEXT:END
122 FORI=1TO11:PRINT"{UP}";:NEXT:PO
    KE525,9:FORI=1TO9:POKE526+
    I,13:NEXT:END
```

COLOR CHART

✓ 8 COLORS
✓ ALPHANUMERICS
✓ HIGH RES. GRAPHICS

COLOR VIDEO BOARD FOR PET, AIM, SYM. . . $139.95

COLOR CHART is a new color video generator board that can be used to provide color CRT capability for the COMMODORE PET, ROCKWELL AIM and SYNERTEK SYM. The **COLOR CHART** board plugs into a 4K ROM socket and with two additional connections appears as 4K of RAM. **COLOR CHART** has several modes of operation from a 32 character X 16 line alphanumeric mode in eight colors to a high resolution graphic mode with 128 X 192 controllable pixels in two colors. **COLOR CHART** generates a composite video signal that can be used with a color monitor or a T.V. set and a RF modulator.

NEED HIGH SPEED RS232 SERIAL I/O FOR PET, AIM, SYM?

Portmaker is the answer! Two ACIA's and a socket for a ROM allow Portmaker to plug into a ROM socket and still allow use of the ROM. The serial ports appear as the top eight bytes of the ROM. Full bipolar RS232 buffering is provided. Baud rate can be jumped from 4800 to 300 baud.

Portmaker DUAL SERIAL PORT **$69.95**

NEW **COMPACK $129.95**

INTELLIGENT TERMINAL COMMUNICATIONS PACKAGE

Compack is the Intelligent Terminal package you have been waiting for. A new STANDARD TERMINAL PROGRAM from Eastern House Software combined with a special single PORTMAKER provides the highest performance available for the PET. The package, complete with RS232 cable, is available on Commodore 4040 or PEDISK diskette

EXCHANGE 8" DISKETTES WITH MINI's, MAINFRAMES - IBM 3740

PEDISK II

FLOPPY DISK FOR PET, AIM, SYM

PEDISK is the high performance IBM compatable floppy disk system. Available for PET, AIM, or SYM in both 5" and 8" versions. The 5" system provides an economical single disk that can be expanded. Buy a single disk now, add more drives later. The 8" system provides full IBM 3740 compatibility. A new program, EXCHANGE, even allows PEDISK to read and write ASCII files to IBM compatible computers.

SINGLE 5" DISK SYSTEM (540-1) $595.00 SINGLE 8" DISK SYSTEM (877-1) $995.00

PEDISK II SOFTWARE

PAPERMATE - WORD PROCESSOR	$ 50.00
MAE - MACROASSEMBLER/EDITOR	$169.95
FLEXIFILE II - DATA BASE MANAGER	$ 60.00
fullFORTH+ - complete FORTH system	$ 75.00
IBM EXCHANGE - read/write ACSII files	$125.00

Spacemaker II $39.95
Switch 1 of 4 ROM's on a single board. Slide switch electronically controlls selection, no glitches. Remote Control switch capability.

Romdriver............... $39.95
Remote Control switch output board.

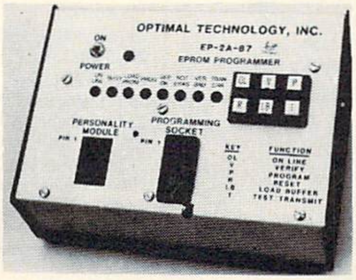
ROM/IO................. $ 9.95
Software for ROMDRIVER.

cgrs **MICROTECH**
P.O. Box 102 • LANGHORNE, PA 19047
215-757-0284

Model EP-2A-87

EPROM Programmer

The Model EP-2A-87 EPROM Programmer has an RS-232 compatible interface and includes a 2K, 4K or 8K buffer. Seventeen RS-232 commands allow another computer to download or remotely control the Programmer. INTEL, TEXTRONIX OR MOTOROLA formats are supported. The buffer may be edited directly from a CRT and EPROMS can be copied off-line. Power requirements are 115v 50/60 Hertz at 15 watts.



EP-2A-87-1	Programmer with 2K Buffer	$575.00
EP-2A-87-2	Programmer with 4K Buffer	$650.00
EP-2A-87-3	Programmer with 8K Buffer	$725.00
	Non-Standard voltage (220v, 240v, or 100)	$ 15.00

Personality Modules

PM-0	TMS 2708	$18.00	PM-5E	2816	$36.00
PM-1	2708	18.00	PM-6	2704	18.00
PM-2	2732	34.00	PM-7	2758	18.00
PM-2A	2732A	34.00	PM-8	MCM68764	36.00
PM-3	TMS 2716	26.00	PM9	2764	36.00
PM-4	2532	34.00	PM-10	2564	36.00
PM-5	2716	18.00			
SA-64-2	2564	39.00	SA-64-3	2764	39.00

Optimal Technology, Inc.
Phone (804) 973-5482
Blue Wood 127 Earlysville, VA 22936

Single Board Computers	Bare Board	Kit	Assm.
80-153 1k RAM 2k EPROM	$19.95	$ 89.95	$110.95
80-280 1k RAM 2k EPROM	$19.95	$119.95	$129.95
81-260 2k RAM 4k EPROM	$39.95	$149.95	$199.95
81-030 4k RAM 16k EPROM	$89.95	*$249.95	$349.95
8085 3 chip system	$24.95		
8088 5 chip system	$29.95		
Apple II boards	**Bare Board**	**Kit**	**Assm.**
79-295 Parallel I/O	$19.95	$59.95	$ 69.95
81-132 A-D 16 Channel	$29.95	$69.95	$ 89.95
80-085 12k EPROM (2716)	$39.95	$49.95	$ 59.95
80-244 EPROM PROGRAMMER	$24.95	$39.95	$ 49.95
80-144 Display add + data	$25.95	$42.95	$ 49.95
80-297 Parallel Printer Interface		$69.95	$ 79.95
Extender Board		$12.95	
81-088 Speech Synthesizer (Votrax SC-01)			$139.90
Other Products	**Bare Board**	**Kit**	**Assm.**
CRT Controller	$39.95		
79-287 A-D and D-A	$19.95	$59.95	$ 79.95
79-282 S S Switch 1 ch.	$ 6.95	$10.95	$ 13.95
79-282 S S Switch 4 ch.	$24.95	$39.95	$ 49.95
80-160 + 5 Volt 500 Ma.	$ 8.95	$16.95	$ 20.95
80-161 ± 12 Volt 120 Ma.	$ 8.95	$18.95	$ 22.95
81-120 Parallel Input Speech Synthesizer			$149.90

Send for complete Catalog and OEM pricing.
*Partially Populated price.

JOHN BELL ENGINEERING, INC.

ALL PRODUCTS ARE AVAILABLE FROM **JOHN BELL ENGINEERING, INC.**
MC 1014 CENTER ST., SAN CARLOS, CA 94070 VISA
ADD SALES TAX IN CALIFORNIA
ADD 5% SHIPPING & HANDLING • 3% FOR ORDERS OVER $100
10% OUTSIDE U.S.A.
SEND FOR CATALOG **(415) 592-8411**
WILL CALL HOURS: 9am - 4pm

KMMM Pascal for PET/CBM $85

A subset of standard Pascal with extensions.
- Machine language Pascal Source Editor with cursor oriented window mode
- Machine Language P-Code Compiler
- P-Code to machine language translator for optimized object code
- Run-time package
- Floating point capability
- User manual and sample programs

Requires 32K Please specify configuration.

EARL for PET (disk file based) $65

Editor, Assembler, Relocater, Linker

Generates relocatable object code using MOS Technology mnemonics. Disk file input (can edit files larger than memory). Links ,multiple object programs as one memory load. Listing output to screen or printer. Enhanced editor operates in both command mode and cursor oriented "window" mode.

RAM/ROM for PET/CBM

4K or 8K bytes of soft ROM with optional battery backup.

RAM/ROM is compatible with any large keyboard machine. Plugs into one of the ROM sockets above screen memory to give you switch selected write protectable RAM.

Use RAM/ROM as a software development tool to store data or machine code beyond the normal BASIC range. Use RAM/ROM TO LOAD A ROM image where you have possible conflicts with more than one ROM requiring the same socket. Possible applications include machine language sort (such as SUPERSORT), universal wedge, Extramon, etc.

RAM/ROM - - 4K	$85
RAM/ROM - - 8K	120
Battery Backup Option	30

SUBSORT by James Strasma $35

Subsort is an excellent general purpose machine language sort routine for PET/CBM computers. Sorts both one and two dimensioned arrays at lightning speed in either ascending or descending order. Other fields can be subsorted when a match is found, and fields need not be in any special order. Sort arrays may be specified by name, and fields are random length. Allows sorting by bit to provide 8 categories per byte. The routine works with all PET BASICs, adjusts to any memory size, and can co-exist with other programs in high memory.

SuperGraphics 2.0

NEW Version with TURTLE GRAPHICS

SuperGraphics, by John Fluharty, provides a 4k machine language extension which adds 35 full featured commands to Commodore BASIC to allow fast and easy plotting and manipulation of graphics on the PET/CBM video display, as well as SOUND Commands. Animations which previously were too slow or impossible without machine language subroutines now can be programmed directly in BASIC. Move blocks (or rocketships, etc.), or entire areas of the screen with a single, easy to use BASIC command. Scroll any portion of the screen up, down, left, or right. Turn on or off any of the 4000 (8000 on 8032) screen pixels with a single BASIC command. In high resolution mode, draw vertical, horizontal, and diagonal lines. Draw a box, fill a box, and move it around on the screen with easy to use BASIC commands. Plot curves using either rectangular or polar co-ordinates (great for Algebra, Geometry and Trig classes.)

The SOUND commands allow you to initiate a note or series of notes (or even several songs) from BASIC, and then play them in the background mode without interfering with your BASIC program. This allows your program to run at full speed with simultaneous graphics and music.

Seven new TURTLE commands open up a whole new dimension in graphics. Place the TURTLE anywhere on the screen, set his DIRECTION, turn him LEFT or RIGHT, move him FORWARD, raise or lower his plotting pen, even flip the pen over to erase. Turtle commands use angles measured in degrees, not radians, so even elementary school children can create fantastic graphic displays.

Specify machine model (and size), ROM type (BASIC 3 or 4)

SuperGraphics (disk or tape)	$ 40
SuperGraphics in ROM	55

Volume discounts available on ROM version for schools.

FLEX-FILE

for PET/CBM Computers

Self Calculating
DATA BASE
REPORT WRITER
MAILING LIST

FLEX-FILE is a set of flexible, friendly programs to allow you to set up and maintain a data base. Print files with a versatile Report Writer or a Mail Label routine. Programmers will find it easy to add subroutines to their own programs to make use of Data Base files.

RANDOM ACCESS DATA BASE

Record size limit is 250 characters. The number of records per disk is limited only by the size of each record and the number of records per disk is limited only by the size of each record and the amount of free space on the disk. File maintenance lets you step forward or backward through a file, add, delete, or change a record, go to a numbered record, or find a record by specified field. The Find command locates any record when you enter all (or a portion of) the desired key. Field lengths may vary from record to record to allow maximum packing of information. Files may be sorted by any field, and any field may be specified as a key. Sequential files from other programs may be converted to Flex-File format, and Flex-File records may be converted to sequential (WordPro, PaperMate, other word processors may also use Flex-File data). Maximum record size, fields per record, and order of fields may be changed at any time.

MAILING LABELS

With typical record size of 127 characters, each disk can handle over 1000 records (about 2800 with 8050 drive). Labels may be printed any number wide, and may begin in any column position. There is no limit on the number or order of fields on a label, and two or three fields may be joined together on one line (like first name, last name, and title). A "type of customer" field allows selective printing.

REPORT WRITER

Print any field in any column. For numeric fields, use decimal point justification (and round to any accuracy). Define any column as a series of mathematical functions performed on other columns. These functions include arithmetic operations and various log and trig functions. Pass results of operations such as running total from row to row. At the end of the report, print total and/or average for any column. Complete record selection, including field within range, pattern match, and logical functions can be specified individually or in combination with other parameters.

FLEX-FILE by Michael Riley $60

Please specify equipment configuration when ordering.

Low Cost Disk Drive for PET/CBM

PEDISK II from cgrs Microtech is a new disk system ready to plug into your large keyboard PET/CBM.

PEDISK II offers speed, reliability, IBM compatibility.

Complete system prices with DOS and cable:

5" 40 track, 1 drive, 143K	$525
5" 40 track, 1 drive, 286K	690
8" IBM 3740 format, 77 track, 250K	995

PROGRAM YOUR OWN EPROMS

Branding Iron for PET/CBM $79

EPROM Programmer with software for all ROM versions. Includes all hardware and software to program or copy 2716 and 2532 EPROMs.

CBM Software

Legal Time Accounting Package	445
Medical Accounting Package	

Complete CBM Business Software Package

Can be tailored to meet most business requirements.

Technician's Investment Analysis Package	500
Dow Jones Portfolio Management	129
Personal Tax Calculator	65
Tax Preparation System	445
Wordcraft 80 Wordprocessor Package	325
Pascal Development Package	235
Assembler Development Package	99
Intelligent Terminal Emulator	30
Softpac-1 (Competitive Software)	29

FORTH for PET

BY L. C. Cargile and Michael Riley **$50**

Features include:
- full FIG FORTH model.
- all FORTH 79 STANDARD extensions.
- structured 6502 Assembler with nested decision making macros.
- full screen editing (same as when programming in BASIC).
- auto repeat key.
- sample programs.
- standard size screens (16 lines by 64 characters).
- 150 screens per diskette on 4040, 480 screens on 8050.
- ability to read and write BASIC sequential files.
- introductory manual.
- reference manual.

Runs on any 16K or 32K PET/CBM (including 8032) with ROM 3 or 4, and CBM disk drive. Please specify configuration when ordering.

Available soon:

Metacompiler for FORTH $30

simple metacompiler for creating compacted object code which can be executed independently (without the FORTH system).

PaperMate 60 COMMAND WORD PROCESSOR

by Michael Riley

Paper-Mate is a full-featured word processor for CBM/PET. Paper-Mate incorporates 60 commands to give you full screen editing with graphics for all 16K or 32K machines (including 8032), all printers, and disk or tape drives.

For writing text, Paper-Mate has a definable keyboard so you can use either Business or Graphics machines. Shift lock on letters only, or use keyboard shift lock. All keys repeat.

Paper-Mate text editing includes floating cursor, scroll up or down, page forward or back, and repeating insert and delete keys. Text block handling includes transfer, delete, append, save, load, and insert.

All formatting commands are imbedded in text for complete control. Commands include margin control and release, column adjust, 9 tab settings, variable line spacing, justify text, center text, and auto print form letter (variable block). Files can be linked so that one command prints an entire manuscript. Auto page, page headers, page numbers, pause at end of page, and hyphenation pauses are included.

Unlike most word processors, CBM graphics as well as text can be used. Paper-Mate can send any ASCII code over any secondary address to any printer.

Paper-Mate functions with 16/32K CBM/PET machines, with any printer, and with either cassette or disk.

To order Paper-Mate, please specify configuration.

Paper-Mate on disk or tape 40.00

BASIC INTERPRETER $200

Designed to support the CBM 8096 (8032 with add-on 64K board). A full interpreter implementation to automatically take advantage of the extra memory available to the 8032.

BPI General Ledger – 8032/8050	300
BPI Accounts Receivable – 8032/8050	300

Hayden Software

Complex Mathematics	12.70
Engineering Mathematics	12.70
General Mathematics	12.70
MCAP:Circuit Analysis Program	21.00
Energy Miser	24.50

CASH MANAGEMENT SYSTEM $45

Easy to use. Keeps track of cash disbursements, cash receipts, cash transfers, expenses for up to 50 categories.

MICRO-REVERSI for PET by Michael Riley **10**

super machine language version of Othello

Tunnel Vision / Kat & Mouse by Michael Riley **10**

two excellent machine language maze programs

252 Bethlehem Pike
Colmar, PA 18915

215-822-7727

A B Computers

WRITE FOR CATALOG

Add $1.25 per order for shipping. We pay balance of UPS surface charges on all prepaid orders. Prices listed are on cash discount basis. Regular prices slightly higher. Prices subject to change.

This is the concluding segment of a three-part review of Microsoft BASIC for Atari. Included is a demonstration program, illustrating some of the features of "AMSB."

Review:

Atari Microsoft BASIC (Part III)

Jerry White
Levittown, NY

This is the third and final part of our series on Atari Microsoft BASIC. A demonstration program called ALARM.MSB has been provided as an example. I used the file extension .MSB to indicate Microsoft BASIC. Since I have three different versions of BASIC, I had to have some way of knowing which version was used in a given program. If you have more than one version of BASIC, I suggest you use filename extensions .MSB to indicate Microsoft, and .BAP to indicate BASIC A+. Programs in Atari 8K BASIC may then be identified by having no extension, or by using .A8K, or .BAS.

Read the introduction display messages found in program lines 310 through 350. The user is prompted to enter the current hour, minute, and second, followed by an alarm hour and minute. For example, enter the current time as hour 9, minute 58, and second 30. Then enter the alarm hour as 10 and the alarm minute as 0. This will set the alarm for 1 minute and 30 seconds from our current time.

You may wonder why someone would want to use a computer as an alarm clock. In this case, we want to demonstrate commands unique to Atari Microsoft BASIC (AMSB). Since our computer has much greater display capability than any alarm clock, we can also tell someone why the alarm was set. The next and final prompt says, "ENTER ALARM MESSAGE BELOW:". This message will be displayed when the current time and the alarm time are equal.

As you begin to type this program into the computer, you will soon notice some strange looking statements. The SETCOLOR 6,9,0 is not an error. In GRAPHICS 0 of AMSB, this sets the background color to dark blue-grey. What ever happened to SETCOLOR 0, you ask? Color registers zero through three are used to set the colors of Player Missiles.

At program line 190, notice the command LINE INPUT MESSAGE$. LINE INPUT takes one screen line of keyboard input and places it into a string.

At program line 200 notice the command PRINT #6 AT (8,2). AT (8,2) replaces the Atari 8K BASIC POSITION statement.

You might also think there is an error in program line 240. Notice that the SOUND command contains five variables. That fifth variable is optional and specifies duration in 60ths of a second or jiffies. Each time the current minute changes, this sound will occur for 15/60ths or one quarter of a second. Up to twenty pending SOUND commands may be saved in what is called the *stack*. This permits your BASIC program to go on about other tasks while SOUND commands are executed automatically.

If you were to set an alarm for more than a few minutes into the future, you could shut off your TV or monitor and save a few watts. The alarm sound is the console bell. When the bell rings twice, you could turn your TV back on to read the message.

This concludes our series on Atari Microsoft BASIC. It was my intention to point out its strengths and weaknesses, make comparisons to Atari 8K BASIC and O.S.S. BASIC A+, and to demonstrate some of its unique commands. AMSB is not for everyone. I hope I've provided enough information so that you can decide if Atari Microsoft BASIC is for you.

```
10 REM ALARM.MSB by Jerry White
20 REM ATARI Microsoft BASIC
30 REM Alarm Clock Demonstration
40 GOSUB 300:GRAPHICS 0:SETCOLOR 6,9,0:P
OKE 82,10
50 PRINT:PRINT"* ATARI ALARM CLOCK *":PR
INT
60 PRINT:INPUT "ENTER CURRENT HOUR:";HR
70 HR=INT(HR):IF HR<0 OR HR>23 THEN 60
80 T=HR*60*60*60
90 PRINT:INPUT "ENTER CURRENT MINUTE:";M
IN
0100 MIN=INT(MIN):IF MIN<0 OR MIN>59 THE
N 90
0110 T=T+MIN*60*60
0120 PRINT :INPUT "ENTER CURRENT SECOND:
";SEC
0130 SEC=INT(SEC):IF SEC<0 OR SEC>59 THE
N 120
0140 T=T+SEC*60:TIME=T:REM TIME$ NOW HOL
DS THE CURRENT TIME
```

Santa Cruz Educational Software * Tricky Tutorials * Santa Cruz Educational Software * Tricky Tutorials * Santa Cruz Educational Software * Tricky Tutorials

ATARI (tm) GRAPHICS AND SOUND MADE EASY!

SANTA CRUZ EDUCATIONAL SOFTWARE HAS WRITTEN A SERIES OF AFFORDABLE PROGRAMS THAT DEMONSTRATES MANY OF THE SPECIAL "TRICKS" THAT ONLY THE ATARI 400/800 COMPUTER IS CAPABLE OF DOING. WE OFFER EVERYTHING FROM A PROGRAM THAT DOES FANTASTIC HIGH RESOLUTION GRAPHICS TO ONE THAT ACTUALLY DIALS YOUR PHONE.....

★★★

MASTER MEMORY MAP(tm) — This is really the key to using the ATARI'S capabilities. We start out by explaining how to PEEK and POKE values into memory so that even new programmers can use this. Then: we give you over 15 pages of the memory locations that are the most useful. The information is condensed from both the ATARI'S Operating System Manual and various articles and programs. It is, of course, useful even for experienced programmers as a reference. Also, we highly suggest that dealers offer this Memory Map to customers who request to be told how to use the power of the machine. We guarantee it will answer many of the questions you have about the machine.

$6.95

TRICKY TUTORIALS(tm)

#1: DISPLAY LISTS — This program teaches you how to alter the program in the ATARI that controls the format of the screen. For example: when you say graphics 8 the machine responds with a large graphics 8 area at the top of the screen and a small text area at the bottom. Now, you will be able to mix the various modes on the screen at the same time. Just think how nice your programs could look with a mix of large and small text, and both high and low resolution graphics. this program has many examples plus does all of the difficult caluculations!

#2: HORIZONTAL/VERITCAL SCROLLING — The information you put on the screen, either graphics or text, can be moved up, down or sideways. This can make some nice effects. You could move only the text on the bottom half of the screen or perhaps create a map and then move smoothly over it by using the joystick.

#3: PAGE FLIPPING — Normally you have to redraw the screen every time you change the picture or text. Now you can learn how to have the computer draw the next page you want to see while you are still looking at the previous page, then flip to it instantly. You won't see it being drawn, so a complicated picture can seem to just appear. Depending on your memory size and how complicated the picture, you could flip between many pages, thus allowing animation or other special effects with your text.

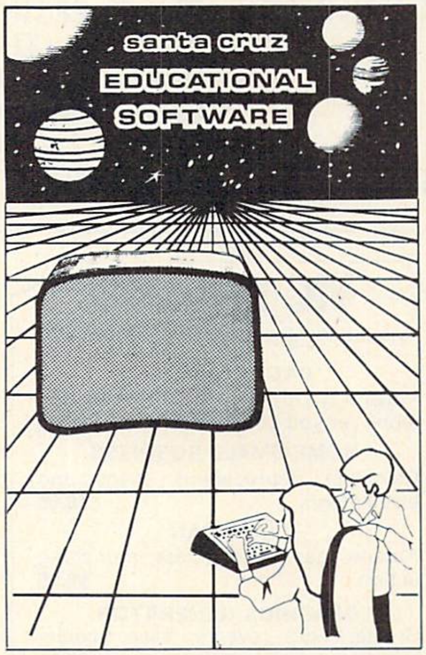
#4: BASICS OF ANIMATION — Shows you how to animate simple shapes using the PRINT and PLOT commands, and also has nice little PLAYER/MISSILE Graphics demo to learn. This would be an excellent way to start making your programs come alive on the screen. Recommended for new owners.

#5 PLAYER MISSILE GRAPHICS — This complex subject is demonstrated by starting with simple examples, and building up to a complete game and also an animated business chart on multiple pages! As always, the computer does most of the calculations. Requires 32K disk or tape and costs **$29.95**

#6: SOUND — From explaining how to create single notes, to demonstrating complex four channel sound effects, this newest tutorial is great. Even those experienced with ATARI's sound capabilities will find the menu of sound effects a needed reference that can be used whenever you are in the need of a special sound for your programs. Everyone will learn something new! Written by Jerry White.

Tricky Tutorials (except #5) require 16K memory for cassette orders and 24K for disk. The price is $19.95 each. You may order 1,2,3, & 4 for $64.95. All six in a colorful binder cost $99.95.

THE GRAPHICS MACHINE!! — Turn your computer into an incredible graphics tool with advanced commands like circle, box, fill, polygon, line, help, etc. 3 colors in graphics 8 with instant text!!! Create colorful business charts or beautiful drawings and then save or retrieve them from disk in 5 SECONDS. YES, it's that fast. Needs all 48K, disk, and costs **$19.95**



MINI-WORD PROCESSOR — This is for those of you who have a printer, but don't want to spend $100 or more for a fancy word processor. It is suitable for simple editing of text, accepts most control characters for your printer, and text is stored on disk for easy retrieval. Holds 2½ typed pages at a time. Requires 32K, disk or tape. **$19.95**

BOB'S BUSINESS — 14 small business type programs for home or office, all chosen from a nice menu. Supports printed output. 169 sectors of output require 16K tape, or 32K disk. **$14.95**

KID'S #1 — Includes the following: 1) TREASURE — search for the lost treasure while trying to keep from falling into the sea. Nice graphics if you find it!; 2)DIALOGUE — talk back to the computer about four subjects; 3) MATH QUIZ — Nice musical and graphical rewards for good scores. Parents input the level of difficulty.

KID'S #2 — A spelling quiz, a "scrabble" type game, and a version of Touch with the computer giving all the directions! Both Kid's programs require 16K tape or 24K disk and cost **$14.95** each.

MINI-DATABASE/DIALER — This unique new program stores and edits up to 8 lines of information such as name, address, and phone numbers, or messages, inventories or anything you want. It has the usual sort, search, and print options, but it also has an unusual feature: If your files include phone numbers and you have a touch-tone phone, the program will DIAL THE PHONE NUMBERS FOR YOU! This is perfect for those who make a lot of calls like salesmen, teens, or those trying to get through to busy numbers (acts as an auto-redialer). It is also a lot of fun to use. Requires 16K cassette or 24K disk and costs **$24.95**

FONETONE — For those who only want to store name and phone numbers and have the dialer feature as above, we offer this reduced version. Same memory requirements, but only costs **$14.95**. Don't forget you must have a touch-tone phone.

PLAYER PIANO — Turns your keyboard into a mini-piano and more. Multiple menu options provide the ability to create your own songs, save or load data files using cassette or diskette, fix or change any of up to 400 notes in memory, and play all or part of a song. The screen displays the keyboard and indicates each key as it is played from a data file or the notes you type. You don't have to be a musician to enjoy this educational and entertaining program. Requires 24K cassette or 32K disk. **$14.95**

BOWLERS DATABASE — Provides the league bowler with the ability to record and retrieve bowling scores providing permanent records. The data may then be analyzed by the program and displayed or printed in summary or detail form. Data may be stored on cassette or diskette and updated quickly and efficiently. The program proivdes such information as highest and lowest scores by individual game, (first, second, and third games throughout the season), high and low series, current average, and more. The program listing and documentation provided are a tutorial on ATARI basic and record keeping. Requires 16K for cassette or 24K for disk. **$14.95**

By the time you read this all computers (400/800) being produced should have the fabled GTIA chips included. ATARI service may upgrade older computers...call and ask (it's easy to do yourself). We have one and the improvements that graphics modes 9,10, and 11 offer are great!! To help you figure out what to do with the new modes a new Tricky Tutorial will be offered in March on Modes 9 to 11. Either give us a call or write around that-time.

★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★

SEE YOUR LOCAL DEALER
OR
ORDER FROM:

SANTA CRUZ EDUCATIONAL SOFTWARE
5425 Jigger Dr., Soquel, CA 95073
C.O.D. Orders call (408) 476-4901
***Includes $3.00 postage ($1.00 for Memory Map)**
***In California, include 6.5% tax**
VISA and MasterCard Accepted.

Santa Cruz Educational Software * Tricky Tutorials * Santa Cruz Educational Software * Tricky Tutorials * Santa Cruz Educational Software * Tricky *

```
0150 PRINT:INPUT "ENTER ALARM HOUR:";AH
0160 AH=INT(AH):IF AH<0 OR AH>23 THEN 15
0
0170 PRINT:INPUT "ENTER ALARM MINUTE:";A
M
0180 AM=INT(AM):IF AM<0 OR AM>59 THEN 17
0
0190 PRINT:PRINT "ENTER ALARM MESSAGE BE
LOW:":LINE INPUT MESSAGE$:POKE 82,2:PRIN
T
0200 GRAPHICS 18:PRINT #6, AT (8,2);"tim
e":AH$=""
0210 NEWMIN$=MID$(TIME$,4,2):MIN=VAL(NEW
MIN$):HR=VAL(LEFT$(TIME$,2))
0230 IF NEWMIN$=OLDMIN$ THEN 250
0240 P=VAL(MID$(TIME$,4,2)):SOUND 0,P+10
,10,15,15
0250 OLDMIN$=NEWMIN$:IF OLDTIME$<>TIME$
THEN OLDTIME$=TIME$:SETCOLOR 4,RND(0)*16
,10:SETCOLOR 5,RND(0)*16,10
0260 PRINT#6, AT (6,4);TIME$
0270 IF HR=AH AND MIN=AM THEN 290
0280 G.210
0290 POKE 766,0:GRAPHICS 0:PRINT CHR$(25
3):PRINT:PRINT MESSAGE$:PRINT CHR$(253):
END
0300 GRAPHICS 0:SETCOLOR 6,1,0:POKE 82,2
:POKE 83,39:POKE 752,1:PRINT
0310 PRINT :PRINT "  This Atari Microso
ft Basic program":PRINT "demonstrates th
e use of some unique"
0320 PRINT "commands such as TIME, TIME$
, and the":PRINT "SOUND command's fifth
variable (dura-"
0330 PRINT "tion.) The SOUND will occur
 on each":PRINT "change of minute.":PRIN
T
0340 PRINT "  The program may be used t
o set an":PRINT "alarm and have the Atar
i ring it's"
0350 PRINT "bell and display a reminder
message.":PRINT :PRINT TAB(9)"PRESS ISTA
RTI TO BEGIN"
0360 FOR VOL=15 TO 0 STEP-0.5:SOUND 0,0,
2,VOL:NEXT VOL
0370 IF PEEK(53279)<>6 THEN POKE 755,3:P
OKE755,2:G.370
0380 RETURN
```

©

To use your VIC to call other computers (or to use such telecommunications services as The Source or Micronet), you need special connecting equipment (an interface) to add a modem *to the VIC. A modem is the device that lets your computer "talk" and "listen" to other computers, on the telephone.*

Review:

Modem Driver Module (MDM-1)

Harvey B. Herman
Associate Editor

The MDM-1 module is an interface which allows the VIC to be connected to a modem. A simple program can then turn the VIC into a terminal which can access remote computers. In the process of evaluating it, I learned a few things about the VIC which were surprising (at least to me) and which I would like to share.

The full name for this equipment is "modem driver module." It plugs into the VIC user port and has connectors for a modem and a printer. You supply the cables. My experience with the PET, which also has a user port, led me to believe that somewhat complicated software would be necessary to drive the modem. Several machine language programs to do this are commercially available for the PET. Much to my surprise, I found the VIC supports the user port as a device for serial communications. That means that a relatively short BASIC program is all that is required to send data to a printer or to a remote computer using the VIC as a terminal. The designers of the VIC should be commended for including this feature in a low cost machine. [*"Terminal" means using the computer to communicate with other computers. In other words, it is not just working within itself as a self-contained device.*]

Connecting To The Modem

It is easy to set up the Modem Driver Module. Built in a sturdy box, it plugs in easily to the user port in the rear of the VIC. (Power off please!) Any standard RS-232 cable (male at each end) can be used to attach a modem. The modem must be capable, however, of accepting TTL logic level signals as input. Ask a knowledgeable person if you are not sure. I had no trouble working with a Novation CAT modem or a ComData modem, at 300 baud (the number of characters sent per second). My guess is that most people will not have trouble with newer modems.

Hints On Software

When the hardware is in place, the VIC can be safely turned on and a terminal program loaded. A BASIC program is normally considered too slow for this application (i.e., using the VIC as a terminal). Two things mitigate against this. One, an area of memory is set aside as a buffer to store incoming characters temporarily if the computer is busy doing other things. Two, the remote computer can be directed to wait a short time after each line is sent, giving the VIC a chance to catch up.

The logic of a BASIC communications program should go something like this:

1. Open a file to the modem.
2. Get a character from the keyboard.
3. If a key is pressed, send that character to the modem.
4. Get a character from the modem.
5. If a character is received, print it on screen.
6. Repeat steps two and five.

Step one automatically allocates the buffer and allows the user to specify the characteristics of the serial transmission. The programmer has complete control of the baud rate, bits per character, and parity – much as he would if this were a "real" terminal with switch-selected options.

I wrote a short program, shown in the listing, to test the hardware for this review. It would take a much longer article to explain it thoroughly, so don't despair if it is obscure. It will make more sense when the Commodore VIC User's Guide becomes generally available. However, the program assumes a typical communications configuration (300 baud, 7 bit word, even parity, and full duplex) so that many people can use it with only minor changes.

The hardware and software worked fine when communicating with a DEC VAX-11/780. The combination of a VIC, modem, and Modem Driver Module makes an inexpensive terminal whose only limitation is the short, 20 character, line length. The VIC had no trouble keeping up with the computer, under worst case conditions, as long as I added fill characters after carriage return. This does slow the speed somewhat, and some users might prefer a faster machine language program. However, I was content staying with BASIC.

There are other ways to use this module which shouldn't be slighted. A printer and a modem can be connected simultaneously to produce a "printing terminal." You can even use it without a modem to get VIC BASIC listings on a printer using a so-called *null modem cable.* The printer must have a proper buffer, however, for this application to work. [*A buffer is a temporary storage area for data.*] I did verify that I was able to make short listings on a Base 2 printer as long as the printer buffer was enabled, but I did not test it as a printing terminal.

This hardware is well constructed and worth the price. The only caveat is that you make sure your modem will accept TTL as input. [*TTL is a particular kind of electronic circuit; it means Transistor-Transistor-Logic.*]

MDM-1
RVR Systems
P.O. Box 265
Dewitt, New York 13214
$59 plus $3 shipping

```
100 REM VIC 300 BAUD TERMINAL
110 REM
120 REM HARVEY B. HERMAN
130 REM
140 OPEN 2,2,3,CHR$(6+32)+CHR$(32+6
    4):REM 300 BAUD, 1 STOP BI
    T, AND EVEN PARITY.
150 FL=0:REM CLEAR CONTROL CHARACTE
    R FLAG
160 PRINT CHR$(14);"{CLEAR}";:REM L
    OWER CASE CHARACTER SET AN
    D CLEAR SCREEN
170 GET B$:REM INPUT FROM KEYBOARD
180 IF B$="" THEN 250
190 IF B$="\" THEN FL=1:GOTO 250:RE
    M SET CONTROL CHARACTER FL
    AG
200 IF ASC(B$)=136 THEN B$=CHR$(127
    ):REM F7 IS DEL
210 IF ASC(B$)=133 THEN B$=CHR$(27)
    :REM F1 IS ESC
220 IF FL=1 THEN B=ASC(B$)-64:B$=CH
    R$(B):FL=0:REM KEY PRESS I
    NTO CONTROL CHARACTER
230 B=ASC(B$):GOSUB 360:B$=CHR$(B)
240 PRINT#2,B$;:REM OUTPUT TO MODEM

250 GET#2,C$:REM INPUT FROM MODEM
260 IF C$="" THEN 300
270 C=ASC(C$) AND 127:REM MASK OUT
    PARITY BIT
280 GOSUB 320
290 PRINT CHR$(C);:REM OUTPUT TO SC
    REEN
300 IF ST<>0 THEN CLOSE 2:STOP
310 GOTO 170
320 REM ASCII TO VIC
330 IF C>64 AND C<91 THEN C=C+128:R
    ETURN
340 IF C>96 AND C<123 THEN C=C-32
350 RETURN
360 REM VIC TO ASCII
370 IF B>64 AND B<91 THEN B=B+32:RE
    TURN
380 IF B>192 AND B<219 THEN B=B-128

390 RETURN
```
©

COMPUTE! has established review panels, each consisting of three or more reviewers whose backgrounds qualify them to analyze new software or hardware products.

To better prepare COMPUTE!'s readers for their buying decisions, and to present fair reviews, we collect the independent opinions of the panelists into one Overview. We think that you will find COMPUTE! Overviews complete, balanced, and informative.

Compilers take, for example, a BASIC program and translate it into a machine language-like form. It should then execute for faster than the original BASIC.

COMPUTE! Overview:

The Galfo Apple Compiler

The Integer BASIC Compiler from Galfo Systems contains two diskettes: one is a system diskette and the other is a compiler diskette.

Compilers are being marketed today primarily because BASIC is slow in execution speed – painfully slow for some applications. This is so because each BASIC command or instruction in a program must be converted by the interpreter before it can be understood and processed by the CPU (central processing unit). This conversion must be done each time the command or instruction is encountered in the program, thus contributing to the slow speed of a BASIC program. Compilers enjoy their speed advantage because these conversions are done only once – *before* the finished (or *compiled*) program is run. After the BASIC program is compiled, it will execute like a machine language program. The Galfo compiler creates code that is comparable to machine language in execution time. It will deliver truly fast programs.

There are several important and desirable features to look for when selecting a compiler. Can the compiled program be stored in any portion of memory desired? With what type of program information is the user furnished after the program is compiled? What is the speed advantage of the compiler? Is the compiled program (object code) longer (and if so, how much longer) than the original program (source code)? How good is the error-handling capability of the compiler? Is it easy to use? Is the documentation complete and easy to follow? Will the compiler handle the many different types of routines demanded by the user, such as graphics, string handling, and I/O routines?

The Galfo compiler allows users of Integer BASIC to make their programs not only faster, but also smaller! This somewhat paradoxical situation is due to the fact that the Integer BASIC Compiler (or IBC) can produce two varieties of output at the user's option:

1. Pure GSL Code
2. Mixed GSL and 6502 Code

GSL stands for "Galfo Stack Language," and is the machine code for an idealized 6502 stack-oriented computer. (Compare Sweet-16 code, which is the machine code for an idealized 16-bit, register-oriented, 6502-like machine). The GSL generated by IBC for a typical Integer BASIC program is more compact than the corresponding tokenized internal form used by the Integer BASIC Interpreter. Even with the addition of 6502 code mixed with the GSL code, reasonably compact programs are achieved. Hence, smaller programs!

...allows users of Integer BASIC to make their programs not only faster, but also smaller!

A runtime system is used to execute the GSL object code. The GSL.SYS program is a Galfo Stack Computer Emulator (Again, compare this to the Sweet-16 program, which is a Sweet-16 Computer Emulator).

It does not have to perform any translation tasks such as the Integer BASIC Interpreter does – such as locating variables or line numbers by searching through memory, converting numbers from ASCII strings to binary, etc. Therefore, it executes the same BASIC program much faster than the Integer BASIC Interpreter can.

About the only overhead is the "fetch-execute" cycle in the emulation. This means retrieving the next Galfo Stack Computer opcode from the object code and dispatching a call to the appropriate subroutine to emulate that opcode. In programs which have a mixture of GSL code and 6502 machine code, this overhead is totally eliminated for those parts of the program in 6502 code. The upshot of all this is that the IBC compiled programs run from 7 to 50 times as fast as their interpreted counterparts. This is the claim made by the documentation. Let's examine the claim.

Benchmarks

The Instruction Manual provided with the compiler contains an Appendix listing some benchmark data. The programs tested to provide that data were obtained from articles appearing in *Kilobaud* magazine, in 1977. The measured speeds for those programs using the normal Integer BASIC Interpreter ranged from 1.4 to 28 seconds. The Appendix lists two corresponding sets of numbers for the same programs when compiled under IBC. One set measures them as compiled to pure GSL code, and another set measures them as compiled to mixed GSL and 6502 code. The data and one of the benchmarks are shown in Figure 1.

Figure 1.

Program #	IBC/GSL (Opt. speed)	IBC/GSL (Opt. space)	APPLE Integer BASIC	APPLE Applesoft BASIC
BM1	0.16	0.16	1.4	1.3
BM2	0.33	0.46	3.2	8.0
BM3	1.5	1.8	8.0	16
BM4	1.0	1.2	7.0	17
BM5	1.2	1.3	9.0	19
BM6	2.1	2.3	18	28
BM7	2.9	3.4	28	45

Listing of BM7:

```
300 PRINT "START"
400 K=0
430 DIM M(5)
500 K=K+1
510 A=K/2*3+4-5
520 GOSUB 820
530 FOR L=1 TO 5
535 M(L)=A
540 NEXT L
600 IF K<1000 THEN 500
700 PRINT "END"
800 END
820 RETURN
```

BM7 was compiled in an attempt to verify the claimed data. The data were gathered using a stop watch, and may not be as accurate as data obtained on a system with a realtime clock. (The Instruction Manual does not comment on how its data were obtained).

These results show that IBC is able to produce most efficient code which really zips along. In fact, the author's suggestion that IBC is the fastest 6502-based high-level language just may be accurate.

Other Speed Tests

A compiler gains much of its speed because there is no longer any need for interpreting each statement. This means different programs will speed up by differing amounts. For example, compiling a FOR-NEXT loop containing the multiplication of decimal numbers will not speed up very much, as most of the time is used in the multiplication. The same loop multiplying Integers will undergo a great increase in speed.

The compiler was also tested using the Benchmark program from *Call-Apple,* March/April, 1980, with the loops increased to 10000. The Benchmark programs do the following: 1. Simple FOR/NEXT loop 2. IF/THEN loop 3. compute using loop variable 4. compute using constants 5.GOSUB 6. GOSUB with additional loop 7. Storing a variable in an array.

The compiler can compile either for speed or space. The speed difference between the two was about 10%. The increase in speed compared to normal Integer BASIC is tremendous. Time is in seconds. Again, timing was done with a stop watch and not a built-in clock. The shorter times are, therefore, somewhat inaccurate:

TEST #	1	2	3	4	5	6	7
INTEGER BASIC	14	35	81	46	130	245	360
COMPILED FOR SPACE	2	6	19	6	14	24	34
COMPILED FOR SPEED	2	4	17	4	12	22	30

Compile time for the Benchmark program was about five seconds.

The compiler was tested under actual conditions. Four pre-existing Integer BASIC programs were compiled using both GSL code (conserving disk space) and mixed code (for optimum program speed). One of the programs compiled was a pinball game. Have you ever tried to play computer-pinball while the ball literally flies across the screen? The compiled program ran so fast that it was impossible to play the game. A second Integer program that played a musical tune was compiled. The individual notes sounded in such rapid succession that the tune played like a continuous musical slur. The individual notes no longer seemed to be separate notes. The time required to *compile* these programs was of such short duration that it bears little mention. Suffice to say that a typical 300 line program compiled in seconds!

There were a few surprises in store when comparing the disk space used by a compiled program to the original program's disk space. When compiling for maximum speed, disk space used increased from a low of 29% to a high of 58% more than the original program. When compiling for maximum economy in disk space, the space used by the compiled program actually *decreased* in every case! Space *saved* ranged from a low of 20%

to a high of 40% of the original program's disk space usage. Additionally, programs compiled for disk space ran almost as fast as those compiled for speed.

The compiler can be used with one disk drive, but this creates an awkward operating situation. It is recommended that the compiler be utilized in a two-drive system.

Using The Compiler

Space does not permit a complete step-by-step description of the compilation process, so the most important procedures and features will be described. Always begin operating by booting the *system* diskette in drive 1. This automatically loads the routines necessary for proper operation of the compiler. It is suggested that a cold start be effected if any utility programs are present in RAM such as the Program Line Editor. Various problems were encountered while attempting to compile with utility programs in RAM. Compiling after a cold start eliminates these problems.

After the system diskette is booted, load the program you wish to compile in the usual manner (from drive 2). Replace your program diskette with the *compiler* diskette and type "BRUN IBC,D1". This command begins the compilation process. The first prompt you will see asks whether you wish to compile using execution speed or disk-space as a priority. This compiler produces code that is *so* fast and so compact, that this option usually becomes a relatively minor consideration. The resultant object code runs almost as fast, and in many cases *as fast*, as a compilation for speed.

The user is then given a number of additional options such as choosing a starting address for the object code and executing or saving the program on disk (as a binary file). An excellent feature is that the object code produced by this compiler will run on *any* Apple II computer. This means that an Integer BASIC program compiled with the Galfo compiler will run on an Apple II+ (a system without Integer BASIC in ROM). This feature was tested, and the code does indeed execute problem-free.

There is a short (3K) program entitled, "GSL.SYS" which must be present on your diskette in order to run the object code (compiled program). The procedure is simple. Using Apple's FID copy program, copy the GSL.SYS program from the "system" diskette to your program diskette. Any compiled program that is BRUN from your program diskette will automatically look for, and then load and run, the GSL. The GSL therefore, must be resident on the program diskette. The GSL program loads at $8800 in memory by default. An advantage noted, when comparing this compiler to others, is that the comparable programs used with the other compilers become an integral part of *each* compiled program. This unnecessarily increases program size and, consequentially, decreases available space on the diskette.

Added Features

Integer BASIC restricts the length of strings to 255 characters. A string length of 32767 is permissible with the Galfo compiler. A symbol table is presented to the user after compilation, listing all variables encountered in the program, the type of variable, (string, integer, or array) and the location of each in memory. A method is provided whereby the user can trick the computer and cause two variables to share the same memory location. This enables the user to refer to either variable during the course of a program and yet retrieve the same information from the variable table. The method used is simple, fully described in the manual, and can easily be managed by even a novice programmer.

The compiler provides the user with many new and modified commands, too numerous to be fully documented in this review. These commands are implemented by typing DSP before each command. Some of the commands that can follow the DSP prefix are: HOME, CLEAR, INVERT, FLASH, NRML, FULL, MIXED, LO, HI, H2, POINT, LINE, and SHAPE. These commands control printing to the CRT, and graphics implementation. If you inspect the commands, you will realize that they emulate the commands available to you in Applesoft. These commands are especially useful as they eliminate the need for the usual cumbersome POKEs (their counterparts in Integer BASIC). The author of the program has ingeniously reconfigured Apple's DSP command, and used it to his advantage for the special operatives.

When writing an original program for compilation or when converting an existing one, the user should be aware of two potential trouble spots. DIM statements *must* be defined by using integers, as the compiler will not function with variables in DIM statements. Additionally, variables used in GOTO's or GOSUB's will cause a large increase in program size. This is because the compiler builds a variable table which is searched each time a variable is encountered in a GOSUB or GOTO statement.

Compiled programs are callable from either Integer or Applesoft. However, after running the program by calling it from Applesoft BASIC, strange errors occur unless the memory pointers are reset by doing an FP. Whether calling compiled Integer BASIC programs from Applesoft is a useful capability is difficult to say. The license to use the run-time system in programs for sale is stated as $5. If this is a one-time fee, it is reasonable. It is not clear whether this fee is for each program sold or is a sub-license fee.

A Few Minor Caveats

The compiler enables the TRACE mode while compiling, and the computer is left in this mode after the compilation is done. The user must then manually execute the NOTRACE command. This may seem like a trivial problem until you run the program only to have line numbers begin printing on the CRT each time. Also, there is no provision made whereby the user may protect a given portion of memory. For example, if one wished to protect the HI-RES page from being overwritten, the only way found to easily effect this was to load the entire object code *after* the HI-RES page in memory.

Several errors of omission were encountered, although none were serious. For example, a file entitled "HI-RES Driver" is needed to utilize the HI-RES routines supplied. The manual states that the file must be *loaded* for the routines to work, while in actuality it must be *run*. Additionally, no mention is made as to how one may automatically effect this. If Apple's HELLO program is used to BRUN the HI-RES Driver, then, after the HI-RES file is loaded, the program will stop. An EXEC file must be created in order to first BRUN the HI-RES Driver and then BRUN your own compiled program. No mention is made of this in the manual.

General Overview

- Panelist #1: "Generally great. Look forward to Galfo Applesoft Compiler."
- Panelist #2: "For heavy users of Integer BASIC, the IBC is well worth having. It should enhance existing programs as well as open many new avenues of application that were formerly unavailable for reasons of performance."
- Panelist #3: "The Galfo Integer BASIC Compiler is an extremely useful utility. Its advantages and features far outweigh its shortcomings (which are few) and future revisions will almost certainly correct these. Source code is compiled extremely fast and object code produced by this compiler executes almost as fast as pure machine language. This is a package that all Integer BASIC programmers should own."

Galfo Integer BASIC Compiler. Galfo Systems, 6252 Camino Verde, San Jose, CA, 95119. $149.50. ©

The Atari GTIA Chip: Here At Last!

Steve Steinberg
Washington, DC

If you've never heard of the legendary Atari GTIA chip, here in brief is the story:

The designers of the Atari 400/800 computers originally had planned on three additional graphics modes – Graphics 9 to provide 16 intensity levels of one color, Graphics 10 to provide nine colors in varying intensities, and Graphics 11 to provide 16 colors. Both Atari BASIC and the Computer's Operating System were designed to access these three graphics modes, but at the time the Atari computers were first released, in 1979, there were still some bugs to be ironed out in the graphics chip, known as the GTIA. The company decided to go ahead, using a less powerful chip, the CTIA, without these three modes.

The bugs in GTIA have long since been dealt with and the question for some time for many of us has been just how and when Atari would release the chip. The GTIA is now being installed in all Atari 400s and 800s coming off the production line and the really good news for those of us who have waited and wondered and hoped is that it is also being shipped to facilities that service Atari computers, to upgrade existing models. I paid $23.92 for mine and, although installation costs depend, of course, on who you are dealing with, replacement of the CTIA is a relatively uncomplicated procedure. I have also heard that it is simple enough that some computer stores are doing it on a "while you wait" basis.

The Legend Is True

The best news of all is that the legend turns out to be quite accurate, and the chip does everything it was said to do. Documentation, as of this writing, is a bit hard to come by, so here is a preliminary effort. One warning for recent Atari owners. If you type in the accompanying programs and get nothing on your screen but a blue (default) background and a series of graphics control characters, it means you don't have the new chip. My suggestion is to buy one.

The first brief program gives you 15 color bars against a black background in Graphics mode 11, the bars representing colors 1 through 15 as described in the table on page 50 of the *Atari Operators Manual.* As you can see, these colors are accessed in BASIC by simply calling up colors 1 through 15. If the colors are off, you may want to adjust the tint controls on your TV set.

The colors in Graphics 11 are set at a default intensity, but you can vary it. Insert a line 15 SETCOLOR 4,0,0. The four does nothing (I'm simply used to it for setting background colors), but the first zero sets the background as black and the second zero sets the luminance at its darkest shade. You can change the background color by changing the second zero to any number up to 15 and you can change the intensity of the 15 displayed colors just as you would with Graphics modes 0 through 7, by changing the last number.

Next, change line 10 to Graphics 9. As you can see, you now have 16 shades of whatever color you have entered in the SETCOLOR command. It's worth noting here that, while all other Graphics modes give you only eight color intensity levels, this one gives you 16, which means that the total number of colors accessible on the Atari computer are now doubled, from 128 to 256 colors altogether.

Graphics mode 10 is a bit more complex than the other two. As I've noted, this is strictly a preliminary look at the GTIA, but let's try it now by changing line 10 to GRAPHICS 10. If you don't have a SETCOLOR statement in line 15, or if the statement is 4,0,0 you should get a long band of black, four different colors, another band of black and the four colors repeated.

Try turning the color level all the way down on your television and you will see that you do have different intensities in the different colors. But there are only five here, where are the other four? First, enter a SETCOLOR in line 15, for example SETCOLOR 4,5,12. The second band of black should now become light violet. Now, enter line 16 POKE 704,15:POKE 705,32:POKE 706,42: POKE 707,52. POKEing into those four memory locations will give you additional hues and intensities, enough for a total of nine colors, including the background. This is as far as my knowledge of this graphics mode has taken me, to date, although I am sure there are many other possibilities to be explored.

Program 2 is simply a primitive attempt at a demo of these dramatic new graphics capabilities for the Atari. All three modes give you 80 pixels across by 192 down. Each of these pixels is actually four pixels wide, which means, for example, that drawing a square requires that the vertical length should be four times the horizontal length.

The large number of variables in line 10 of Program 2 were designed to give you an opportunity to do as much fiddling around with the program as you choose. While it is in GRAPHICS 10, it's no problem to try out the other two new modes by changing line 20. If you get tired of squares, and like the works of Piet Mondrian, eliminate line 70. You should enjoy the results.

One final comment seems appropriate on the occasion of this first hardware update for the Atari computers. All of us who have worked with the Atari, and especially those of us who became introduced to computer programming through the Atari, know that we have one magnificent machine. With the addition of the GTIA chip, its graphics capabilities far surpass those of any other home computer on the market. We were told, when we made the decision to buy, that this was a "friendly" computer, and one that wouldn't be outdated overnight. I feel that this is one company that has lived up to its promises. I have found in my own dealings with people at Atari that if the computer occasionally isn't all that friendly, they certainly are. I'd simply like to express my thanks to, and admiration for, all of them.

Program 1.

```
10 GRAPHICS 11
20 C=1
30 COLOR C
40 A=A+1
50 PLOT A,1:DRAWTO A,180
60 IF A=79 THEN GOTO 60
70 IF A/5=INT(A/5) THEN C=C+1:GOTO 30
80 GOTO 40
```

Program 2.

```
10 M=64:N=150:P=6:Q=24:R=150:T=12:U=T:Y=
1:C=A+P:D=B+Q
20 GRAPHICS 10
30 SETCOLOR 4,5,10
40 POKE 704,15:POKE 705,28:POKE 706,33:P
OKE 707,54
50 A=(INT(RND(1)*M)+1)
60 B=(INT(RND(1)*N)+1)
70 C=A+P:D=B+Q
80 FOR Z=1 TO T:SOUND 0,A+B,10,U-Z:NEXT
Z
90 FOR Z=1 TO T:SOUND 1,A,10,U-Z:NEXT Z
100 Z=(INT(RND(0)*15)+1)
110 COLOR Z
120 K=INT(RND(1)*3)
130 IF K=2 THEN GOSUB 180
140 PLOT A,B:DRAWTO C,B:DRAWTO C,D:DRAWT
O A,D:DRAWTO A,B-1:DRAWTO C,B-1
150 SOUND 0,0,0,0:SOUND 1,0,0,0:SOUND 2,
0,0,0
160 IF Y>R THEN GOTO 150
170 GOTO 50
180 PLOT A,B:DRAWTO C,B:DRAWTO C,D:DRAWT
O A,D:DRAWTO A,B:DRAWTO C-1,B:DRAWTO C-1
,D-1:DRAWTO A+1,D-1
190 DRAWTO A+1,B+1:DRAWTO C-2,B+1:DRAWTO
 C-2,D-2:DRAWTO A+2,D-2:DRAWTO A+2,B+2:D
RAWTO C-3,B+2:DRAWTO C-3,D-3
200 FOR Z=1 TO T:SOUND 2,A/2,12,U-Z:FOR
W=1 TO 20:NEXT W:NEXT Z
210 GOTO 50
```

©

This program lets you visualize geometry. It's for the Apple (Applesoft with 48K memory) and for the Atari.

Computer-Assisted Geometry Discovery

Fred Ventura
Westlake Village, CA

During the summer of 1981, Intergate, Incorporated of Westlake Village, California offered a number of courses for gifted and talented youngsters. We selected the Atari 800 computer for our computer literacy and computer programming courses. Our program presented a unique opportunity for exceptional children ranging in age from seven to 14 to be challenged by the computer. The courses included an introduction to computer programming, problem solving with the computer, computer-assisted algebra, and an interesting course in which parents and children worked together to learn about computers.

One student, Ricky Masters of Oxnard, California, age 11, developed a clever program that might be of interest to mathematics teachers. The program allows the user to specify the directions for a series of vectors which are simultaneously plotted on the screen in graphics mode seven. From a starting point in the center of the screen, any of eight directions can then be chosen. A maximum of one thousand vectors can be used with 32K RAM, which is more than enough for very complicated designs.

By entering the digit "0" for a vector, the screen is then cleared and the computer proceeds to redraw the design. However, the program takes the drawing one step further and reflects the figure across the x and y axis.

It was rewarding as a teacher to share in the excitement with which each student watched the computer transform a simple design into an intricate, symmetrical computer graphic display. But, more than providing an amusing experience, the program can be used by teachers as a springboard for the study of the geometry of reflection.

Valuable mathematical discoveries were made by the students who used the program. For example, figures which were drawn so that the intersection of lines of symmetry which corresponded with the center of the graph did not change when reflected, (See example 1). Also, the end points of rotated vectors are determined by a change in sign of the magnitude which is held constant. This can be seen by examining the subroutines used for plotting the reflected figure (starting at lines 90, 510, 710, and 910). It should be pointed out to students who use the program that vector 1 is the result of addition of vectors 2 and 4. This also applies to vectors 3, 6 and 8. In this way, the concept of addition of vectors can be introduced to students.

Students were fascinated by the optical illusion, in many of the designs, which seemed to move in and out from the screen. Follow-up activities for students who used the program were provided to extend the student's learning. Students were presented with figures drawn in one quadrant of a Cartesian plane and were asked to predict what the reflected image would look like. Discussions of integers and their opposites were also used in an analysis of the plotting subroutines.

I found that using the computer to illustrate a rather abstract mathematical concept leads to greater understanding. The learning process was an enjoyable experience for all.

The program described above was originally written on an Atari 800. An adapted version was prepared for the Apple computer with Applesoft and 48K. Both listings are presented for comparison. In the Apple version, improvements were made to the program so that either reflection or rotation can be selected. In addition, option (O) permits the overlay of the reflected figure on the rotated figure. Some of the syntactical differences of the two BASICs can be studied by examining the two programs.

Examples And Challenges:

1. Enter a magnitude of 10 and the following series of vectors: 3,4,4,8,8,4,4,3 and 0 to reflect. You will notice that, since the hourglass shape is symmetrical and centered, no change occurs when reflected.

2. Try to enter the correct vectors to make a 3-d cube. When reflected, this figure makes an interesting illusion.

3. Enter the vectors for a complicated figure and predict the reflected image.

Program 1: Atari Version

```
5 DIM V(1000)
6 A=79:B=40
10 GRAPHICS 0
11 COLOR 2
15 POSITION 5,7
20 PRINT "COMPUTER GRAPHICS PROGRAM"
21 PRINT :PRINT "CODES: "
22 PRINT "(1-8) DIRECTIONS FOR VECTORS"
23 PRINT "(9)   RESET DRAW MODE"
24 PRINT "(0)   REFLECT FIGURE"
25 PRINT "(10)  COLOR OFF"
26 PRINT "(11)  COLOR ON"
30 PRINT
50 PRINT "ENTER A LENGTH FOR EACH SEGMEN
T (1-10)"
60 PRINT "     >>>";:INPUT L
70 GRAPHICS 7:PLOT A,B
75 POSITION 20,30:? " 1 2 3 "
76 POSITION 20,30:? " 4 * 5 "
77 POSITION 20,30:? " 6 7 8 "
80 N=N+1:IF N=1000 THEN 200
85 PRINT "VECTOR ";N;": ";:INPUT V:V(N)=
V
86 GOSUB 90:GOTO 75
90 IF V=1 THEN A=A-L:B=B-L:GOTO 190
100 IF V=2 THEN B=B-L:GOTO 190
110 IF V=3 THEN A=A+L:B=B-L:GOTO 190
120 IF V=4 THEN A=A-L:GOTO 190
130 IF V=5 THEN A=A+L:GOTO 190
140 IF V=6 THEN A=A-L:B=B+L:GOTO 190
150 IF V=7 THEN B=B+L:GOTO 190
160 IF V=8 THEN A=A+L:B=B+L:GOTO 190
170 IF V=9 THEN 300
175 IF V=10 THEN COLOR 0
177 IF V=11 THEN COLOR 2
180 IF V=0 THEN GOTO 400
190 IF A<0 THEN A=0
191 IF A>157 THEN A=157
192 IF B<0 THEN B=0
193 IF B>79 THEN B=79
200 DRAWTO A,B:RETURN
300 A=79:B=40:GRAPHICS 7:PLOT A,B
310 N=0
320 GOTO 75
400 GRAPHICS 7:A=79:B=40:PLOT A,B
410 FOR I=1 TO N-1:V=V(I):GOSUB 2000:GOS
UB 90:NEXT I
415 A=79:B=40:PLOT A,B
420 FOR I=1 TO N-1:V=V(I):GOSUB 2000:GOS
UB 510:NEXT I
430 A=79:B=40:PLOT A,B
435 FOR I=1 TO N-1:V=V(I):GOSUB 2000:GOS
UB 710:NEXT I
440 A=79:B=40:PLOT A,B
445 FOR I=1 TO N-1:V=V(I):GOSUB 2000:GOS
UB 910:NEXT I
449 A=79:B=40:PLOT A,B
450 FOR I=1 TO N-1:V=V(I):GOSUB 90:NEXT
I:N=N-1:GOTO 75
500 GRAPHICS 7:REM DRAW AND MIRROR
505 N=0:GOTO 75
510 IF V=1 THEN A=A+L:B=B-L:GOTO 590
515 IF V=2 THEN B=B-L:GOTO 590
520 IF V=3 THEN A=A-L:B=B-L:GOTO 590
530 IF V=4 THEN A=A+L:GOTO 590
540 IF V=5 THEN A=A-L:GOTO 590
550 IF V=6 THEN A=A+L:B=B+L:GOTO 590
560 IF V=7 THEN B=B+L:GOTO 590
570 IF V=8 THEN A=A-L:B=B+L:GOTO 590
580 IF V=10 THEN COLOR 0
585 IF V=11 THEN COLOR 2
590 IF A<0 THEN A=0
600 IF A>157 THEN A=157
610 IF B<0 THEN B=0
620 IF B>79 THEN B=79
630 DRAWTO A,B:RETURN
710 IF V=1 THEN A=A-L:B=B+L:GOTO 790
715 IF V=2 THEN B=B+L:GOTO 790
720 IF V=3 THEN A=A+L:B=B+L:GOTO 790
730 IF V=4 THEN A=A-L:GOTO 790
740 IF V=5 THEN A=A+L:GOTO 790
750 IF V=6 THEN A=A-L:B=B-L:GOTO 790
760 IF V=7 THEN B=B-L:GOTO 790
770 IF V=8 THEN A=A+L:B=B-L:GOTO 790
780 IF V=10 THEN COLOR 0
785 IF V=11 THEN COLOR 2
790 IF A<0 THEN A=0
800 IF A>157 THEN A=157
810 IF B<0 THEN B=0
820 IF B>79 THEN B=79
830 DRAWTO A,B:RETURN
910 IF V=1 THEN A=A+L:B=B+L:GOTO 990
915 IF V=2 THEN B=B+L:GOTO 990
920 IF V=3 THEN A=A-L:B=B+L:GOTO 990
930 IF V=4 THEN A=A+L:GOTO 990
940 IF V=5 THEN A=A-L:GOTO 990
950 IF V=6 THEN A=A+L:B=B-L:GOTO 990
970 IF V=8 THEN A=A-L:B=B-L:GOTO 990
980 IF V=10 THEN COLOR 0
985 IF V=11 THEN COLOR 2
990 IF A<0 THEN A=0
1000 IF A>157 THEN A=157
1010 IF B<0 THEN B=0
1020 IF B>79 THEN B=79
1030 DRAWTO A,B
1050 RETURN
2000 REM SOUND EFFECTS
2010 SOUND 1,INT(RND(1)*255),10,8
2020 FOR T=1 TO 10:NEXT T
```

```
2030 SOUND 1,0,0,0
2040 RETURN
```

Program 2: Apple Version

```
10  REM  ***  GEOMETRIC EXPLORATI
     ONS
20  REM  ***  FRED VENTURA
30  REM  ***  INTERGATE, INCORPOR
     ATED
40  REM  ***  WESTLAKE VILLAGE, C
     A
50  DIM V(1000)
60 X1 = 139:Y1 = 79:C = 3
70 X = X1:Y = Y1
80  TEXT
90  HOME
100  VTAB 2: HTAB 2: PRINT "GEOAR
    T - A COMPUTER GRAPHICS PROG
    RAM"
110  PRINT : INPUT "ENTER THE MAG
    NITUDE OF THE VECTORS: ";M
120  PRINT : PRINT "ENTER THE NUM
    BER FOR THE DIRECTION OF    E
    ACH VECTOR. YOU DO NOT NEED
    TO PRESS   RETURN. OTHER COM
    MANDS ARE: "
130  PRINT : PRINT "  (Q) QUIT.
          (S) START OVER. "
140  PRINT : PRINT "  (O) OVERLAY
     REFLECTED FIGURE. "
150  PRINT : PRINT "  (F) REFLECT
    .     (T) ROTATE."
160  PRINT : PRINT "  (C) CLEAR H
    IRES SCREEN. "
170  PRINT : PRINT "PRESS RETURN
    TO CONTINUE. ";: INPUT X$
180  HGR
190  VTAB 21: HTAB 30: PRINT "1 2
     3"
200  VTAB 22: HTAB 30: PRINT "4 *
     5"
210  VTAB 23: HTAB 30: PRINT "6 7
     8"
220 N = N + 1: VTAB 23: HTAB 1: PRINT
    "ENTER VECTOR ";N;": ";: GET
    V$
230  IF V$ = "Q" THEN  END
240  IF V$ = "S" THEN  RUN
250  IF V$ = "O" THEN  GOTO 330
260  IF V$ = "C" THEN  HGR : GOTO
     220
270 I = N:V(I) =  VAL (V$)
280  IF V$ = "F" THEN 310
290  IF V$ = "T" THEN 880
300  GOSUB 380: GOTO 220
310  REM  REFLECT FIGURE
320  HGR
330 X = 139:Y = 79: FOR I = 1 TO
    N - 1: GOSUB 380: NEXT I
340 X = 139:Y = 79: FOR I = 1 TO
    N - 1: GOSUB 480: NEXT I
350 X = 139:Y = 79: FOR I = 1 TO
    N - 1: GOSUB 580: NEXT I
360 X = 139:Y = 79: FOR I = 1 TO
    N - 1: GOSUB 680: NEXT I
370  GOTO 220
380  REM  PLOT VECTORS
390  IF V(I) = 1 THEN X = X - M:Y
     = Y - M
400  IF V(I) = 2 THEN Y = Y - M
410  IF V(I) = 3 THEN X = X + M:Y
     = Y - M
420  IF V(I) = 4 THEN X = X - M
430  IF V(I) = 5 THEN X = X + M
440  IF V(I) = 6 THEN X = X - M:Y
     = Y + M
450  IF V(I) = 7 THEN Y = Y + M
460  IF V(I) = 8 THEN X = X + M:Y
     = Y + M
470  GOSUB 780: RETURN
480  REM   PLOT VECTORS Q2
490  IF V(I) = 1 THEN X = X + M:Y
     = Y - M
500  IF V(I) = 2 THEN Y = Y - M
510  IF V(I) = 3 THEN X = X - M:Y
     = Y - M
520  IF V(I) = 4 THEN X = X + M
530  IF V(I) = 5 THEN X = X - M
540  IF V(I) = 6 THEN X = X + M:Y
     = Y + M
550  IF V(I) = 7 THEN Y = Y + M
560  IF V(I) = 8 THEN X = X - M:Y
     = Y + M
570  GOSUB 780: RETURN
580  REM   PLOT VECTORS Q3
590  IF V(I) = 1 THEN X = X + M:Y
     = Y + M
600  IF V(I) = 2 THEN Y = Y + M
610  IF V(I) = 3 THEN X = X - M:Y
     = Y + M
620  IF V(I) = 4 THEN X = X + M
630  IF V(I) = 5 THEN X = X - M
640  IF V(I) = 6 THEN X = X + M:Y
     = Y - M
650  IF V(I) = 7 THEN Y = Y - M
660  IF V(I) = 8 THEN X = X - M:Y
     = Y - M
670  GOSUB 780: RETURN
680  REM   PLOT VECTORS Q4
690  IF V(I) = 1 THEN X = X - M:Y
     = Y + M
700  IF V(I) = 2 THEN Y = Y + M
710  IF V(I) = 3 THEN X = X + M:Y
     = Y + M
720  IF V(I) = 4 THEN X = X - M
730  IF V(I) = 5 THEN X = X + M
740  IF V(I) = 6 THEN X = X - M:Y
     = Y - M
```

```
750  IF V(I) = 7 THEN Y = Y - M
760  IF V(I) = 8 THEN X = X + M:Y
      = Y - M
770  GOSUB 780: RETURN
780  REM  PLOT
790  HCOLOR= C
800  IF I = 1 THEN  HPLOT 139,79
810  IF X < 0 THEN X = 0
820  IF X > 279 THEN X = 279
830  IF Y < 0 THEN Y = 0
840  IF Y > 159 THEN Y = 159
850  HPLOT  TO X,Y
860  VTAB 21: HTAB 3: PRINT "X=";
     X;" Y=";Y;"   "
870  RETURN
880  REM  ROTATE FIGURE
890  HGR
900 X = 139:Y = 79: FOR I = 1 TO
     N - 1: GOSUB 380: NEXT I
910 X = 139:Y = 79: FOR I = 1 TO
     N - 1: GOSUB 950: NEXT I
920 X = 139:Y = 79: FOR I = 1 TO
     N - 1: GOSUB 580: NEXT I
930 X = 139:Y = 79: FOR I = 1 TO
     N - 1: GOSUB 1050: NEXT I
940  GOTO 220
950  REM  ROTATE Q2
960  IF V(I) = 1 THEN X = X - M:Y
      = Y + M
970  IF V(I) = 2 THEN X = X - M
980  IF V(I) = 3 THEN X = X - M:Y
      = Y - M
990  IF V(I) = 4 THEN Y = Y + M
1000  IF V(I) = 5 THEN Y = Y - M
1010  IF V(I) = 6 THEN X = X + M:
     Y = Y + M
1020  IF V(I) = 7 THEN X = X + M
1030  IF V(I) = 8 THEN X = X + M:
     Y = Y - M
1040  GOSUB 780: RETURN
1050  REM  ROTATE Q4
1060  IF V(I) = 1 THEN X = X + M:
     Y = Y - M
1070  IF V(I) = 2 THEN X = X + M
1080  IF V(I) = 3 THEN X = X + M:
     Y = Y + M
1090  IF V(I) = 4 THEN Y = Y - M
1100  IF V(I) = 5 THEN Y = Y + M
1110  IF V(I) = 6 THEN X = X - M:
     Y = Y - M
1120  IF V(I) = 7 THEN X = X - M
1130  IF V(I) = 8 THEN X = X - M:
     Y = Y + M
1140  GOSUB 780: RETURN
```

©

This article shows beginners how to use joysticks with the VIC-20 and it also contains an excellent game.

The Joystick Connection: Meteor Maze

Paul L. Bupp and Stephen P. Drop
Kent, WA

Let the games begin! Your VIC can be easily connected to the readily available Atari joysticks. We will show the new VIC-20 owner how to use these joysticks. Also there's a new VIC game called "Meteor Maze," which demonstrates the use of the joystick.

Program 1.

```
10 PRINT CHR$(147)
15 PRINT SPC(3) CHR$(95) "JOYSTICK
    DIRECTION"
20 PRINT SPC(3) CHR$(95) "BUTTON"
25 PRINT SPC(177) CHR$(144) "** JO
    YSTICK DEMO **"
30 POKE 37154,127
35 PRINT CHR$(19)
40 A = ( PEEK(37137) AND 28) OR (P
    EEK(37152) AND 128)
45 A = ABS((A-100)/4)-7
50 ON A GOSUB 100,110,120,,130,140
    ,150,,,,160,170,180
55 B = PEEK(37137)AND 32
60 PRINT CHR$(19)
65 PRINT
75 IF B GOTO 85
80 PRINT " ON": GOTO 90
85 PRINT "OFF"
90 POKE 37154,255
95 GET A$: IF A$ = "" GOTO 30
99 END
100 PRINT " SW": RETURN
110 PRINT " NW": RETURN
120 PRINT "  W": RETURN
130 PRINT "  S": RETURN
140 PRINT "  N": RETURN
150 PRINT "   ": RETURN
160 PRINT "  E": RETURN
170 PRINT " NE": RETURN
180 PRINT " SE": RETURN
```

But first, let's look at how the joystick connects to the VIC. Program 1 is a BASIC joystick demonstration program. A line-by-line description of the program follows:

10 PRINT CHR$(147)
15 PRINT SPC(3) CHR$(95) "JOYSTICK DIRECTION"
20 PRINT SPC(3) CHR$(95) "BUTTON"

Lines 10 through 20 clear and print the display screen used by the program. CHR$(95) prints the left arrow.

25 PRINT SPC(177) CHR$(144) " JOYSTICK DEMO **"**

Line 25 uses the SPC command to print 177 spaces, then a CHR$(144) turns on the Black print mode before printing the title of the program.

30 POKE 37154,127

This line resets the direction of the 6522 *A* side Data Direction register which was already set by the system to check the keyboard. The other Data Direction register used by the joystick is already set by the system default. (Note: With this register altered, some keys will now no longer be recognized by the system. See important note to line 90 below.)

35 PRINT CHR$(19)

This line homes the cursor to the top left of the screen.

40 A=(PEEK(37137) AND 28) OR (PEEK(37152) AND 128)

This line pulls together the two input register values used by the joystick and combines them to make a single value (A).

45 A=ABS((A-100)/4)-7

This line reduces the joystick value (Variable A) from line 40 to a simple number value between one and thirteen, with some number values (four and eight through ten) not being used. This value is kept in variable A. For each direction of the joystick, Figure #1 provides a visual display of the original value (boxed) and the condensed value placed in variable A by lines 40 and 45.

50 ON A GOSUB 100,110,120,,130,140,150,,,,160, 170,180

This line directs the program to go to the chosen joystick direction subroutine and then return to the next line of BASIC.

55 B=PEEK(37137) AND 32

Here the variable B is set to zero if the joystick button is pushed, or set to 32 if it is not being pushed. This PEEK is looking at only the one bit which shows whether the button has been pushed by the player.

```
60 PRINT CHR$(19)
65 PRINT
```

These two lines home the cursor and move it down one line to place it at the right location on the screen to print whether the button is ON or OFF.

```
75 IF B GOTO 85
```

This IF statement only goes to line 85 if the variable B is not zero. In this case, it means the button was not pushed. Otherwise the IF fails and the BASIC program proceeds to the next statement.

```
80 PRINT " ON":GOTO 90
85 PRINT "OFF"
```

These lines print whether the button is ON or OFF depending on the IF statement in line 75.

```
90 POKE 37154,255
```

This line resets the Data Direction register altered in line 30. This internal system register is used to check the keyboard. With this POKE, all the keys are again recognized by the VIC-20. If, in your program, the STOP button is pushed, or for some other reason the program is accidently stopped, this Data Direction register becomes correctly reset by using the RUN/STOP and RESTORE button combination or by using POKE 37154,255.

```
95 GET A$:IF A$ = "" GOTO 30
```

This line of BASIC provides a way to end the program by looking at the keyboard input buffer with the GET, and then, if no key has been pushed, the program branches back to line 30 to begin again.

Lines 100 through 180 are the direction indicator subroutines which print the direction indicator at the top of the screen. These are reached from line 50 above.

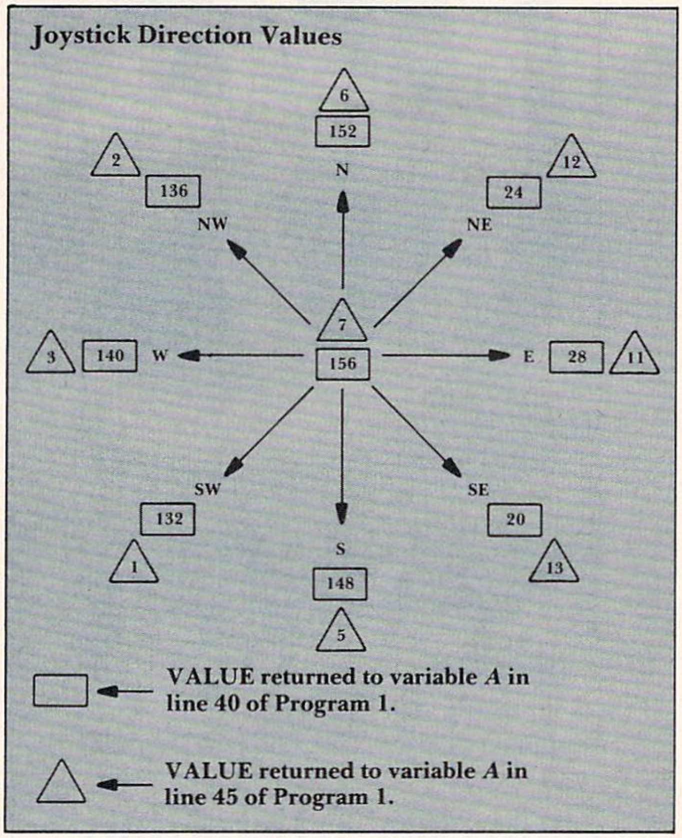
This BASIC program is written in "portable" code which can be simply typed into the VIC-20. However, for those planning to include the Joystick Connection in their own programs, the routine can be speeded up and condensed to take up less room. A condensed version of the same routine is included in the Meteor Maze game (Program 2). That completes the discussion of the Joystick Connection and how it works.

Now, let's look at the game program "Meteor Maze."

Meteor Maze

Meteor Maze is a fast game using the Joystick Connection routine described above. The object of the game is to move your Scout Ship through the meteor field to the Base Ship at the bottom of the screen as quickly as possible. Two levels of play are available, Novice and Advanced. The difference in levels is the computer's tolerance for navigational error. Details of these differences can be displayed by pushing Function Key #1 (F1). Speed is of the essence. The player must learn to manipulate the controls well to be the fastest to reach the Base Ship.

Figure 1.



Program 2.

```
2 PRINT"METEOR MAZE"SPC(96)"FOR I
   NSTRUCTIONS"SPC(34)"PRESS"
   SPC(4Ø)"F1
3 POKE56,28:POKE52,28:CLR:FORA=71
   68TO7375:READB:POKEA,B:NEX
   T
4 V=36878:N=V-1:S=N-2:POKE37154,1
   27:L=768Ø:GOSUB97:F=13:IFG
   THENE=3:GOTO6
5 E=99
6 POKE36869,255:POKEV+1,27:PRINT"
   {OFF}{YEL}{CLEAR}V{BLK}";:
   FORA=1TO48Ø:GETA$:IFA$=CHR
   $(133)GOTO4Ø
7 PRINTMID$("@A@B@C@D@E@F",RND(TI
   )*12+1,1);:NEXT
8 PRINT"{RED}@LM{PUR}{REV} ELAPSE
   D TIME        {OFF}{RED}NO{Ø
   7 LEFT}{REV}{PUR}";:POKE81
   85,16:POKE389Ø5,2:TI$="ØØØ
   ØØØ"
9 A=(PEEK(37137)AND28)OR(PEEK(371
   52)AND128):A=ABS(A-1ØØ)/4-
   7:IFA=7THENA=F
1Ø ONAGOSUB9Ø,91,92,,93,94,,,,,95,
   96,97:F=A
11 A=PEEK(37137)AND32:IFATHENPOKEV
   ,Ø:GOTO27
12 POKEN,255:POKES,22Ø:POKEV,3:IFP
   EEK(C)>6GOTO27
13 IFPEEK(C)GOTO19
14 POKE3Ø72Ø+C,7:POKEC,PEEK(L):POK
   EL,Ø:L=C:IFC<>8161GOTO27
15 GOSUB99:POKE8161,22:PRINT"{13 L
   LEFT}DOCKING";:POKEV,15
16 FORL=1TO4:FORC=18ØTO235STEP2:PO
   KES,C:FORA=1TO1Ø:NEXT:NEXT
   :POKES,Ø
17 FORA=1TO1Ø:NEXT:READA,B:POKE816
   1,A:POKE8162,B:NEXT
18 RESTORE:FORA=1TO2Ø8:READB:NEXT:
   K=Ø:GOTO4
19 ONGGOTO21
2Ø GOSUB99:POKEC,7:POKEN,255:FORA=
   15TOØSTEP-1:POKEV,A:FORB=1
   TO35:NEXT:NEXT:GOTO27
21 GOSUB99:POKEN,22Ø:FORA=16TO1STE
   P-1:POKEV,A:FORB=A*16-1TO(
   A-1)*16STEP-1:POKEV+1,B
23 NEXT:NEXT:POKE36865,132:POKEV+1
   ,59:POKE36869,242:K=K+1:GO
   SUB99
24 PRINT"{CLEAR}{BLK}{OFF}SCOUT"K"
   TO BASE:{DOWN}":PRINT"REQU
   EST"INT(EXP(K))"BOTTLES":P
   RINT"OF SUPER GLUE!
25 FORA=131TOØSTEP-1:POKE36865,A:F
   ORB=1TO45:NEXT:NEXT
26 PRINT"{CLEAR}":GOSUB99:POKE3686
   5,25:GOTO4
27 PRINTMID$(TI$,3,2)":"MID$(TI$,5
   )"{Ø5 LEFT}";
28 GETA$:IFA$=CHR$(133)GOTO4Ø
29 IFA$<>CHR$(135)GOTO9
35 IFE=ØORPEEK(C)>8GOTO9
36 GOSUB99:POKEN,22Ø:FORA=15TOØSTE
   P-1:POKEC,8:POKEV,A:FORB=1
   TO2Ø:NEXT:POKEC,Ø
37 FORB=1TO2Ø:NEXT:NEXT:GOSUB99:E=
   E-1:GOTO9
4Ø POKEV+1,127:POKE36869,242:PRINT
   "{CLEAR}{OFF}{RED}      $$$$
   $$$$$$"SPC(12)"{REV}SELECT
```

```
    ONE{BLK}{DOWN}
41 A$="    -ABLE TO DESTROY":PRINT"
    F1=NOVICE LEVEL":PRINTA$
42 B$="    -SHIP WILL ":PRINT"      9
   9 METEORS":PRINTB$"SURVIVE
   "SPC(5)"METEOR COLLISIONS{
   DOWN}
43 PRINT" F3=ADVANCE LEVEL":PRINTA
   $SPC(7)"3 METEORS
44 PRINTB$"EXPLODE"SPC(5)"ON IMPAC
   T WITH A"SPC(6)"METEOR{DOW
   DOWN}
45 PRINT" F5=INSTRUCTIONS{DOWN}":P
   RINT" F7=END THE GAME{02 D
   DOWN}":PRINT"* CURRENT LEV
   EL
46 IFGTHENPRINT"{HOME}"SPC(198)"*"
   :GOTO48
47 PRINT"{HOME}{03 DOWN}*
48 POKE37154,255:GETA$:IFA$=""GOTO
   48
49 A=ASC(A$)-132:ONABS(A)GOTO51,52
   ,55,80
50 GOTO4
51 G=0:GOTO4
52 G=1:GOTO4
55 POKEV+1,127:PRINT"{BLK}{CLEAR}{
   DOWN}GOAL-MOVE IN FRONT OF
      THE DOCKING BAY AND  T
   HE BASE WILL LAND
56 A$="  YOUR SHIP.{DOWN}":PRINTA$
   :PRINT"JOYSTICK WILL POINT
      "A$
57 PRINT"FIRE BUTTON WILL MOVE "A$
58 PRINT"F1 ALLOWS CHANGE OF     G
   AME DIFFICULTY.{DOWN}
59 PRINT"F5 DESTROYS METEORS      T
   HAT ARE IN FRONT     OF"A$

60 PRINT"{02 DOWN}HIT A KEY TO CON
   TINUE
61 GETA$:IFA$=""GOTO61
62 GOTO40
80 SYS4096
90 POKEL,24:C=L+21:RETURN
91 POKEL,23:C=L-23:RETURN
92 POKEL,20:C=L-1:RETURN
93 POKEL,18:C=L+22:RETURN
94 POKEL,17:C=L-22:RETURN
95 POKEL,19:C=L+1:RETURN
96 POKEL,21:C=L-21:RETURN
97 POKEL,22:C=L+23:RETURN
99 POKEV,0:POKEN,0:POKES,0:RETURN
1000 DATA0,0,0,0,0,0,0,0,0,12,62,127
```

```
,62,28,0,0,32,120,28,62,60
    ,24,0,0
1002 DATA24,60,126,124,60,56,28,0,0,
    70,55,114,120,60,28,0,0,0,
    16,88,124,56,28,0
1004 DATA56,126,60,32,6,15,2,6,10,84
    ,4,161,34,136,133,40,234,5
    6,239,78,98,198,48,96
1006 DATA0,0,0,0,0,114,127,114,0,0,0
    ,0,0,7,7,7,63,83,143,143,1
    37,174,254,174
1008 DATA63,83,143,143,137,142,142,1
    42,255,255,255,255,255,72,
    75,72,0,0,0,0,0,0,127,159
1010 DATA142,137,143,143,83,63,255,6
    3,78,24,255,255,255,255,25
    5,231,8,28,8,8,28,28,28,28

1012 DATA28,28,28,28,8,8,28,8,0,0,11
    4,127,114,0,0,0,0,0,78,254
    ,78,0,0,0
1014 DATA0,6,6,24,56,112,32,0,0,32,1
    12,56,24,6,6,0,0,96,96,24,
    28,14,4,0
1016 DATA0,4,14,28,24,96,96,0,63,83,
    143,143,137,190,190,190
1018 DATA9,12,10,11,0,25,0,12
```
©

An ATARI Learning Program

Language Lab

Steve Steinberg
Washington, DC

Language Lab is a program to use your ATARI to help you build vocabulary in a foreign language. It is basically a computerized version of that old standby of language education, the flash card set, and I have found it extremely simple and effective to use. It is structured so that you can drill and score yourself on as many words as you like, but I find it most useful if you display a fifteen or twenty word vocabulary drill on the screen, spot the errors and review them, then try again. If you have enough memory, you can also use it to create a fair sized foreign language dictionary.

I have used a handful of French words in the program example but you can easily change this to any language you want. Simply change line 55 LANG$="FRENCH" to LANG$="GERMAN", "SPANISH", "NAVAJO" or whatever you like and enter the appropriate word pairs in DATA.

The DATA, beginning on line 1000 is easy to expand as your language skill increases and can be used in conjunction with either a self teaching or school language course. Just enter the DATA in word pairs, the first in English, the second in whatever language you are working with.

The key to the vocabulary drills is the random word subroutine, lines 500 through 550. In line 510 X=(1+INT(RND(1)*25)) the 25 is equal to the number of word pairs entered as DATA. As you increase the number of word pairs by adding new DATA this number should also be appropriately increased. You can also alter this line to drill yourself on only part of your total foreign language vocabulary.

Let's assume, for example, that you have 600 word pairs in DATA but only want to drill yourself on the last 100 words you have entered. In that case, change line 510 to X=(500+INT(RND(1)*100)).

Lines 160,180,260,280,330 and 430 use the ATARI cursor advance and line "up" arrow keys to provide a format that will display as much of your language drill or translations on the screen at one time as possible, but you can replace these with just "PRINT" statements if you prefer. This would be useful if you want to use the program for drill and translation of whole phrases and sentences instead of just single words. Don't forget, however, to increase the size of the appropriate string dimensions (ENGLISH$,WORD$,TRANSLATE$) in line 50.

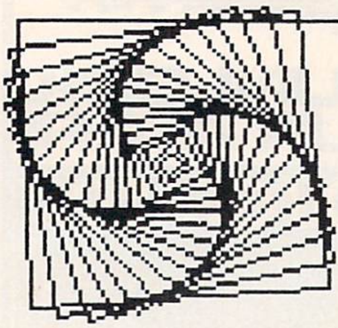
One final note; if you happen to own IRIDIS 2 (and if you have an ATARI computer I don't think you can find a better bargain in software) you can easily add the appropriate subroutine to use Language Lab for Russian, Greek, Hebrew or whatever you wish by adding the foreign alphabet in lower case. I am currently using the program to teach myself Classical Greek. I hope this program will be useful for budding language students and in any case good luck with it, bon chance, and auf wedersehen.

...basically a computerized version of that old standby of language education, the flash card set.

```
0 DIM A$(20),B$(20),C$(20),U$(44),X$(20)
,Y$(1),P(16):GOTO 2000
1 DATA 1,IM____,NOT,NOT
2 DATA 2,MIS____,WRONG,WRONG
3 DATA 3,CON____,WITH,TOGETHER
4 DATA 4,SUB____,UNDER,UNDER
5 DATA 5,SUPER____,OVER,OVER
6 DATA 6,PRE____,BEFORE,BEFORE
7 DATA 7,INTER____,BETWEEN,AMONG
8 DATA 8,TRANS____,ACROSS,ACROSS
9 DATA 9,EX____,OUT OF,FROM
10 DATA 10,EX____,OUT OF,FROM
110 OPEN #1,4,0,"E:"
115 Q=0:M=0:POKE 752,1:?
120 U$="____________________________________
______________"
125 TRAP 150
130 READ P,A$,B$,C$
140 GOTO 130
150 TOTAL=P
160 X=INT(TOTAL*RND(0)+1)
170 RESTORE X
180 READ P,A$,B$,C$
190 B1=LEN(B$):C1=LEN(B$)+LEN(C$)+4
200 ? "          HOMEWORK PRACTICE"
205 ? "          _________________"
210 ? :? :? :? :? :POKE 752,0
220 POKE 85,17-LEN(A$)/2:? A$;" ? ";
230 INPUT #1;X$
240 POKE 752,1:? :?
```

```
250 IF X$=B$ OR X$=C$ THEN GOSUB 1000:GO
TO 500
260 ? "WRONG......TRY AGAIN."
270 M=M+1:Q=Q+1:P(Q)=P:IF Q>15 THEN ? CH
R$(125):GOTO 600
280 ? :? :POKE 752,0
290 POKE 85,17-(LEN(A$)/2):? A$;" ? ";
300 INPUT #1;X$
310 POKE 752,1
320 IF X$=B$ OR X$=C$ THEN GOSUB 1000:GO
TO 500
330 ? :? :? "NOPE........THE CORRECT A
NSWER IS":? :?
340 IF B$<>C$ THEN POKE 85,19-C1/2:? B$;
" or ";C$:GOTO 360
350 GOTO 370
360 POKE 85,19-C1/2:? U$(1,C1):GOTO 500
370 IF B$=C$ THEN POKE 85,19-B1/2:? B$
380 POKE 85,19-B1/2:? U$(1,B1)
500 ? :? :? :?
510 ? "  PRESS RETURN FOR ANOTHER PROBLE
M":? "  OR TYPE L FOR A LIST OF MISTAKES
 ";
520 INPUT #1;Y$
530 IF Y$<>"" AND Y$<>"L" THEN 510
540 ? CHR$(125)
550 IF Y$="L" THEN 600
560 GOTO 160
600 SETCOLOR 1,12,4:SETCOLOR 2,12,0
610 ? "          LIST OF MISTAKES"
615 ? "
620 IF M<1 THEN 700
630 FOR R=1 TO Q
650 RESTORE P(R)
660 READ P,A$,B$,C$
670 PRINT A$;
680 IF B$=C$ THEN ?".....";B$
685 IF B$<>C$ THEN ? "...";B$;" or ";C$
690 NEXT R
700 PRINT
710 ? "DO YOU WANT MORE PRACTICE ";:INPU
T Y$
720 IF Y$="Y" THEN FOR R=1 TO 16:P(R)=0:
NEXT R:M=0:Q=0:GRAPHICS 0:POKE 752,1:GOT
O 160
730 ? :? "O.K., GOODBYE FOR NOW."
740 ? :? "IF YOU ADDED NEW PROBLEMS THIS
":? "TIME, BE SURE TO CSAVE THIS PROGRAM
."
750 FOR TIME=1 TO 1400:NEXT TIME
997 GRAPHICS 0:POKE 752,0
998 CLOSE #1
999 END
1000 ? :? :?
1010 FOR FLASH=1 TO 7
1020 POKE 85,15:? "CORRECT !";
1030 FOR TIME=1 TO 10:NEXT TIME
1040 POKE 85,15:? "CORRECT !";
1050 FOR TIME=1 TO 10:NEXT TIME
1060 NEXT FLASH
1070 POKE 85,15:? "CORRECT !"
1080 RETURN
2000 ? CHR$(125)
2005 POSITION 10,2:? " HOMEWORK PRACTICE
 "
2010 POSITION 8,10:? "DO YOU WANT TO ENT
ER":POSITION 8,11:? "NEW HOMEWORK PROBLE
MS ";:INPUT Y$
2015 IF Y$<>"Y" AND Y$<>"N" THEN 2010
2020 IF Y$="N" THEN 110
2030 POKE 752,1:? CHR$(125):POSITION 3,1
1:? "PLEASE WAIT WHILE I GET READY....":
FOR TIME=1 TO 250:NEXT TIME
2040 SETCOLOR 1,9,4:POKE 752,0
2050 TRAP 2080
2060 READ P,A$,B$,C$
2070 GOTO 2060
2080 IF P>100 THEN GRAPHICS 0:? :? "CHEC
K DATA STATEMENTS !!":? :? "LINE 100 IS
HIGHEST ALLOWED FOR DATA.":STOP
2085 FOR ERASE=1 TO P
2090 SOUND 0,4*ERASE*RND(1),12,8
2100 ? CHR$(125)
2110 ? "↓";ERASE
2120 ? :? :? "CONT"
2130 POSITION 0,0
2140 POKE 842,13:STOP
2150 POKE 842,12
2160 NEXT ERASE
2200 SOUND 0,0,0,0:? CHR$(125):SETCOLOR
1,9,10
2210 ? :? :? "O.K., I'M READY FOR YOUR"
2211 ? "NEW HOMEWORK PROBLEMS."
2220 ? :? :? "IGNORE ";CHR$(34);"STOPPED
 AT LINE 2270";CHR$(34);"."
2230 ? :? "....Use this format for enter
ing....."
2240 ? "24 DATA 24,QUESTION,ANSWER 1,ANS
WER 2"
2250 ? :? "TYPE CONT WHEN YOU HAVE FINIS
HED."
2260 ? :? :? "TYPE IN YOUR NEW PROBLEMS
NOW....":? CHR$(253)
2270 STOP
2300 ? CHR$(125):POKE 752,1:POSITION 4,1
0:? "O.K., I'M READY TO TEST YOU ON"
2310 POSITION 4,11:? "YOUR NEW HOMEWORK
PROBLEMS...."
2320 FOR TIME=1 TO 300:NEXT TIME:POKE 75
2,0
2330 RESTORE :GOTO 110
```

©



Friends Of The Turtle

David D. Thornburg
Los Altos, CA

Your letters keep pouring in, and I am writing answers as fast as possible. Many of you have asked about the availability of Apple LOGO. There are rumored to be several versions of this language which either are, or will be, available from several vendors (it is risky writing columns a month or so in advance in a field as active as this one). The version available from Apple is a product of Logo Computer Systems Inc. I have looked at it and like it very much. I also saw two draft manuals – one reference manual and one introductory manual which introduces LOGO through the use of the turtle. Readers with 64K Apples will be most impressed with this language.

COMPUTE! reader Thomas Granvold wrote to tell me that he and his wife use Atari WSFN's turtle graphics to help them in quilt design. His wife selected the pattern resulting from the following procedure for use in a quilt pattern:

HCN8 (R8(R15F))

This procedure draws eight octagons around a common point. These will then be put in a square which circumscribes the octagon to form the basis for the quilt pattern. For those of you who don't understand the alphabet soup of WSFN, the above procedure translates as follows: "Home, Clear, point North, repeat 8 times the task of turning right by 45 degrees and 8 times turning right by 45 degrees and drawing a line 15 units long." In WSFN the R command turns the turtle to the right by 45 degrees and the F command moves the turtle forward by one unit.

Randolph Schleef of the Miami-Dade Community College (11380 NW 27 Ave., Miami, FL 33167) is particularly interested in the use of turtle geometry with disabled users. Any of you who have information concerning applications in this area should contact him directly.

Is This Trip Really Necessary?

This winter I had the opportunity to teach computer programming to children at a local school. Each week I spent one-half hour in each class from second through sixth grade. The principal teaching tool was (of course) the turtle – both in the form of Atari PILOT and the Milton Bradley Big Trak. During these classes I was delighted to see that children have a pretty accurate intuitive sense when it comes to geometry. One of my favorite geometrical rules is called the Total Trip Theorem. This theorem states that if you send the turtle off to trace out a simple closed path, that by the time the turtle has returned to its original location and orientation it will have turned by exactly 360 degrees.

A few examples should convince you that this is a plausible idea. Suppose we first have the turtle trace out a square. In Apple LOGO, we would type:

REPEAT 4 [FORWARD 30 RIGHT 90]

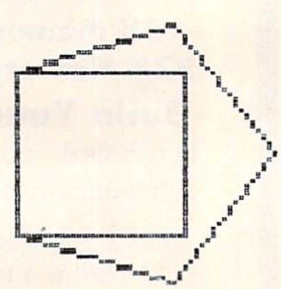
to draw a square 30 units on a side.

```
REPEAT 4 (FORWARD 30 RIGHT 90)
```

Figure 1.

Notice that in drawing this square, the turtle turned 90 degrees four times, or 360 degrees overall. To draw a pentagonal path with the same length sides, we would type:

REPEAT 5 [FORWARD 30 RIGHT 72]



```
REPEAT 4 (FORWARD 30 RIGHT 90)
REPEAT 5 (FORWARD 30 RIGHT 72)
```

Figure 2.

COLLEGE BOARD SAT* PREP SERIES

TRS-80, APPLE, PET, OSI, ATARI, CP/M, PDP-11

Each program confronts the user with a virtually limitless series of questions and answers. Each is based on past exams and presents material on the same level of difficulty and in the same form used in the S.A.T. Scoring is provided in accordance with the formula used by College Boards.

S.A.T., P.S.A.T., N.M.S.Q.T. — Educator Edition set includes 25 programs covering Vocabulary, Word Relationships, Reading Comprehension, Sentence Completion, and Mathematics. **Independent tests of S.A.T. series performance show a mean total increase of 70 points in students' scores.** Price **$229.95**

GRADUATE RECORD EXAM Series - Educator Edition includes 29 programs covering Vocabulary, Word Relationships, Reading Comprehension, Sentence Completion, Mathematics, Analytical Reasoning and Logical Diagrams. Price **$289.95**

COMPETENCY PROFICIENCY EXAM PREP SERIES

This comprehensive set of programs consists of simulated exam modules, a thorough diagnostic package, and a complete set of instructional programs. It is designed to teach concepts and operations, provide drill and practice and assess achievement levels through pre and post testing. The Competency Exam Preparation Series provides a structured, sequential, curriculum encompassing mathematical, reading and writing instruction.

This program is designed for individual student use or use in a classroom setting. Programs provide optional printer capability covering worksheet generation and performance monitoring. C.E.P.S. are available in three software formats.

Special editions available for California Proficiency Assesment Test and New York Regents Competency Tests. Call for Prices.

Krell's M.I.T. Logo for Apple* $179.95

Complete 4 disk set includes Library Disk, 20 program Tutorial Disk, **Alice In Logoland** and **2 COPIES OF M.I.T. LOGO FOR APPLE***

Odyssey In Time

This spectacular adventure game adds a new dimension of excitement and complexity to **Time Traveler.**

Odyssey In Time includes all the challenges of **Time Traveler** plus 10 additional eras. Each game is different and may be interrupted and saved at any point for later play. **$39.95**

Time Traveler

The best of the adventure games. Confronts the player with complex decision situations and the demand for real time action. Using the Time Machine, players face a challenging series of historical environments. To succeed you must build alliances and struggle with the ruling power. Each game is unique. **$24.95**

Isaac Newton +

Perhaps the most fascinating and valuable educational game ever devised - **ISAAC NEWTON** challenges the players (1-4) to assemble evidence and discern the underlying "Laws of Nature" that have produced this evidence. **ISAAC NEWTON** is an inductive game that allows players to intervene actively by proposing experiments to determine if new data conform to the "Laws of Nature" in question. Players may set the level of difficulty from simple to fiendishly complex.

F.G. Newton

Full Graphics Newton. This version of Isaac Newton presents all data in graphic form. Because data is graphic rather than symbolic, this game is suitable for very young children. Players may select difficulty levels challenging to the most skilled adults. **$49.95**

☆ NEW ☆ ☆MICRO-DEUTSCH☆

Micro-Deutsch set includes 24 grammar lessons, covering all material of an introductory German course. Four test units also included. Grammar lessons use substitution transformation drills, item ordering, translations and verb drills. Drill vocabulary based on frequency lists. Suitable for use with any high school or college textbook. Extensively field tested at SUNY Stony Brook. Available for Apple II and PET/CBM. (PET version includes a special foreign language character chip.) Also available soon: MICRO-FRANCAIS, MICRO-ESPANOL, MICRO-IVRIT, MICRO-YIDDISH, MICRO-CHINESE, MICRO-JAPANESE. **$179.95**

☆ NEW ☆ ☆Pythagoras and The Dragon☆

Mathematics in a fantasy game context. Based on **The Sword of Zedek, Pythagoras and The Dragon** introduces Pythagoras as a mentor to the player. When called on for aid, Pythagoras poses math questions, and depending on the speed and accuracy of the player response, confers secret information. With Pythagoras as an ally, the quest to overthrow Ra, The Master of Evil, assumes a new dimension of complexity. Depending on the level chosen, problems range from arithmetic through plane geometry. 32K **$39.95**

Free Bonus with purchase of $300.00 or more: Applesoft Tutor Series

Krell Software Corp. has no official ties with the College Entrance Examination Board or the Educational Testing Service. Krell is, however, a supplier of products to the E.T.S.

PROGRAMS AVAILABLE FOR TRS-80, APPLE II, PET & ATARI

N.Y.S. residents add sales tax.

All programs require 16K • TRS-80 programs require LEVEL II BASIC • APPLE programs require Applesoft BASIC

*Trademark Apple Computer Corp.

KRELL SOFTWARE CORP.

"The State of the Art in Educational Computing"

21 Millbrook Drive, Stony Brook, N.Y. 11790 (516) 751-5139

For this figure, the turtle turned 72 degrees five times, or 360 degrees overall. Finally, let's look at a hexagonal path:

REPEAT 6 [FORWARD 30 RIGHT 60]

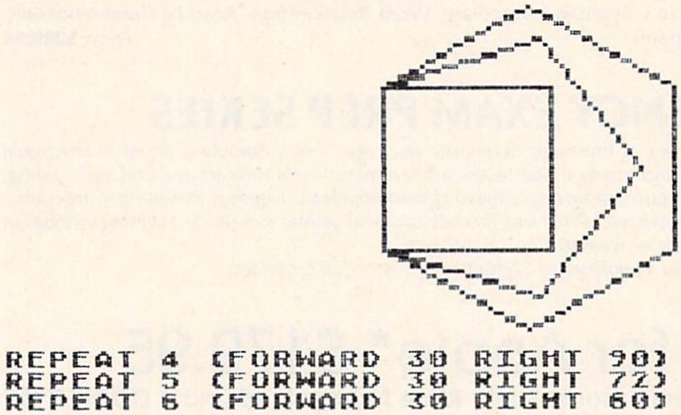


Figure 3.

It should come as no surprise to find that the total amount turned is once again 360 degrees.

Abelson and diSessa cover this topic quite thoroughly in their book *Turtle Geometry;* and, as I said before, grade school children seem to have an intuitive feel for this result.

But if this theorem works well for a turtle walking on a flat surface, how does it work for a turtle walking on another surface – say that of a cube?

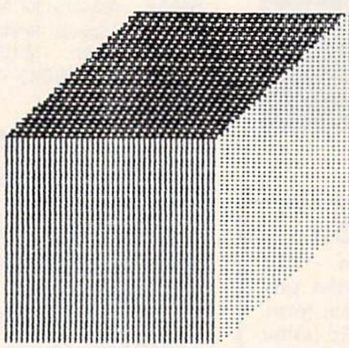


Figure 4.

Suppose the turtle starts off at the center of the front face and is pointing up. We could have the turtle walk forward to the center of the top face, being careful to walk straight over the edge without turning.

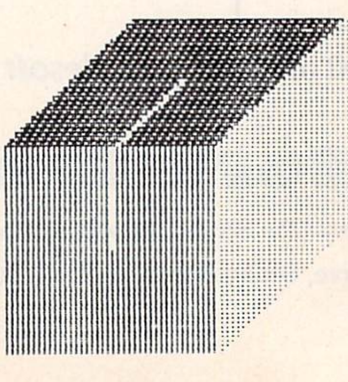


Figure 5.

Next, we can have the turtle turn to the right by 90 degrees and walk in a straight line to the center of the right face.

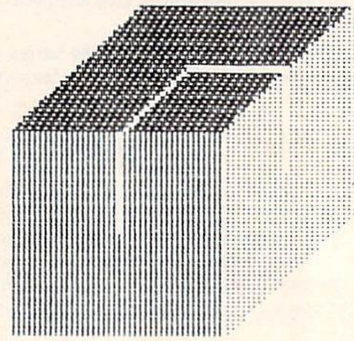


Figure 6.

Once the turtle has arrived there, the turtle once again turns to the right by 90 degrees and walks in a straight line to the center of the front face. When it turns right by 90 degrees, it is then back at its starting location and direction.

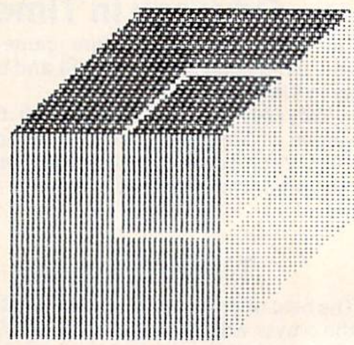


Figure 7.

If you have been keeping up with the number of turns, you have probably noticed that our turtle has made a nice closed path but has only turned 270 degrees. What happened to the missing 90 degrees?

To see what happened, we only need to spread part of the cube out into a flat surface (since we know what turtles do on flat surfaces). If we fold the front and right face up to be on the same surface as the top of the cube we see this picture:

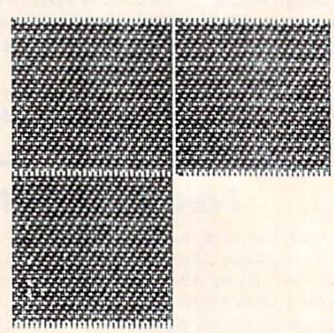


Figure 8.

Now let's have the turtle trace its path once again. Seeing what happens with the first few steps is pretty simple. The turtle moves forward and turns right twice.

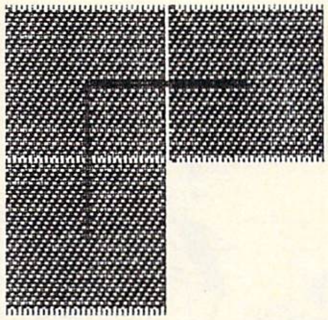


Figure 9.

Next, the turtle has to get back to its starting position. Since the cube was opened flat, the edges of the front and right side which normally touch are now spread apart. As you can see, they are spread

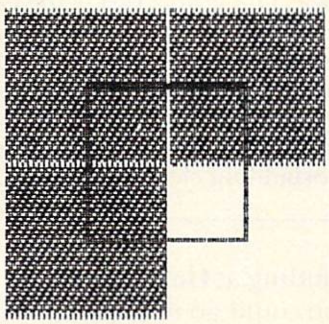


Figure 10.

apart by 90 degrees. So if we want to complete the closed path, we have to turn the turtle the extra 90 degrees to make it connect with its starting point. This shows that, if the turtle isn't walking on a plane surface, the total turning angle for a closed path may not be 360 degrees. If the angle is some other value, then the difference between this value and 360 degrees is the size of the angular "gap" that would be created by spreading the curved surface flat.

I hope you find little excursions like this to be interesting. One of the nice features of turtle geometry is its ability to make some difficult mathematical concepts easy to see.

If you know any other of these types of illustrations (or would like me to find some more of my own) let me know what you have or want, and I will share the results with you all.

Once again, please keep me posted of your activities and interests with turtles so that I can share them with your fellow Friends of the Turtle.

Friends of the Turtle
P.O. Box 1317
Los Altos, CA 94022

©

The World Inside The Computer

Last month **COMPUTE!** welcomed Fred D'Ignazio, whose The World Inside the Computer column will appear each month.

Fred is a computer enthusiast and author of several books on computers for young people. He is presently working on two major projects: he is writing a series of books on how to create graphics-and-sound adventure games. He is also working on a computer mystery-and-adventure series for young people.

As the father of two young children, Fred has become concerned with introducing the computer to children as a wonderful tool rather than as a forbidding electronic device.

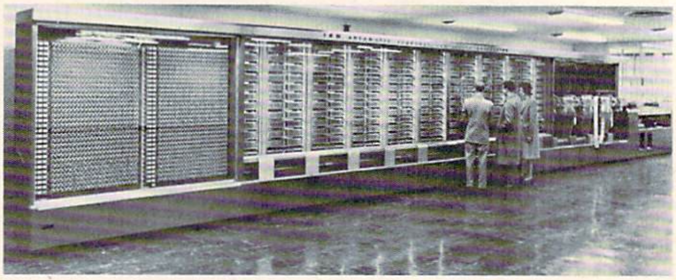
Down The Rabbit Hole

> ... this bottle was *not* marked "poison," so Alice ventured to taste it, and, finding it very nice (it had, in fact a sort of mixed flavour of cherry-tart, custard, pine-apple, roast turkey, toffy, and hot buttered toast), she very soon finished it off.
>
> "What a curious feeling!" said Alice, "I must be shutting up like a telescope!"
>
> And so it was indeed: she was now only ten inches high, and her face brightened up at the thought that she was now the right size for going through the little door into that lovely garden.
>
> Lewis Carroll
> *Alice's Adventures in Wonderland*

From Pickles to Sugar Wafers

Once upon a time, less than a lifetime ago, computers were very, very large. But they are dwindling fast. Already they are very, very small. Soon they will be even smaller.

The Mark I, the world's first general-purpose digital computer, was built in the early 1940's. The Mark I filled an entire wall inside a red brick physics building at Harvard. When the Mark I was working, one could go in and listen to its horde of electromechanical relays gently clicking, like a roomful of ladies knitting.



The Harvard Mark I in the early 1940s: The computer 'world' filled an entire wall. Credit: *Courtesy of IBM.*

Then came the world's first electronic computer, the ENIAC. The ENIAC had 20,000 vacuum tubes – hot, glowing "pickles" that acted as switches and amplifiers inside the computer's brain. Each time you turned the ENIAC on, two of its "pickles" would explode, just from the surge of current.

In 1947, the transistor was invented. Instantly, each logic and memory cell inside the computer shrank to the size of a tootsie roll. Each transistor was individually packaged with three protruding wires, representing input, output, and control voltages.

Even with the advent of the transistor, most computers in the late 1940's and early 1950's were chips...chips...chips...chips...chips...chips...

Limited Time: FREE Disk-O-Pro.

Special: You can get this Disk-O-Pro **absolutely free...** when you order the new Commodore 2031 single disk drive from Skyles. (See page 8.)

Command-O or Command-O-Pro?

It's called the Command-O-Pro in Europe, Command-O in the U.S.

But whatever you call it, this 4K byte ROM will provide your CBM BASIC 4.0 (4016, 4032) and 8032 computers with 20 additional commands including 10 Toolkit program editing and debugging commands and 10 additional commands for screening, formatting and disk file manipulating. *(And our technical writer dug up 39 additional commands in the course of doing a 76-page manual!)*

Extends Commodore's 8032 advanced screen editing features to the ultimate. **SCROLL** up and down, insert or delete entire lines, delete the characters to the left or right of the cursor, select **TEXT** or **GRAPHICS** mode or ring the PET's bell. Redefine the window to adjust it by size and position on your screen. And define any key to equal a sequence of up to 90 key strokes.

Resides in hexadecimal address $9000, the rightmost empty socket in 4016 and 4032 or the rearmost in 8032. If space conflicts, Socket-2-ME available at a very special price.

Command-O from Skyles Electric Works $75.00
Complete with **Socket-2-ME** 95.00

Disk-O-Pro ROM:

25 new commands at $75.00, that's only $3.00 a command!

The powerful chip for all "classic" PETs. In any PET with Version III (BASIC 2.0) ROMs (**### COMMODORE BASIC ###**) Disk-O-Pro will give 19 software compatible disk instructions*: 15 identical with the new BASIC 4.0 (or with 8032 ROMs) compatible with both old and new DOS. Plus 4 additional disk commands...including program appending (**MERGE**), program overlaying (**MERGE #**) and **PRINT USING**, allowing formatting output of strings and numbers on the PET screen or on any printer.

*** NOTE:** *Old DOS doesn't recognize 3 commands.*

Plus softtouch key (**SET**) which allows you to define a key to equal a sequence of up to 80 keystrokes; **SCROLL** whereby all keys repeat as well as slow scrolling and extra editing features; **BEEP** which allows you to play music on your PET.

Completely compatible with the BASIC Programmer's Toolkit. *(The chip resides in the socket at hexadecimal address $9000, the rightmost empty socket in most PETs.)* And for the owners of "classic" (or old) PETs, we do have interface boards.

(For those owning a BASIC 4.0 or 8032, even though the Disk-O-Pro may not be suitable, the Command-O is. We have never abandoned a PET owner.)

Complete with 84-page manual written by Greg Yob.

Disk-O-Pro from Skyles Electric Works $75.00
Complete with **Interface Board** (for "Classic" PETs) 95.00

"The PET with the new PicChip...is like home movies."

The new ROM that took Europe by storm is available only from Skyles Electric Works in the U.S. and Canada.

PicChip, a ROM extension of the BASIC Version III (BASIC 2), BASIC 4.0 or BASIC 8032 interpreter. Over 40 commands that allow you to create programs with dynamic graphics displays: plots, bar graphs, pictures; and rolling, scrolling, shifting and inverting. All instantly and easily added to your BASIC program.

The address for the 2000/3000 (which would require PicChip module PC2), for the 4000 (PC4) and for the 8000 (PC8) is $A000 ...unless you have a Mikro, Word Pro 3 or 4, or Jinsam, which occupy that same address. In those cases, you will need the PicChip on an interface board... for the 2000/3000 series, PCB2, for the 4000, PCA4, and for the 8000, PCA8. In all cases, the Mikro or WordPro would be switchable manually using the Skyles Socket-2-ME...available at a very special price.

PicChip from Skyles Electric Works $60.00
(Please indicate PC2, PC4, PC8)
Complete with Socket-2-ME 80.00
(Please indicate PCB2, PCA4, PCA8)

The New Mikro:

Now you can program in machine language as easily as writing BASIC.

4K machine language assembler ROM plugs into your main board. Does all the machine language work for you; all you have to do is start laying down the code.

Retains all the great screen editing features of the PET... even all the Command-O or Toolkit commands. Write your own machine language subroutine. The program you write is the source code you can save. Machine language monitor saves the object code. Not as professional as the Skyles MacroTeA...not as expensive, either.

A great learning experience for those new to machine language programming but who want to master it easily. Superb 50-page manual—including brief introduction to machine language programming—included. But we also recommend several of the books in the Skyles Book Barn.

Skyles Mikro Machine Language Assembler
MK 2/3 for 2000 or 3000 Series PETs $80.00
MK 4/8 for 4000 or 8000 Series PETs and CBMs 95.00
Complete with **Socket-2-ME**

Skyles Catalogue Page 1

For PET and CBM owners only:

This is just 1 of 20 pages of the newest and biggest Skyles catalog, hot off the press.

We know you'll want this page, in its full 8½ x 10 splendor, and another 19 pages of peripherals, software and books that will make your PET or CBM computer even nicer to live with. So, if we missed sending you your very own copy within the last few weeks, call us at **(800) 227-9998 (unless you live in California, in which case call (415) 965-1735.**

From Skyles Electric Works, the oldest and largest professional specialists in the business.

Skyles Electric Works
231-E S. Whisman Road
Mountain View, California 94041

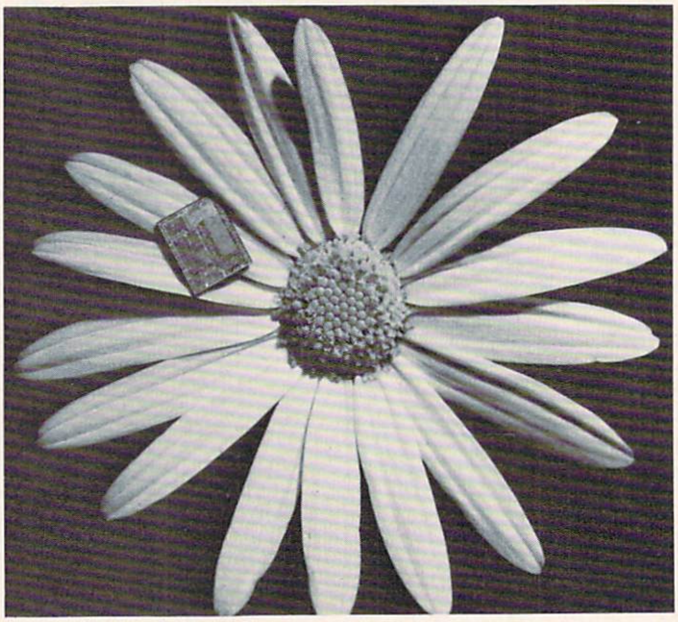
enormous vacuum-powered "dinosaurs," corralled inside warehouse-sized research laboratories. The Whirlwind computer at MIT, for example, was the grandaddy of the minicommputer. Yet it consumed 150 kilowatts of electricity and filled a two-story building.

In 1959, scientists at Texas Instruments and Fairchild Semiconductor uncovered a trapdoor to smallness. They found a way to flatten a barrel-shaped transistor into a paper-thin "sugar wafer" – the integrated circuit.

The scientists created each circuit – each transistor by photographing its circuits, reducing the photo and "printing" the circuits on a small slice of silicon using a gourmet recipe of chemicals and light.

During the 1960's, scientists found ways to etch transistor circuits into smaller and smaller plots of silicon real estate. As a result, more and more transistors, like sardines in a can, were squeezed into fingernail-sized integrated circuits. By the late 1960's, engineers were able to fit 10,000 logic elements, or "gates," on a sliver of silicon only a quarter of an inch square.

Then, in 1969, a major breakthrough occurred. Ted Hoff and a team of engineers at Intel Corporation built a microprocessor, a tiny computer "brain" on a single chip.



Computer on a Daisy: The Intel 8748 Microcomputer. Credit: *Courtesy of Intel Corporation.*

During the 1970's, miniaturization proceeded at a rapid pace. By the mid-1970's, whole computers were constructed on chips.

Now it's the early 1980's. Scientists at Bell Labs are fabricating computer brain chips with a million components. In Japan, circuit designers are building memory chips capable of storing a quarter of a million bits of information. Million-bit memory chips are forecast by the late 1980's or early 1990's.

As these new chips emerge from the labs, they head like homing pigeons to people's houses and alight inside their personal computers. Three-year-olds are now playing with "typewriters" that can outthink million-dollar electronic brains of just a decade ago.

The Candle Flame

Like a telescope shutting up – like Alice in Wonderland – computers are shrinking. The question is, how far can they go? How small can they get?

As Alice got smaller and smaller, she began to worry:

> ... "for it might end, you know," Alice said to herself, "in my going out altogether, like a candle. I wonder what I should be like then?" And she tried to fancy what the flame of a candle looks like after the candle is blown out, for she could not remember ever having seen such a thing.

New, ultra-miniature computers are created by circuit designers. Circuit designers are like surgeons. Using a scalpel made of light, they etch circuits, like fine incisions, into the flakes and wafers of silicon.

But the silicon surgeons' light-wave scalpel has become a hefty sculptor's chisel. It is too big, too clumsy, and too blunt to slice the tiny circuits required to shrink computers even smaller. As a result, scalpels made of light waves are being replaced by scalpels made of X-rays and scalpels made of electrons. X-rays and electron beams' smaller wavelengths give them a super-thin, razor-sharp cutting edge.

Using these new tools, designers have etched channels in silicon only 100 billionths of a meter wide. A mere two hundred silicon atoms, strung like pearls on a necklace, would bridge one of these tiny trenches.

How small can computers get? How narrow can circuit channels become?

One of the designers' latest scalpels – the X-ray – has a wavelength equal to the diameter of a single atom of silicon. So circuit channels can get even smaller: from 200 atoms wide, to 100, then to 50, 20, then 10.

Alice wondered what a candle flame looked like after the candle was snuffed out. Similarly, what does a circuit "pipe" look like when its surface is only twenty atoms thick, or ten atoms, or only five? Is the surface still solid? Or is it something else?

The world of the super-small computer is stranger than anything Alice ever saw in Wonder-

land. It is a world where dozens of entire circuits – an entire chip – could be laid like tiles on the surface of a single red blood cell. It is a world where the radioactivity of common objects – of tables, walls, your fingertips – is a threat.

It is a world where electrons leak from transistors, a world of "soft" switches and "fuzzy" memory cells. It is a world where quantum mechanics replaces common-sense cause and effect. At this level, computer circuits' polished edges become rough and grainy. It is a world of uncertainty and surprise.

Yet the computer may shrink even further.

Scientists at Cornell University, Bell Laboratories, and elsewhere have begun research into "breeding" organic molecules that could be used as memory and logic components inside a future computer. A single protein molecule – say, inside the hemoglobin in a red blood cell – could act as a logic switch or carry a bit of information.

How?

Scientists have observed that organic molecules change their shape slightly in the presence of an electric charge. Thus, an egg-shaped molecule might represent a "1" bit and a golfball-shaped molecule a "0" bit.

How small is this new computer world? Molecular computers would be a thousand times smaller than computers etched with X-ray beams.

Is this the end? Perhaps. Yet if we can have molecular computers, why not atomic computers? If atomic computers, why not computers made from quarks?

The Frozen Baseball

We have descended into the nether world of fundamental particles. Now we need to return to our super large, everyday world. To do that, let's follow Alice's lead. After Alice became small, she found a tiny glass box underneath a table in the rabbit-hole. Inside the box was a very little cake, with the words EAT ME marked in currants. Alice ate the cake, and like a telescope opening up, she quickly grew larger.

Let's eat the cake, too, and grow larger.

As we do, we swiftly leave behind the world of the very small, and we return to an "everyday" world where scientists are starting to design computers out of the microscopic building blocks we have just examined.

What sort of computers are they designing?

The latest "race horse" of the microminiature world is the Josephson Junction, a switching device that achieves incredible speeds since it is supercooled to a temperature near Absolute Zero (-460 degrees Fahrenheit, or -273 degrees Celsius).

Scientists at the University of California predict that, in the near future, we will be able to build a Josephson Junction computer. It will come equipped with tiny refrigerators called micro-coolers. It will be only slightly bigger than a baseball. Yet it will fly! Operating on a lightbulb's power, it will process information faster than the largest computer now available.

We won't see a Josephson Junction personal computer for awhile. Yet other types of frozen computers are not tucked away in the far-off future. The Japanese, for example, are spending $400 million to build a supercomputer for the 1990's. The computer will use Artificial Intelligence techniques built into tiny, frozen circuits and be the first of a new generation of true "thinking machines."

By 1985, another frozen computer, the Cray-2, will be in operation. Computer genius, Seymour Cray, has submerged entire cards of microminiature circuits in a bathtub filled with a super-cooled liquid. The densely packed circuits never overheat, and the electric pulses move at incredible speeds, six to twelve times faster than one of today's fastest computers, the Cray-1.

Elsewhere, scientists at MIT and Carnegie-Mellon are constructing a radically new computer with the motto: divide and conquer! The computer will consist of a network of hundreds of microprocessor chips that would split up a complex problem into tiny, simpler parts. Each mini-problem would be attacked simultaneously by a separate chip, much the way our brain carves up problems and feeds them to the high-level neurons – the tiny processors inside our brains that solve problems and manage the huge flood of information pouring in from our senses.

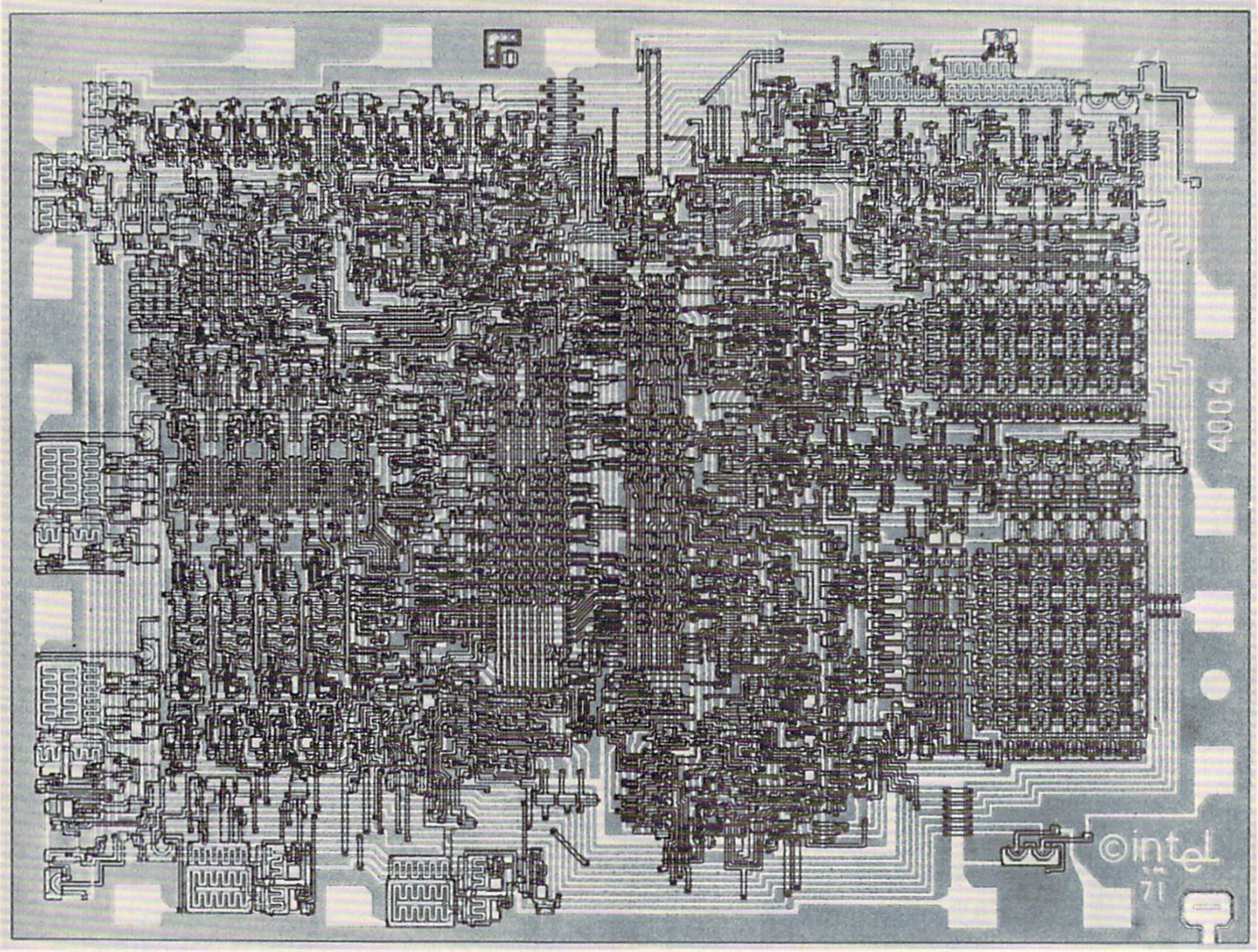
These new fishnet computers – dubbed "omni-computers" and "ultracomputers" – may eventually evolve into 3-D computers. That is, today's flat, two-dimensional computer chips and wafers will be replaced by computer cubes, boxes, and baseballs. In a 3-D computer, information will no longer enter single-file in snakelike strings. Instead it will be shovelled in in great gobs. Likewise, it won't be processed sequentially or emerge sequentially. Instead, it will emerge as two-dimensional "slices" of the cube – as gigantic "burps" of trillions of bits.

Street Map of the US

According to one expert, by 1990, circuit designers will be able to fabricate a computer chip with a level of detail equal to a street map of the entire United States.

And therein is the metaphor, the focal point of this column: the street map, the city, the world inside the computer.

Take a photo of a chip and blow it up large,



The Intel 4004: The First Microprocessor on a Chip. Credit: *Courtesy of Intel Corporation.*

really large. What do you see?

I'll tell you what *I* see. I see a tiny world. I feel like Jules Verne floating in a hot-air balloon over Manhattan. I have a bird's-eye view of entire city blocks, of monuments, trees, elliptical promenades. I see parks, cathedrals, railroad tracks, warehouses, and bus stops. I see more than a city. I see an entire world.

Until recently, these tiny, chip-sized worlds – these Oz-like "Emerald Cities" – were nearly all alike. They were all optical echoes of some master image created by computer designers cloistered in some futuristic semiconductor laboratory.

Microscopic trenches and overpasses formed tiny transistors. The transistors were grouped into logic *gates* to help the computer make decisions and process information, or into memory cells, such as capacitors, where information, a bit at a time, could be safely filed away. The gates and memory cells, in turn, formed orderly ranks, like soldiers on parade. En masse, they became a computer's ALU (Arithmetic-Logic Unit), CPU (Central Processing Unit), or its RAM (Random Access Memory) or ROM (Read Only Memory).

Through The Electronic Looking Glass

A couple years ago, I wrote a picturebook, called *Katie and the Computer* (Creative Computing Press, 1979). The story is about a little girl named Katie. Katie is a modern Alice – an "Alice in Computerland."

In the story, Katie's father brought home a personal computer. Katie climbed atop a tall stool. She began typing the word *FLOWER* on the keyboard to get a program to display a bright yellow daffodil on the picture screen. As she searched for the letter *R*, she leaned too far forward, lost her balance, and toppled into the picture screen:

> But instead of bumping her nose on the glass, she went right through it and began spinning and falling, just as if she'd tumbled off the top of a tall mountain.
>
> Inside the computer it was snowing. As

Katie fell, a snowflake as big as a house fluttered past her. "Wow!" she thought. "I'm really getting tiny!"

Katie landed in a feathery bank of snow. Up rushed a "Colonel," who represented the computer's control program, or operating system. The Colonel took Katie on a tour of Cybernia, the world inside the computer, and showed Katie how the computer processed her program and "painted a flower" on the picture screen.

Katie visited all the major tourist spots in the world of the silicon chip. ROM was the chilly, mountainous region where the Colonel lived. The CPU was a huge, bustling train station. RAM was a huge hotel, so tall it "reached into the clouds." There was even a bug – a ferocious robot spider that loved to gobble yellow airplanes.

The point is, the world Katie visited was a familiar world, repeated inside of billions of computer chips. Cybernia's major landmarks – RAM, ROM, the CPU – were duplicated endlessly. If you had, like Katie, visited one world inside a chip, you had seen them all.

Yet this is no longer true.

Scientists can squeeze so many circuits on a chip, why stick just to the traditional landmarks like the CPU, or ROM, or RAM?

Consequently, radically new chips are being designed. The new chips create whole new silicon cities and worlds. There are now chips with rings of processors, circling around a control processor like planets around the sun. There are chips with dozens, hundreds of tiny CPU-RAM pairs. There are chips with entirely new kinds of processors and memories.

In 1981, only a fifth of the $12 billion in chips sold were custom-made and non-standard. Now experts predict that by 1985, more than half of all chips will be custom-made.

Using *gate arrays* and other new building blocks, silicon building contractors will erect entire new cities, entire new systems, all on a plot of matter only a quarter of an inch square.

These new cities will be just as complex, just as diverse as human cities, from Anchorage to Rawalpindi, from Rio de Janeiro to Baghdad, from Melbourne to Novosibirsk.

The Neighborhoods and Boroughs of Silicon City

Many of the new chips will be special purpose "engines," each an expert or specialist at some task.

The chips will perform a myraid of functions, but, the essence of their job will be to reproduce – to mimic – a real-world event or process. The chips will become increasingly good at creating miniature electronic copies of our regular, natural world and everything in it.

Music-engine chips will make music. Machine-vision chips will "see." Speech-understanding chips will "hear." Graphics chips will produce dazzling pictures and animated cartoons and games. Speech-synthesis chips will talk like people. Expert-system chips will advise scientists and help doctors perform diagnoses.

Meanwhile the circuits inside the chips will continue to shrink. The time is not far off when the many specialist chips will shrink and become specialist "neighborhoods," coexisting inside a single "metropolis" chip.

Imagine a chip with neighborhoods and districts, just like in New York City. Each neighborhood would consist of a maze of microminiature circuits – specialists in music, machine vision, or in brute, high-speed computation.

Each neighborhood in a chip "metropolis" would have its own specialty. Like a real-world neighborhood, it would also have its own flavor, its own personality, its own reality.

A number of highly respected computer scientists believe that what they are all working toward is not AI, but "AL" – *Artificial Life.* Machine vision, machine touch, machine speech, machine mobility, machine intelligence will eventually merge and become machine life. Human beings will act as midwives to help give birth to a new species.

This development is probably many, many years away. It may have to wait until computer circuits shrink to the size of molecules, or until the electronic computer is replaced by new, light-based *optical* computers.

The reality of life on a chip, of entire, living cities existing on a chip is still sometime in the far future. But, in the meantime, the *illusion* of life on a chip is growing, as scientists are able to fit more and more complex structures onto tinier and tinier motes of matter.

Next Time:
The Architects of the Micro World. And what does all this have to do with personal computers and our children? ©

This program, for VIC with the usual 5K of memory, demonstrates an extraordinary method of getting numbers from the user. You can enter expressions *like 3+2 as well as the simple number five.*

Amortize

Amihai Glazer
Assistant Professor Of Economics
University Of California, Irvine

You're planning to buy a new house. Or perhaps a new car. But money is short and you must take out a loan. What is the monthly payment on the loan? What is the total interest charge? How much interest can you deduct from your income tax in the first year? Answers to these and other questions are provided by the program Amortize. As an added bonus, the program incorporates some techniques which you may want to use in your own programs.

Key in the program. On line 63993, simultaneously press the SHIFT key and the letter "O" key; this is an abbreviation for GOTO. On line 63992 the PRINT statement consists of a quote mark, a blank space, pressing the CTRL and the "2" keys simultaneously, and finally a quote mark. On line 63996 to enter the PRINT statement type a quote mark, then press the CTRL and the "7" keys simultaneously, then press the space bar seven times, and then close with a quote.

Once the program is in memory, type RUN and you will be prompted for the input. Notice that you can type as input not only numbers, but expressions as well. For example, suppose we let the loan be for ten years, so that for the number of months we enter 10*12; we let the interest rate be 13+3/8, and the amount of the loan (the principal) is 50000. Your friendly VIC20 will respond by showing that the monthly payment on the loan (PMT) is $757.65. You will then learn that after your third payment (MONTH=3) you still owe $49392.20, you will have paid a total of $1665.15 interest over these three months, and that $552.80 out of your third payment went to pay interest.

Look at the results for the last month, month 120. You will find that the total interest paid on the loan is $40918.87. (Yes, that sure is a lot of interest). Don't let the small amount of principal remaining, 87¢, bother you; such inaccuracies are inevitable when you can't make monthly payments including a fraction of a cent. One final caveat: some consumer lenders use the Rule of 78s to determine the reduction in principal each month. Therefore, the results the program gives you for any but the last month may be slightly different from what the bank may tell you. But the program will still give you the correct value for the monthly payment, and the correct value for the total interest charge.

The INPUT Technique

That's it for you folks who want to use the program without worrying about how it works. As mentioned, the user's input can be in the form of an expression, not merely a number (which is what the INPUT statement allows). Here's how this is done. Suppose we want to get a value for variable N. In statement 50, the computer printed out the characters "N=". The user types in any expression, say 10*12. Lines 63990 and 63991 accept the characters for this expression and print it out. The screen will now show N=10*12. (The pokes into locations 204 and 207 allow the cursor to be shown when the GET is invoked.) We then switch (in statement 63992) to printing in white so that the user will not be confused by the tricks we are about to play.

In statement 63993 the computer prints G┌ 63996. In statements 63994 and 63995 we POKE (into the keyboard buffer) instructions to go up to the screen line which says N=10*12, to execute that line, to go to the screen line which says G┌ 63996, and to execute that line. These instructions are executed when the END in statement 63995 is encountered. (For further details on this technique, see the fine article by Jim Wilcox, "Automatic Line Numbers" in the Premiere issue of *Home and Educational COMPUTING!*)

Having executed the instruction on the screen to GOTO 63996, the computer is now executing that statement. The computer switches back to printing in blue, erases from the screen all the garbage which it had printed in its machinations, and returns to the calling program.

The subroutine which starts in statement 63990 can be used in any program you wish. The calling sequence is exactly as shown in statement 50.

There is another useful technique in lines 132-136. These instructions allow the user to stop execution by pressing any key, and to continue execution by pressing that or any other key; the instructions transform the keyboard into a toggle switch. The logic is simple: if no key is pressed when line 132 is encountered, the program does not stop. If any key is pressed, the program waits until all keys are released, and then waits until a key is pressed. Execution then continues.

```
1 REM AMORTIZE  --BY
2 REM  AMIHAI GLAZER
3 REM UNIV. OF CALIF.
4 REM IRVINE,CA.92717
5 DEF FNR(X)=INT(100*      X+.5)
   /100
10 PRINT "{CLEAR}{REV}AMORTIZE"
20 PRINT "{03 DOWN}"
30 PRINT "NO. OF PERIODS"
40 PRINT "  (IN MONTHS)"
50 PRINT "N= ";:GOSUB    63990
60 PRINT "ANNUAL %INTEREST RATE"
70 PRINT "AR=";:GOSUB    63990
80 MR=AR/1200
90 PRINT "PRINCIPAL"
95 PRINT "P=";:        GOSUB 63
   990
100 PMT=(P*MR)/(1-(1+      MR)^(-N)
   )
105 PMT =FNR(PMT)
110 PRINT "{02 DOWN}PMT=",       F
   NR(PMT)
111 PRINT "{DOWN}PRESS RETURN KEY"
112 PRINT "TO CONTINUE OR STOP"
113 GETA$:IF A$=""        THEN 113

120 PRINT "{02 DOWN}"
130 FOR I=1 TO N
132 GET A$:IF A$=""       THEN GOT
   O 140
134 GET A$: IF A$<>""     THEN GOT
   O 134
136 GET A$:IF A$=""       THEN GOT
   O 136
140 RDUE =FNR(P*MR)
150 CUMR=FNR(CUMR+RDUE         )
160 P=P-PMT+RDUE
170 PRINT "{REV}MONTH=";I
180 PRINT " PRINCIPAL =";FNR(P)
190 PRINT " TOTAL INT.="; (CUMR)
200 PRINT " INT. DUE  =" ; (RDUE)
210 NEXT I
220 END
63990 POKE 204,0:POKE  207,0:GET A$
63991 IF A$<>CHR$(13)  THEN PRINT A$;
  :GOTO   63990
63992 PRINT " {WHT}"
63993 PRINT "GO63996"
63994 POKE 631,145:      POKE632,145:P
  OKE633,  145:POKE634,145:P
  OKE   635,13
63995 POKE 636,145:      POKE637,145:P
  OKE638,  13:POKE198,8:END
63996 PRINT"{02 UP}":FOR ZZ  =1TO3:PR
  INT"{BLU}              ":NEXT:PR
  INT"{03 UP}"
63997 RETURN
```

©

The Atari permits an excellent graphics animation technique – Player/Missile Graphics. However, smooth horizontal motion is easier to achieve than vertical; this article shows how to solve this problem using USR.

No Commotion Motion

Tina Halcomb
Carrollton, TX

In this article I will cover a fast motion routine for Player/Missile graphics.

The Atari has a built-in hardware register called the horizontal position register, which applies only to Player/Missile graphics. When the value in this register is changed, the Player/Missile moves to it's new position. If you were to change the number continuously by small amounts (such as in a program loop) you can obtain a smooth, sweeping motion.

Unfortunately, there is not a register available to us that pertains to the vertical position of the Player/Missile. Vertical movement can be achieved easily, however, by adjusting the RAM which represents this position. In routines that I have used in the past, the Player/Missile image was erased from the old position and redrawn in the new position. This technique works fine; however, any time you draw a picture you have to define its shape. This means that for each different shape, you need a separate drawing routine. Moving an image in this manner also produces a crawling effect.

I found that by rotating all 256 bytes of the Player-Missile directly in memory, the movement looked much like that of the horizontal move. And since we are moving all 256 bytes of the image, the shape is not important.

A program of this nature written entirely in BASIC would run very slow, and there would be no advantage to it. If I had attempted to write this program in assembly language, it would be obsolete before it was finished.

The USR function in the Atari allows you to add assembler subroutines to your BASIC programs. By using this function I was able to draw the Player/Missiles with BASIC and move them with my assembler routine. Line 90 of the BASIC program shows the parameters that the USR function operates from. The first number, 1536, is the address of the Assembly language routine. The second number is the actual Player/Missile number. The third and fourth numbers are the X and Y offsets (or how far you want to move the image), respectively.

In Player/Missile graphics RAM positioning, the first section of RAM is unused. The size of the unused or "free" RAM area is dependent upon the

Figure 1.

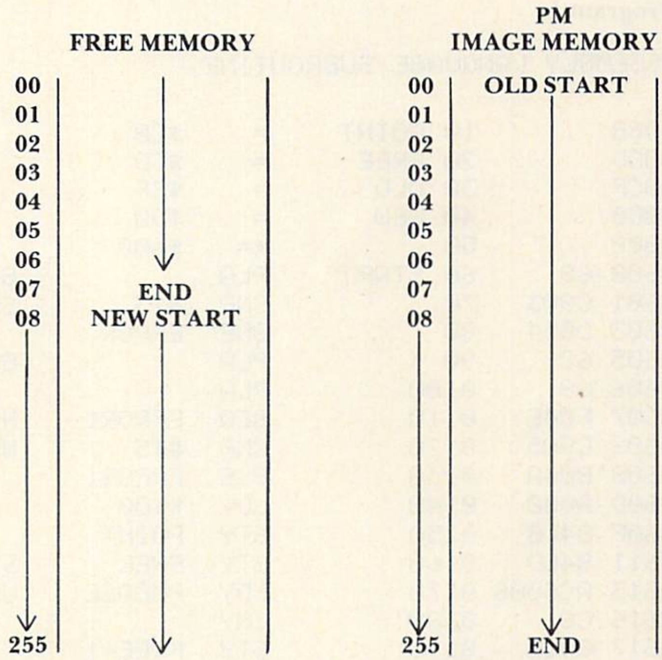
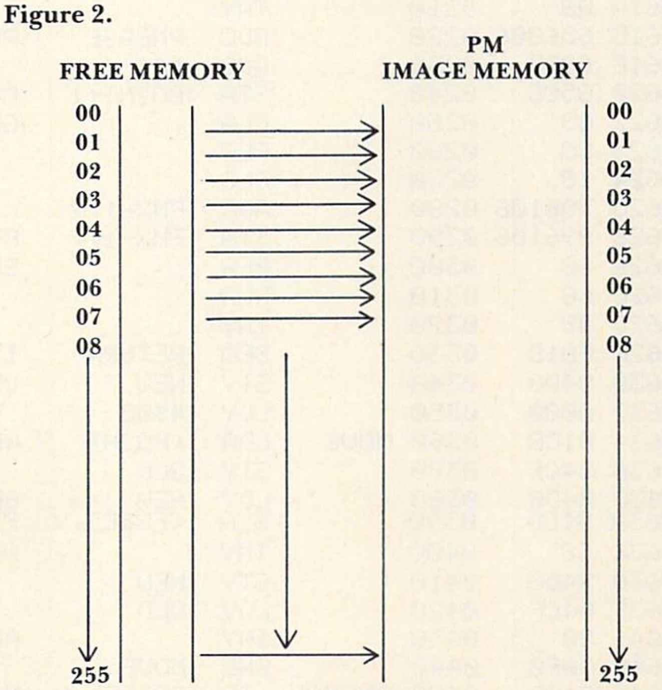


Figure 2.



line resolutionn that you choose to use. For the purpose of this program, I will discuss single line resolution. But bear in mind that the same program can be used for double line resolution with a few minor adjustments. So we have 768 bytes of free RAM at the top of our PMBase and the allocated RAM area for each player is 256 bytes.

To move the image I reposition the entire 256 bytes, with respect to the Y offset, into the free RAM space. For example, if the Y offset is seven, then byte zero of the player's present memory location is moved to byte seven of the free memory. This continues until byte 255 of the player is moved into byte six of the free memory and the rotation is complete. Figure 1 illustrates how this is done.

At this point the image is transferred byte for byte back to its original memory location (Figure 2.), and the Player/Missile image has swiftly moved into its new screen position.

With the speed and flexibility provided by this assembly language subroutine, you can write complex Player/Missile programs, all from BASIC.

Program 1.

```
ASSEMBLY LANGUAGE SUBROUTINE

00CB          10 POINT    =    $CB
00CD          20 FREE     =    $CD
00CF          30 OLD      =    $CF
00D0          40 NEW      =    $D0
0000          50          *=   $600
0600 68       60 START    PLA            GET # OF PARAMETERS IN USR
0601 C903     70          CMP  #$3       SHOULD BE 3
0603 D054     80          BNE  ERROR
0605 68       90          PLA            GET PLAYER #,THROW AWAY 1ST BYTE
0606 68       0100        PLA
0607 F04E     0110        BEQ  ERROR1    NO PLAYER 0
0609 C905     0120        CMP  #$5       NO PLAYER GREATER THAN 5
060B B04A     0130        BCS  ERROR1
060D A000     0140        LDY  #$00
060F 84CB     0150        STY  POINT
0611 84CD     0160        STY  FREE      SET UP FREE POINTER
0613 AC6006   0170        LDY  PMBASE    USING PMBASE+256 FOR FREE MEMORY
0616 C8       0180        INY
0617 84CE     0190        STY  FREE+1
0619 18       0200        CLC            GET START OF MEMORY IMAGE FOR THIS PM#
061A A8       0210        TAY
061B 6D6006   0220        ADC  PMBASE    ADD IN UNUSED MEMORY
061E 6903     0230        ADC  #$3
0620 85CC     0240        STA  POINT+1   PUT IT IN POINTER
0622 68       0250        PLA            GET THE X OFFSET, THROW AWAY FIRST BYTE
0623 68       0260        PLA
0624 18       0270        CLC
0625 796106   0280        ADC  PMX-1,Y
0628 996106   0290        STA  PMX-1,Y   PUT IT IN THE X REGISTERS
062B 68       0300        PLA            GET THE Y OFFSET, THROW AWAY FIRST BYTE
062C 68       0310        PLA
062D A8       0320        TAY
062E F01B     0330        BEQ  RETURN    IF 0 SKIP THIS SECTION
0630 84D0     0340        STY  NEW       USE Y TO ADD OFFSET TO MOVE
0632 A000     0350        LDY  #$00
0634 B1CB     0360 MOVE   LDA  (POINT),Y GET BYTE OF IMAGE
0636 84CF     0370        STY  OLD
0638 A4D0     0380        LDY  NEW       GET OFFSET
063A 91CD     0390        STA  (FREE),Y  PUT BYTE IN FREE WITH OFFSET
063C C8       0400        INY            ADVANCE OFFSET
063D 84D0     0410        STY  NEW
063F A4CF     0420        LDY  OLD
0641 C8       0430        INY            ADVANCE POINTER
0642 D0F0     0440        BNE  MOVE
0644 B1CD     0450 CHANGE LDA  (FREE),Y  MOVE REARRANGED IMAGE BACK
```

```
0646 91CB   0460         STA  (POINT),Y
0648 C8     0470         INY
0649 D0F9   0480         BNE  CHANGE
064B A005   0490 RETURN  LDY  #$05        WRITE THE X POSITION REGISTER TO THE HARD
WARE REGISTER
064D B96106 0500 PUT     LDA  PMX-1,Y
0650 99FFCF 0510         STA  53247,Y
0653 88     0520         DEY
0654 D0F7   0530         BNE  PUT
0656 60     0540         RTS              GO BACK TO BASIC
0657 A902   0550 ERROR1  LDA  #$02        ONLY 2 ITEMS LEFT OF STACK IF ENTERED HER
E
0659 A8     0560 ERROR   TAY              A HOLDS # OF ITEMS ON STACK WHEN ENTERED
HERE
065A 68     0570 PULL    PLA              PULL TWICE FOR EACH ITEM
065B 68     0580         PLA
065C 88     0590         DEY
065D D0FB   0600         BNE  PULL
065F 60     0610         RTS              STACK IS RESTORED, GO BACK TO BASIC
0660        0620 PMBASE  =    *           CONTAINS MSB OF MEMORY FOR PMGRAPHICS
0661        0630 TEMP    =    *+1
0662        0640 PMX     =    *+2         5 X POSITION REGISTERS
0660        0650         .END
```

Program 2.

```
1 REM LOAD ASSEMBLY ROUTINE INTO MEMORY
4 GRAPHICS 0
5 RESTORE 110
6 B=1536:I=0
7 FOR L=B TO B+100
8 READ A:I=I+A
9 POKE L,A
10 NEXT L
11 IF I<>13309 THEN PRINT "CHECK DATA ST
ATEMENTS FOR ERRORS":STOP
15 SETCOLOR 2,0,0:Y=48
19 REM CALCULATE ADDRESS FOR PLAYER-MISS
ILE GRAPHICS
20 A=PEEK(106)-16:POKE 54279,A:POKE 1632
,A:PMBASE=256*A
30 POKE 559,62:POKE 53277,3
49 REM CLEAR OUT PM MEMORY
50 FOR I=PMBASE+1024 TO PMBASE+2048:POKE
 I,0:NEXT I
60 POKE 704,216:POKE 705,85:POKE 706,45:
POKE 707,120
65 RESTORE 80
69 REM SET SHAPES OF PLAYERS
70 FOR I=PMBASE+1024+Y TO PMBASE+1028+Y:
READ A:POKE I,A:NEXT I
71 RESTORE 81
75 FOR I=PMBASE+1280+Y TO PMBASE+1288+Y:
READ A:POKE I,A:NEXT I
76 RESTORE 82
77 FOR I=PMBASE+1536+Y TO PMBASE+1542+Y:
READ A:POKE I,A:NEXT I
78 RESTORE 83
79 FOR I=PMBASE+1792+Y TO PMBASE+1798+Y:
READ A:POKE I,A:NEXT I
80 DATA 153,189,255,189,153
81 DATA 255,255,195,219,195,255,255
82 DATA 16,8,4,255,4,8,16
83 DATA 24,60,126,255,126,60,24
89 REM CALL ASSEMBLY ROUTINE
90 FOR X=1 TO 8000:C=USR(B,1,2,2):C=USR(
B,2,254,2):C=USR(B,3,1,255):C=USR(B,4,25
5,255):NEXT X
100 GOTO 90
110 DATA 104,201,3,208,84,104,104,240,78
,201,5,176,74,160,0,132,203,132,205,172,
96,6,200,132,206,24,168
111 DATA 109,96,6,105,3,133,204,104,104,
24,121,97,6,153,97,6,104,104,168,240,27,
132,208,160,0,177,203,132,207
112 DATA 164,208,145,205,200,132,208,164
,207,200,208,240,177,205,145,203,200,208
,249,160,5,185,97,6
113 DATA 153,255,207,136,208,247,96,169,
2,168,104,104,136,208,251,96,0,0,0,0,0,0
```
©

Learning With Computers

Glenn Kleiman and Mary Humphrey
Teaching Tools: Microcomputer Services
P.O. Box 50065
Palo Alto, CA 94303

Preschool Computing

Several friends of ours recently used computers for the first time. After we helped them get started, they enjoyed themselves and were eager to do more with the computer. They also had many comments, both positive and negative, about the programs they used. This is all very familiar – most people we have introduced to computers have responded in the same way. The difference is that these particular friends are between two-and-a-half and five years old.

Personal computers can be programmed to present lessons and games which encourage children's learning of such things as color names, numbers, letters, vocabulary, and perceptual skills. While using computers in these ways, children also take their first steps towards computer literacy. They realize that computers are an integral part of the world, and they learn about the keyboard, the cursor, disks and other aspects of computer use. We find children to be far more comfortable playing with computers than are most adults.

Programs For Preschoolers

A number of available programs are suitable for preschool children. Some of these programs are designed for children to use by themselves (once they are helped to start), others are for two or more children to use together, and others are designed for three way interactions among child, adult and computer. The programs all use graphics and minimize the need for reading. Some also make good use of sound. Many of these programs explicitly teach certain things, others are designed for the child to explore and create. The following programs are our young friends' favorites.

Hodge Podge (for Apple computers, from Dynacomp, Inc. 1427 Monroe Ave., Rochester, NY 14618). The instructions accompanying this program describe it as a "surreptitious learning" program. When any key is pressed, a song, animation, or picture related to the key is presented. Press A for an apple, B for a bear, C for a cat, D for a dog and so on. Some keys result in more elaborate displays. Press F and a farm is shown. "Old McDonald Had a Farm" is played and, at the appropriate time, an animal appears – a different animal each time. Press U and steps appear, a marker moves up the steps while tones going up the scale are played. Press ? and the alphabet song is played, with each letter appearing in turn. The numbers 1 through 8 each play a musical note and show that note on a staff. Zero turns the number keys into a miniature piano so children play their own tunes. Other keys result in displays illustrating addition facts and the concepts of smaller and bigger. This program is packed with easy to use educational features which can entertain a child for some time.

Above/Below/Left/Right (for Apple computers, from Advanced Learning Technology, Inc., 4370 Alpine Road, Suite 201, Portola Valley, CA 94025). This set of programs teaches the concepts given in its title. One program is on *above* and *below,* one on *left* and *right,* and one on all four terms. Each program follows a similar sequence. For example, to use the Above/Below program a blue bar (provided with the programs) is placed across the keyboard, dividing it into two sections, one for *above* and one for *below.* The program begins by showing a horizontal blue line across the middle of the screen. In the first part of the program the child can press any key. Pressing a key above the blue keyboard divider results in a colored bar appearing above the blue line on the screen. Pressing a key below the keyboard divider results in a colored bar below the blue line on the screen. Tones play each time a key is pressed. In the second part of the program the child is shown an incomplete colored bar either above or below the blue line. Pressing an appropriate key completes the bar. The next two parts of the program are similar, but two boxes are shown on the screen, one above the other. Pressing keys causes either the upper or the lower box to change color. In the final part of the program a colorful picture is created and the colors change in response to the child's key presses. Keys above the divider cause a change in the top half of the picture, keys below the divider cause a change in the bottom half. This program uses colors, music and pretty pictures to hold the child's attention.

Bumble Games (for Apple computers, also from Advanced Learning Technology, see address above) is a set of six programs for beginning number skills. The first program is a Guess My Number game, suitable for older preschoolers. A number line appears with the numbers zero to five. The child guesses a number, and is told whether his guess is less than, greater than, or equal to, the actual number. The child continues until the correct number is guessed. Then a marker on the number line flashes, music plays, a large colored number appears, and the corresponding number of beeps play. The other games involve two dimensional grids and other things more suitable for older children. Like Above/Below/Left/Right, the Bumble Games have some of the prettiest screen displays we have seen.

Letters and Numbers (for PET computers, from Teaching Tools: Microcomputer Services, P.O. Box 50065, Palo Alto, 94303). This program provides practice in matching and fill-in drills with letters and numbers. When matching is chosen, large letters or numbers (created with PET graphics) appear on the top of the screen. The child presses the matching letters or numbers on the keyboard. Correct answers result in smile faces, incorrect answers in an X and another try. When fill-in practice is chosen, a sequence of letters or numbers with one missing appears. The child is to type the missing one. Pressing the question mark key provides hints. The first hint is a display of the alphabet or digits, the second hint changes the area of the answer to reverse field. There are a number of options to be set by an adult, such as whether upper case letters, lower case letters, or numbers are displayed, how many practice sets are given, and how many letters or numbers appear in each set.

Frog! (for PET computers, from Cursor #19, The Code Works, Box 550, Goleta, CA 93116). This is a playful program which is enjoyed by people from age two on up. We know a three-year-old who calls all computers "Froggy" since he played this program. Froggy captures bugs and makes terrific sounds each time he gets one. The player controls Froggy by pressing keys on the number pad. The 1 to 9 keys form a 3 by 3 square. The higher up on the square you press, the higher up Froggy jumps. The more to the right you press, the longer Froggy's tongue extends. You do not need to know the numbers, just the location of the key. The bugs keep moving (you can set the speed) and you must catch them quickly enough to prevent Froggy from starving. Very young children easily learn which keys make something happen, and can understand how to get Froggy to jump higher.

Printsit (for the PET, from Cursor #24, see address above). This program lets the child create pictures. The child can select any of the graphic or alphanumeric symbols on the PET by simply pressing the appropriate key. The symbol can be changed at any time, and even reverse field characters can be used. The symbol is then plotted on the screen using the number pad to control the movement – the direction of movement corresponds to the position of the key on the number pad. If you have a printer with PET graphic symbols, the entire picture can be printed. Children enjoy creating a picture and being able to change it easily. We were told by one child that it's much better than trying to draw and erase with a pencil. PET graphics make it possible to create many interesting displays, and children especially enjoy getting a printed copy of their work.

Music! (for the PET, from Cursor #20, see address above) turns the PET into a one octave toy piano. Older PETs, which do not have built-in speakers, need a CB2 sound add-on to use this program. The child presses a key to play a note, and have it shown on a staff on the screen. The length of the note is determined by how long the key is held down. The child can create a tune, play it, change it, and save it on tape. This program makes it possible for young children to play with music, create their own tunes, and learn something about musical notation, without first learning to play an instrument.

Some Principles Of Software Design For Young Children

We have seen many programs designed for young children in addition to those described above. In our opinion and the opinions of children we have observed, the programs described are among the best available. What makes them better than the others? There are many important considerations in designing good educational software. Special care is needed in programs for young children since they cannot compensate for a program's shortcomings as well as older people, and can become confused or distracted easily. Six principles of software design we regard as especially important in programs for young children are discussed below. Although the programs we have described follow these principles for the most part, all the programs could be improved. As all software designers agree, there is no such thing as a perfect program.

Make the program easy to get started. For example, the Above/Below/Left/Right program starts with a simple four choice menu in which each choice is described by a colorful picture. This makes it easy for young children to select the option they want. The Letters and Numbers program requires answering a series of questions and so must be set up by an adult.

Make it easy for the child to understand how his actions cause things to happen. Children learn by realizing the relationship between what they do and the resulting action of the computer. For example, a child using the Hodge Podge program will not learn to associate A with apple until he realizes that an apple appears each time he presses

the A key. Programs for young children require great care so the child is able to discern these relationships. One thing we have observed is that young children often press a key and then keep holding it down. On some computers this registers as many key presses. We have seen children do this using the Froggy program. Froggy then jumps many times. Since the child thinks he has pressed a key just once, he does not realize how he caused the action on the screen.

Make it easy for the child to enter responses. All the programs described above require just one or two key presses for each response. Some of them also give the child a chance to erase and reenter a response before anything happens. This is useful as young children often press keys they did not intend to press. Input devices such as joysticks, game paddles, and light pens are often easier for children to use than keyboards, but few available programs use these devices.

Let the child work at his own pace. Children are very variable in how quickly they respond. Therefore, programs should be paced by the child's responses, not by internal timers (unless, of course, speed of response is an important part of the lesson or game). Programs that move on too quickly become frustrating. Programs that make the child wait become boring.

Hold children's attention, but do not distract them from the important information. Preschool children typically have short attention spans and may not be able to tell which information on the screen is most important. Screen displays have to be carefully designed to make them interesting without being confusing or distracting. Color, sound, and movement are very salient for children. They can be used to draw attention to educational aspects of the lessons or games. However, they can easily be misused and distract or confuse the child. For example, in the Above/Below program the lines and boxes that appear vary in color. This creates a pleasing visual display. However, it can also lead to confusion – several children first thought they were controlling the color of the lines, not the location. Another example of distraction is found in the Hodge Podge program. When music plays, the words DO, RE, ME, FA, SO, LA and TI appear in the corner of the screen as each note plays. This distracted some children from the main part of the display, and confused those who could not read the words. Several programs also leave a flashing cursor in the corner of the screen. Many children find this annoying. A prompt symbol that does not flash would better serve the same purpose.

Make sure the child can understand the feedback. In order to learn, the child must understand when his answer is correct and when it is not. We have found many cases of feedback that children can misinterpret. For example, some programs flash the child's name on the screen when a correct answer is entered. We have observed some children become upset by this – they thought they were wrong and the computer was yelling at them. We have even seen programs for preschoolers which use the words correct and wrong, with no other feedback. For pre-readers, things like smiling and frowning faces are much more appropriate.

Some minimal requirements for software design have not been included in our list of principles. We assume all software designers realize such things as programs should not crash when an unexpected key is pressed and feedback should not be insulting to the child. There is no excuse for programs that do not meet these minimal standards. The quality of educational software has increased greatly in the past few years and will continue to do so. Software designers are now placing more emphasis on making programs user-friendly and pedagogically effective. We expect the most exciting new developments in the next few years to be in software, rather than hardware, and look forward to seeing many innovative and well-designed educational programs. ©

Part I:

Machine Language: First Steps

Jim Butterfield
Toronto, Canada

Let's do a simple machine language project from ground zero and try to follow all the steps. We'll pick something very easy, and our coding will be for the CBM/PET. And we'll write it in BASIC first to make our objectives – and the flow of logic – clear.

Programmer F. R. Vescent wants to draw a bar graph of several numbers to the screen. He wants to output the graph as a series of decimal digits so that the viewer can see at a glance the length of a bar (for example, a bar 39 units long will end in a 9 digit). In BASIC, he codes:

```
200 DATA 15,10,30,35,28,28,15,0
210 READ V:IF V=0 GOTO 300
220 J=48:FOR K=1 TO J
230 J=J+1:IF J=58 THEN J=48
240 PRINT CHR$(J);
250 NEXT K
260 PRINT
270 GOTO210
300 END
```

You may code this and try it in BASIC. It's reasonably fast and a convenient way of representing numbers. But F. R. is a speed demon, and wants to code lines 220 to 260 in machine language. How does he go about it?

A few preliminary decisions: BASIC will pass the value of V (read from the DATA statement) to the program by POKEing it to some convenient place in memory. F. R. chooses hex $0300, or decimal 768, as the location. No particular reason except that it's not in use.

We'll place the program itself into the first cassette buffer, too. Now to plan out the logic.

F. R. grabs the back of an envelope and starts scribbling. He knows that he has three registers in the 6502 he can use for data: A, X, and Y. He knows that X and Y are especially handy for adding 1, so they seem to be useful for the BASIC variables J and K. After all, the K loop is stepping by 1, and we have the J=J+1 calculation on line 230. We'll need to use register A for output.

F. R. looks at the first BASIC command: J=48. He writes down LDX #$30. Meaning: Load into X the value hexadecimal 30, or decimal 48. X will hold the J-value, you may remember.

Now he looks at the next command: FOR K=1 TO J. He writes, LDY #$01, which means, Load Y with value 1. That's where K will start. Now he scrawls himself a note on the next line: "the Y loop comes back here." This part isn't ML coding, it's just a note so that everything can connect up. If he likes abbreviations, he might just note "YLOOP".

There's more to be done to complete the FOR K statement, but we'll do it later when the NEXT K comes up. On to the next command.

We read J=J+1, and F. R. codes INX. This means Increment X, and the value in the X register will become one greater. Since X contains the J-value, that's just what we want. Next line: IF J=58 .. calls for a test of the value of X. Let's invert the logic, and read this as [if j<>58 skip the rest of the line]. Same logic, right? And it will make the job easier. Now F. R. codes: CPX #$3A, or Compare X to hex 3A, decimal 59. On the next line, he codes BNE SKIP as a note to himself ... when he gets to the start of the next line, he'll be able to connect everything up, as the lawyers say. Now for the remainder of the current line: J=48 becomes LDX #$30 as before.

Now we've reached point SKIP where the code joins up, and F. R. notes this by writing SKIP in the left margin. We are ready to PRINT CHR$(J). Now, the value of J is held in the X register; to print it, it must be in the A register. That's easy: our hero codes TXA, Transfer X to A, and a copy of the value in X is transferred to A. Once we have it there, we print by calling a subroutine with JSR $FFD2. The address FFD2 hexadecimal contains the start of a subroutine to print the contents of A. When it's done the job, it will return and the program that F. R. is writing will resume where it left off. It's very much like a GOSUB in BASIC.

We've arrived at NEXT K. We can mentally translate that to K=K+1:IFK<=V..go back. Okay, K=K+1 neatly codes as INY, Increment Y. Now we'll need another compare, this time to value V which has been POKEd (we hope) to address hex 0300. F. R. writes: CPY $0300, Compare Y to address hex 0300, and below it, BCC YLOOP. Remember YLOOP? That was the note that F. R. wrote to himself quite a bit earlier. BCC means Branch Carry Clear: we can read it here as Branch Less Than, since the Branch will take place if Y is less than $0300. Note that we don't also Branch

Equal as we would like. F. R. scratches his head and makes a note to fix that up later somehow.

All we have left is a bland PRINT statement on line 260. Print what? A Return character, of course, to end the line. F. R. knows that this character is a decimal thirteen or hexadecimal OD, so he codes LDA #$0D to bring it into the A register and then JSR $FFD1 to print it as before.

The coding job's done, and F. R. notes down the last instruction: RTS, Return from Subroutine, which will cause the machine language program to return to the BASIC program which called it. He sits back. Then he notices that he's scribbled the whole thing on the back of his subscription renewal to **COMPUTE!** (the magazine gets a lot of programs that way), and decides to make a copy.

This is the program he transcribes onto a stenographer's note pad. The instructions are to the right of the vertical center line; the two notes, YLOOP and SKIP are just to the left. There's plenty of space to the left; he'll be using that later. Here is what his coding looks like:

```
      LDX #$30
      LDY #$01
YLOOP INX
      CPX #$3A
      BNE SKIP
      LDX #$30
SKIP  TXA
      JSR $FFD2
      INY
      CPY $0300
      BCC YLOOP
      LDA #$0D
      JSR $FFD2
      RTS
```

You should be able to relate this program to the BASIC program previously given. One important point: where the "#" sign is used – you may call it hash, numbers, or pounds sign – we want the program to use the actual value. Where it is not used, we want the program to use the contents of an address. The "#" sign signals immediate addressing – not really an address at all, but the actual value.

F. R. Vescent's program is not ready to run yet. He has written it in Source (or Assembler) language for his own convenience. The computer can run only Object (or Machine) language. He must translate from Source to Object language by knowing the codes. This translation job is called *assembly*. We'll tackle it next time. ©

Telecommunications:

Sending Programs Over The Phone

Michael E. Day
Chief Engineer
Edge Technology

One use for a modem is to transfer data between your own computer and another one. This mundane aspect of telecommunications can be one of the more interesting and rewarding uses of both the modem and your computer.

The usual method of getting programs into your computer is to either enter them by hand through your keyboard, or to obtain a disk or cassette with the desired program on it. While this is fine if the program desired is readily available, it can make things a bit difficult if it is not. If, for instance, a friend of yours has a BASIC program you would like to have, the usual procedure is for him or her to make a copy of the program on cassette or disk and give it to you. If it is not too large, you might get a printed copy. If you have a cassette and your friend has a disk, the usual response is to not bother. If both of you had a way to transfer the program over the phone, though, you could easily get the program. Another interesting aspect of this is that the program can be transferred to you instantly. With the modem all it takes is a phone call.

In order to make these calls, there must be some agreement as to how you will transfer this information. The actual mechanics can be quite complex. An agreement about how to make the actual transfer is called a *communication format* or *protocol*. Although there are certain basic requirements needed to make the transfer, there is no standard format for the actual details of the transfer.

Making A Link

The general structural requirements of data transfer are standardized to some extent. This is largely due to the basic requirements of an actual transfer.

The first thing that is done is to establish the communications link. This is done when you call your friend, make the arrangement to do the transfer, and turn on the modems to begin the transfer. Next, the computers must synchronize themselves, and then, finally, make the actual transfer.

The transfer itself is broken into small pieces called *records*. The records generally consist of 128, 256, or 1024 characters (bytes). The actual record size is normally chosen to fit the particular system that the transfer program is running on and, in fact, is one of the reasons for the lack of standardization of these transfer programs.

One popular operating system that is used on some computers is CP/M*. Because CP/M was originally based on the IBM standard eight inch floppy disk which uses a storage method of 128 bytes per record. So, naturally enough, the record length that is usually chosen for the data transfer through the modem is 128 bytes too. This doesn't mean that the transfer *has* to take place in 128 byte records. But a decision had to be made as to what the record length would be and, since the system stored data in 128 byte groups, this was chosen. It could just as easily be 256 or 1024 bytes.

Actually a 128 byte record is a fairly reasonable size since, at 300 baud, (the number of characters sent per second) takes a little over four seconds to transmit a record. 256 bytes would take over eight seconds, and 1024 bytes would take over 30 seconds. The idea here is to keep the transfer size down so that, if an error does occur, not too much time is wasted retransmitting it. On the other hand, you don't want it broken down into such small pieces that the overhead involved in handling the records significantly retards the transfer time.

Overhead time can be considered the time it takes to acknowledge the receipt of the record. In a simple transfer program this would be a single character. Another part of the overhead that must be considered is the turnaround time of both the computer systems and the phone line. On a local call this generally averages out to about three or four character times (assuming 300 baud). On a long distance call, this can stretch out to eight to twelve character times. (If the call is via satellite, it will be around 40 character times.)

Assuming that the call is local, this means that the overhead would be about five characters. For 128 byte records this would be about 4% overhead. For 256 byte records it would be 2%, and for 1024 byte records it would be 1/2%. This has to be balanced against the expected error rate. The phone line has an average error rate of about one error in every 10,000 bytes of data that is transferred. If the phone line is weak or noisy, it can get much worse. With 1024 byte records, this means that about one out of every ten records will be bad, (a 10% error rate) so the 1/2% transfer rate is lost in the 10% error rate. With 256 byte records, the error rate is down to 2.5%, and with 128 byte records it is 1.25%.

Assuming a 128 byte record format, and ac-


APPLE • ALTOS • ATARI • MAXELL • DYSAN • EPSON • CCS • SHARP • CASIO • HP • VERBATIM • MEMOREX • SOROC • CORVUS • ADDS
MICRO
BUSINESS WORLD INC.
Information Line (213) 996-2252
TOLL FREE MAIL ORDER LINES
(800) 423-5886 Outside Calif
HEWLETT PACKARD
SPECIAL OF THE MONTH
HP 41CV
MSL 325.00
OUR PRICE $239.00
Save 86.00
EXPANSION BOARD
Q STAR
16K RAM BOARD
MSL 199.00
OUR PRICE $99.00
Save 100.00
MICRO-SCI
APPLE II+ COMPATIBLE DRIVE
W/ CONTROLLER
MSL 549.00
OUR PRICE 429.00
Save 216.00
COMPARE TO APPLE DRIVE
W/O CONTROLLER
MSL 449.00
OUR PRICE 379.00
Save 146.00
COMPARE TO APPLE DRIVE
APPLE IS A REG. TRADE MARK OF APPLE COMPUTER
IBM
IBM PERSONAL COMPUTER INCLUDES:
2 Drives
Colour Graphic Board
64K Memory
CALL FOR PRICE
*Subject to availability
FRANKLIN 64K
ACE 100
APPLE COMPATIBLE
INTRODUCTORY OFFER
MSL 1595.00
CALL FOR PRICE
The Franklin ACE* 100 is a professional personal computer that is hardware and software compatible with the Apple II† and includes many features not found on the Apple unit. All programs written for the Apple II will run on the Franklin ACE 100 without modification including those using high and low resolution black and white graphics. The ACE 100 is plug compatible with Apple. All peripherals designed for the Apple II will operate with the ACE 100 without modification.
The Franklin ACE 100 is a personal computer with the power, quality and reliability to meet the most demanding applications of the business professional. It comes complete with 64K of RAM memory which fulfills the demanding memory requirements of VisiCalc and CP/M based programs. The system includes a full upper and lower case keyboard with true shift capability and a numeric pad. The computer generates a full character set to the video screen that displays upper and lower case characters.
The 72 key keyboard includes an alpha lock key which simplifies operation with existing Apple software. The numeric pad cluster includes special keys such as period, plus, minus, greater than (go to) and asterisk (multiply) that are used frequently with VisiCalc.
The Franklin ACE 100 includes a joystick/ game paddle connector, a speaker and eight peripheral connectors. A built-in fan and a 50 watt power supply permit all eight peripheral connectors to be used without power or overheating problems.
Franklin ACE 100 Features
• Apple II compatible
• 64K of RAM memory
• Upper and lower case
• Typewriter-style keyboard
• Twelve key numeric pad
• Alpha lock shift key
• VisiCalc friendly
• 50 watt power supply
• Built-in fan
COMPARE OUR COMPUTER
FRANKLIN ACE 100 compared to APPLE II
FEATURES | APPLE II | FRANKLIN ACE 100
Computer with 48K RAM Memory | $1530.00 | STD
Microsoft 16K RAM Card (16K RAM Memory) | 195.00 (option) | STD
ABT 10 Key Pad | 125.00 (option) | STD
Videx Keyboard Enhancer | 149.00 (option) | STD
R.H. Electronics Super Fan II (Muffin Fan) | 69.00 (option) | STD
TOTAL | $2068.00 | $1595.00
APPLE IS A REG. TRADE MARK OF APPLE COMPUTER
LE MONITOR
METAL CABINET
SANYO TYPE
9" GRN. PHS. MONITOR
12" GRN. PHS. MONITOR
MSL 189.00
OUR PRICE $119.00
Save 70.00
APPLE II PLUS 48K
*Subject to availability
CALL FOR PRICE
APPLE IS A REG. TRADEMARK OF APPLE COMPUTER
HEWLETT PACKARD
HP 85
The professional's choice: Hewlett-Packard's HP-85
MSL 2750.00
OUR PRICE $1975.00
Save 775.00
HEWLETT PACKARD
HP 125
OUR PRICE $1975.00
MSL 2750.00
Save 775.00
DUAL MASTER DRIVE
MSL 2500.00
OUR PRICE 1595.00
Save 905.00
SAM CAN!
XEROX 820
WORD PROCESSING SYSTEM (INCLUDES SOFTWARE)
MSL 3495.00
OUR PRICE $2649.00
Save 846.00
NEC
STUDENT SYSTEM 64K
*NEC PC 80001
*NEC PC 8012
*NEC PC 8031
12" Grn. Phs. Video Monitor
MSL 2839.00
OUR PRICE $2095.00
Save 744.00
HEWLETT PACKARD
HP-87
MSL 2495.00
OUR PRICE $1749.00
Save 746.00
commodore
VIC-20
Vic-TV Modual $19.00
Vic Cassette $69.00
Vic Disk Drive $ Call
Vic 6 Pack program $44.00
MSL 299.00
OUR PRICE $255.00
Save 44.00
EPSON
MSL | OUR PRICE | Save
MX 80 645.00 429.00 216.00
MX 80FT 745.00 535.00 210.00
MX 100 995.00 719.00 276.00
commodore
STUDENT SYSTEM
4032 - 32K 40 COL CRT
4940 - DUAL DISK DRIVE
MSL 2590.00
OUR PRICE $1978.00
Save 612.00
WE RESERVE THE RIGHT TO CORRECT TYPOGRAPHICAL ERRORS. THIS AD SUPERCEDES ALL PREVIOUS ADS.
RADIO SHACK 48K
TRS 80 W/2 DRIVE MOD III
MSL 2495.00
OUR PRICE $1799.00
Save 696.00
1 Year Extended WARRANTY
$99.00
INQUIRE
$AVINGS
$ERVICE
$ELECTION
$ATISFACTION
MICRO MAIL ORDER
BUSINESS WORLD WAREHOUSE
18720 Oxnard, #108
Tarzana, CA 91356
OUTSIDE CA CALL TOLL FREE 1 (800) 423-5886 IN CA (213) 996-2252
Telex: 182852
Answer: MICKO TZNA
Name (Please print)
Address
City State Zip
Qty Make Model Description Price Total
Certified Check or M.O.
Bank Wire Transfer
Allow 2 weeks clearance for personal check
CREDIT CARD #
Exp. Date
Signature
*California residents add 6% sales tax
**Add 3% Shipping & Handling — Add 3% surcharge for credit cards. Orders cannot be shipped unless accompanied by payment, including shipping, handling and tax where applicable.
TOTAL ORDER $
TAX IF APPLICABLE*
SHIPPING & HANDLING**
TOTAL ENCLOSED $
VISA
MasterCard
MAXELL • DYSAN • EPSON • CCS • SHARP • CASIO • HP • VERBATIM • MEMOREX • SOROC • CORVUS • PERSONAL SOFTWARE • CCS
HP • D.C. HAYS MICROCOMPUTERS • PAPER TIGER • GTE • DIABLO • BMC • CENTRONIC • NEC • HAZELTINE • MEMOREX • AMPEX • PERSONAL SOFTWARE • ADDS • XEROX • DIABLO
BMC • CENTRONIC • NEC • HAZELTINE • AMPEX • VERBATIM • HITACHI • HAYDEN • CANON • NOVATION • D.C. HAYS MICROCOMPUTERS • PAPER TIGER • PERSONAL SOFTWARE • GTE

counting for one error in a transfer of 10,240 bytes, the transfer time would be about six minutes. If the records were 256 bytes long, the transfer time would be about 5.9 minutes, and with 1024 byte records it would be about 6.5 minutes. It would seem that the 256 byte format would be the best choice, but another factor must be taken into account: the error detection method used.

In the method used for the CP/M system, it is very simple, and the more bytes it is required to check, the greater the chance is that it will miss an error. Because of this, the 128 byte format is a better choice even though there is a small increase in the transfer time. If a better error detection method were to be used, it would probably be better to use the 256 byte record format.

A Transfer Format

The actual structure of the record that is transferred varies from system to system as well. In fact, it is even less standardized than most other parts of the data transfer format.

Since there is no real standard for the record format, I will describe one of the more heavily used formats. This format got its start on CP/M-based systems and originally appeared in a program written by Ward Christensen called, appropriately enough, "MODEM." The first problem when dealing with CP/M is its refusal to acknowledge the existence of a modem. So a transfer program must provide its own link to the modem.

To begin the transfer, the receiving computer sends an ASCII NAK (15H) signal every couple of seconds until the sending computer sends an ASCII ACK (06H). This is the synchronization part of the transfer. The original modem program assumed that the program was predefined at both ends, so, once synchronization was achieved, the data was immediately sent.

The record format that is used consists of a *header,* the data, and finally a checksum character for error detection. The header consists of an ASCII SOH character (01H) followed by the current record number (starting with number one) which is an eight bit value. That is followed by the same number, but inverted. (That is, if record 01H is being sent, then the second number sent will be FEH.) This is then followed by the data itself for the next 128 bytes. Finally, one more character is sent which is the checksum.

The checksum is an eight bit value that is the sum (without carry) of all the data bytes sent. The sending computer then waits for the receiving computer to acknowledge that it received the data. The receiving computer compares its own calculated checksum against the one that the one that the sending computer sent and, if they match, it sends an ASCII ACK character (06H). If they don't match, it sends an ASCII NAK character (15H), indicating that it didn't receive the data correctly. If the sending computer receives a NAK, it will send the record again. If, after ten tries it is unable to send the record it gives up and aborts the transfer. After all of the records have been sent, a final ASCII EOT (04H) is sent indicating that the transmission is completed.

Some Problems

There are several problems with the format that is used and some of the later versions attempted to correct for this. Unfortunately, this created a new problem since any change in the basic format meant it was incompatible with the old format. This tended to lead towards a real mess with patches to allow for compatibility to the old programs. Discounting the versions which were simply adaptations for different modems, some of the differences that have occurred are the addition of the program identifier so that the sending computer can tell the receiving computer the program name instead of requiring the operator at the receiving computer to specify it. There was also change from the checksum format to a CRC format. The identifier has been implemented several ways, but the most popular version is also one of the strangest implementations.

After synchronization has been achieved, the currently popular program (MODEM7) sends the filename a character at a time. That is, it sends a character and then waits for an acknowledge (ASCII ACK) from the receiving computer. Then it sends the next character of the filename and repeats this until the entire filename has been sent. After that, it waits for the receiving computer to send the checksum of the filename and then compares the received checksum against its own internally calculated checksum. If they are equal, the sending computer sends an ASCII ACK character. The sending computer goes back and waits for resynchronization. (Waiting for an ASCII NAK character.) At last, it starts receiving data normally after the resynchronization is achieved.

If there was a checksum error, the sending computer sends a bad name character which, for no particular reason, was defined as an ASCII *u* (75H) and goes back to allow resynchronization and retransmission of the name. If, after ten tries the name cannot be sent, the transfer is aborted.

Improved Error Detection

Another problem that some versions have corrected for is that the checksum method of error detection is not the most accurate means of detecting an error. A CRC (cyclic redundancy check) is a far

better way of detecting errors. The most common CRC that is used is a 16 bit polynomial, defined as (X+1)*(X^15+X+1). By starting with a value of zero in the CRC and passing all the received data through the CRC routine, when the final CRC received passes through the routine, the final result will be 0 (if no errors were encountered).

There is no single best way to transfer programs via a modem, but there are some methods which are better than others.

CP/M is a trademark of Digital Research

Computing A Checksum

Checksum generates a sum of the data that is passed through it by adding (without any carry) each byte with the sum of the previous bytes. The checksum is initially set to zero and the final result is sent to be compared against the independently computed checksum at the receiver computer. Since the checksum is an eight bit value, only a single byte of data needs to be sent. It is quick and easy to perform this with a computer. The checksum method can only reliably catch single bit errors. Although it does do reasonably well with multibit errors, the percentages can rapidly drop to the realm of coin toss odds.

To use the checksum program enter it with the data in the accumulator. The result is saved in location CHKSUM for later use. The location CHKSUM should be cleared to zero at the start of sending the data stream.

8080 VERSION

```
CHKSUM  PUSH PSW
        PUSH B
        MOV  C,A
        LDA  CHKSUM
        ADD  C
        STA  CHKSUM
        POP  B
        POP  PSW
        RET
```

6502 VERSION

```
CHKSUM  PHA
        PHP
        CLC

        ADC  CHKSUM
        STA  CHKSUM
        PLP
        PLA
        RTS
```

This shorthand version for the 8080 assumes that a register in the CPU does not get destroyed by the calling program, and that it is acceptable for the flags to be destroyed.

8080 VERSION

```
CHKSUM  ADD C
        MOV C,A
        RET
```

The CRC Method

CRC computes CRC-16 crcsum from the polynomial (X+1)*X^15+X+1)

A CRC method of error detection is far superior to the checksum. By using a 16 bit value for the sum instead of an eight bit value, a much improved detection capability is achieved. By using a polynomial of the proper type, the 16 bit value can be used far more effectively as well. The provided polynomial can detect errors of up to 17 bits.

Since the CRCSUM generated is a division remainder, a CRCSUMed data sequence can be verified by running the data through the CRC, and then running the previously obtained CRCSUM through the CRC. The resultant CRCSUM should be zero. When the CRCSUM itself is transmitted, it should not be run through the CRC as this would disrupt the result. Also, the MSB {H} must be run through the CRC first then followed by the LSB {L} when checking the CRCSUM.

To use this routine, enter with the byte to be CRCSUMed in *A* (accumulator). The CRCSUM is automatically updated upon passing the data through this routine.

8080 VERSION

```
CRC     PUSH PSW
        PUSH B
        PUSH H
        MVI  B,8
        LHLD CRCSUM
CRC1    RCL
        MOV  C,A
        MOV  A,L
        ADD  A
        MOV  L,A
        MOV  A,H
        RAL
        MOV  H,A
        RAL
        XRA  C
        RRC
        JNC  CRC2
        MOV  A,H
        XRI  80H
        MOV  H,A
        MOV  A,L
        XRI  05H
        MOV  L,A
CRC2    MOV  A,C
        DCR  B
        JNZ  CRC1
        SHLD CRCSUM
        POP  H
        POP  B
        POP  PSW
        RET
        END
```

6502 VERSION

```
CRC     PHP
        PHA
        STX  XTEMP
        LDX  #08H
CRC1    ASL  A
        ADC  #00H
        STA  CRCTMP
        LDA  CRCSUM
        ASL  A
        STA  CRCSUM
        LDA  CRCSUM+1
        ROL  A
        STA  CRCSUM+1
        ROL  A
        EOR  CRCTMP
        LSR  A
        BCC  CRC2
        LDA  CRCSUM+1
        EOR  #80H
        STA  CRCSUM+1
        LDA  CRCSUM
        EOR  #05H
        STA  CRCSUM
CRC2    LDA  CRCTMP
        DEX
        BNE  CRC1

        LDX  XTEMP
        PLA
        PLP
        RTS
        END
```

Table 1. Send A 256 Byte Program

Sending Computer	Receiving Computer
— Synchronization Time —	
	NAK
No Response	
	NAK
ACK	
— Synchronization Achieved —	
— Send A Record —	
SOH	
Record #1	
Record #1 (Inverted)	
Data	
¦	
128 Bytes	
¦	
Checksum	
	ACK
SOH	
Record #2	
Record #2 (Inverted)	
Data	
Checksum	
	NAK
SOH	
Record #2	
Record #2 (Inverted)	
DATA	
Checksum	
	ACK
— All Done, Stop Sending —	
EOT	
	ACK
— Transmission Done —	

©

FORTH is a programming language which falls in between BASIC and machine language – in speed, and difficulty. **COMPUTE!** *will be covering FORTH on a regular basis and, to start things off, we have Jim Butterfield's introduction to this increasingly popular language.*

And So FORTH

Jim Butterfield
Toronto, Canada

Suppose you were offered a collection of pre-tested machine language subroutines which could do many common programming jobs. Nice? Of course. And suppose it were pointed out to you that if there was anything missing from the collection, you could write your own, mostly by calling sequences of the pre-written subroutines. In fact, you could build your own library.

With such a package, you'd have the speed of machine language and the ease of programming that comes with calling up prewritten code. The best of all possible worlds: and that's more or less what you get with FORTH.

FORTH isn't exactly a language. It's a set of useful routines organized in such a way that you can expand the collection with your own items, building on what's previously been written. You don't exactly program; you build larger and larger modules out of the smaller pieces, and what results is in machine language (more or less). The code doesn't need to be interpreted or compiled; each program module is ready to run as soon as you have defined it. And many FORTH writers do just that: they check out each module as it is written.

How It Works

FORTH was originally written for the PDP-11 computer. This computer has an unusual addressing mode which allows subroutines to be called indirectly. All you need is a list of the subroutine addresses – no instructions or op codes – and you can arrange to execute each subroutine in the list in turn. This type of organization is called *threaded code*. It's economical of memory: what could be briefer than just the address of the subroutine?

In most microprocessors, threaded code is achieved with an "inner interpreter" that picks out each address from the list and then sets up the subroutine call. There's a small penalty in running time for doing this extra chore, but it's not great.

FORTH allows each routine to have a name. Type in the name, and the routine runs. Type in several names, separated by spaces, and each routine runs one after the other. If I type "1 2 + ." the computer will perform the routine called "1", placing the value one onto the stack; then routine "2", placing the value 2 above the 1 on the stack; then "+" which takes the two values from the stack, adds them, and returns the result to the stack; and finally "." which takes the value from the stack and prints it. Result: the total, 3, is printed.

You may define your own routine. The colon is used to start a definition line: everything following it (up to the ending semicolon) is compiled into the appropriate sequence of subroutine calls. So we may type ":1+2 1 2 + . ;" and define a new routine called "1+2" which will add 1 to 2 and print the result. This new item becomes a permanent part of FORTH at least until we power down; and can become truly permanent if we save the new FORTH to storage.

This is one of the problems in trying to pin down FORTH as a language. Which is the true FORTH – the one you brought home from the store, or the one you are now using which includes handy routines that you have subsequently devised? It becomes very hard to say "FORTH can't do this..."; if you do, scores of outraged users will reply, "Yes it can; I added it," or, "I bought a version with extensions that do that..."

There's little point in defining a routine which calculates a fixed value of three; but we can call on the whole FORTH vocabulary to build ever larger commands. The existing vocabulary provides input, output, calculate, testing, and loops. With these, you can construct almost any logical combination. If there's anything that's not there, you can usually put a built-in assembler to work and do the job in machine language.

The FORTH Estate

There's a standard, more or less, for FORTH. It's defined by FIG-FORTH. But what you are likely to obtain from a supplier – at prices ranging from free to several hundred dollars – may have a few pieces missing (like the assembler) and is likely to have a number of pet extensions and support features, such as editors or file handlers.

One aspect of FORTH can give difficulty to beginners. The language is based on the use of a stack (two of them, in fact), and data is manipulated

OLYMPIC SALES COMPANY

Authorized Dealer

apple computer

LARGE INVENTORY LOWER-THAN-EVER PRICES

NEW APPLE FAMILY SYSTEM
- ★ 48K Apple II +
- ★ Disk w/controller
- ★ RF Modulator
- ★ 7 Personal & game disks

Retail: $2495.00 Call for BEST Price!

Large Inventory of:
Disk with controller DOS 3.3
Second Disk Drives
Pascal–Fortran–Cobol languages
Dow Jones & Quote reporter
Graphics Tablets
Visicalc for Apple II & III
Smarterm 80 column card
Micromodem II by DC Hayes

WE ALSO CARRY SOFTWARE!
Personal Software, Peach Tree Software, Microsoft, Micropro, Innovative American, System Plus and more.

We are an authorized servicing Apple dealer for Apple II & III. Immediate delivery on all models—we carry an enormous inventory of Apple products at all times!

Just some of the software in stock NOW—
Magic Window, Raster Blaster, Ultima, Home $ Minder, Tax Preparer, Pool 1.5, Typing Tutor, Olympic Decathalon, ABM, Robot War, Castle Wolfenstein, Cranston Manor, Expediter, OO-Topos, Cyborg, Gorgon, Falcons, Wizardry, Personal Filing System, PFS: Report, Flight Simulator. . .

Texas Instruments Home Computer

New—1982 Model with full typewriter-style keyboard, U/L case & more!
TI-99/4A New KEYBOARD **$369.95**

Retail $950.00 *Your Cost* $369.95!

New keyboard

	Retail	Your Cost
10" color monitor for 99/4	650.00	319.95
32K Exp. mem. module	399.95	314.95
Extended Basic, a MUST for 32K module	100.00	75.00
Speech synthesizer	150.00	129.95
Disk memory drive	500.00	397.95
RF modulator	69.50	49.50
Telephone coupler (modem)	225.00	189.95
Printer (solid state)	400.00	319.95

We carry a large inventory of software, & accessories

HEWLETT PACKARD

HP-11C Advanced Programmable Scientific LCD *Retail: $130.00* *Your Cost:* **$114.95**

HP-12C Advanced Programmable Financial LCD *Retail: $150.00* *Your Cost:* **$129.95**

	Retail	Your Cost
HP-125 New Microcomputer 64K CPU/Terminal/Keyboard	3750.00	3195.00
HP-85 Microcomputer—built-in printer/monitor	3250.00	2499.00
HP-83 Microcom—built-in mon	2250.00	1799.00
HP-2631B Printer, dot mtrx	3950.00	3199.00
HP82905A 80 col printer	945.00	755.00
HP-2601A Letter quality prter		3495.00
HP-82901M Dual master	2500.00	1999.00
Call us for the lowest prices on 7 disk drives		
Graphics plotter 7225	2450.00	2089.00
Personality module—a must for 7225	750.00	636.00
Graphics Tablet 9111A	2050.00	1699.00
HP-41CV	325.00	256.00
Card Reader for 41CV	215.00	171.00
Printer for 41CV	385.00	294.00
Quad Ram	95.00	84.95
Optical Wand for 41CV	125.00	99.95
HP-41C Calculator	250.00	188.95
Memory mod. for 41C		26.95
HP-97 Progrmble printer	750.00	579.95
HP-34C Progrmble scientific	150.00	117.95
HP-38C Progrmble business R/E	150.00	119.95
HP-32E Advanced scientific	55.00	48.95
HP-37E Business management	75.00	58.95

We carry a complete inventory of Libraries, accessories & supplies.

ATARI COMPUTERS

Model		Retail	Your Cost
800	16K	1080.00	759.95
810	Disk Drive	600.00	469.95
Atari	VisiCalc	200.00	169.95
825	Impact Printer (Centronics 737)	1000.00	779.95

SUPER SPECIAL—ATARI 400 (16K)
Retail: $595.00 Your Cost: $339.95
Language Cart. not included—Opt'l at $54.95

Mattel 1670 Race Horse Computer
Ret. $125.00 Y/C: $24.95

ATARI

Programmable Color TV Game Video Console: *Retail:* $225.00 **$137.95**

MATTEL ELECTRONICS
INTELLIVISION
Retail: $325.00 *Your Cost:* **$234.95**

VIDEO TAPES

Min order: 3, mixed OK

SONY	L500	2 hr	$11.89
SONY	L750	3 hr	14.69
RCA	VK250	6 hr	13.95

TEXAS INSTRUMENTS

	Retail	You Pay
TI-59	295.00	179.95
TI-3011 LCD Stu Slide/NEW		18.95
TI-35SP LCD Sci		22.50
TI-40 LCD Sci/NEW		28.95
Bus. Anal. I		19.95
Bus. Anal. II		44.95
Bus. Card		39.95
MBA		54.95
Invest. Analyst		48.95
TI-54 Adv. LCS Sci/NEW		39.95
TI-5511 NEW		44.95
TI-57 Prog. Sci		39.95
TI-58C 480 Step. Prog.		89.95
PC-100C Print/Plot		149.95
LCD-Programmer/NEW		59.95

OHIO SCIENTIFIC

Professional Computers

	Retail	Your Cost
C8PDF—48K	3495.00	3195.00

- Dual 8" Drives
- 64 col x 32 line/color
- .7 MIPS —*FAST!*
- Many more stndard featres

Fortran & Pascal available

Many other OSI products available —at discounted prices, of course!

VISA / master charge

Main Showroom & Offices:
216 South Oxford Avenue
Los Angeles, CA 90004
WE HONOR VISA and MASTERCHARGE
TELEX: 67 34 77
ORDER DESKS open 7 Days a Week!
7:00 AM to 7:00 PM Mon thru Sat
Sunday Noon to 5:00 PM
Order Desks: (213) 739-1130

TOLL-FREE (outside Calif.) 800-421-8045
TOLL-FREE (within Calif.) 800-252-2153

All goods subject to availability; this ad supersedes all previous ads; we are not responsible for typographical errors; we will meet or beat any advertised prices if the competition has the goods on hand.
Minimum shipping and handling $4.95.
All orders subject to verification and acceptance.

EPSON PRINTERS

	Retail:	Your Cost
MX80	645.00	499.00
Optional Graftrax Chip 80		95.00
MX80 FT	745.00	599.00
MX100	995.00	749.00

DIABLO 630

Letter Quality daisy wheel w/tractors
Retail: $3395.00
Y/C: $2595.00

PAPER TIGER PRINTERS

460	995.00	895.00
460G graphics	1094.00	969.95
560	1295.00	1099.00
560G graphics	1394.00	1195.00
445	795.00	695.00
445G	894.00	789.00

SANYO MONITORS High resolution

13" Color (new) high quality	550.00	419.95
12" Green phosphorous	360.00	259.95
12" Black & white	340.00	239.95
15" Black & white	370.00	259.95
9" Black & white (the best seller)	235.00	169.95

AMDEK (Leedex) High Quality Monitors

100	12" B/W, 12 MHz	179.00	139.95
100-G	12" Green, 12 MHz	199.00	174.95
300-G	12" Green. 18 MHz	249.00	199.95
Color I	13" Color, NTSC comp. input,	449.00	339.95

SPRING SPECIAL

No need to wait any longer for our controller development system;

The MMC/03S is back! For $250.00 you get:

1 MMC/03D Microcomputer, IKRAM, 2-6522's ZIF sockets for CPU and EPROM, 20MA full duplex current loop, & crystal clock.

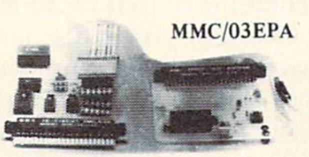
1 MMC/03ICE In circuit emulator, works off any 6502-based system including AIM, SYM, KIM, OSI. (Apple version slightly more, PET version a lot less)

1 MMC/03EPA EPROM programming adaptor; Programs - 2758's, 2716's, and 2532's.

MMC/03D

MMC/03ICE

MMC/03EPA



R. J. BRACHMAN ASSOCIATES, INC.
P.O. Box 1077
Havertown, PA 19083
(215) 622-5495

SEE COMPUTE!
April 1981
For Eric Rehnke's Review.

OLD PET OWNERS NEVER DIE

☆ . . . . .THEY JUST CALL ON US! ☆

Our innovative line of hardware products protect the investment in your original version 4K/8K PET by extending its life and usefulness. Our software works on most PET/CBMs.

• 2114-TO-6550 RAM ADAPTER HUNDREDS SOLD!

Replaces up to eight 6550 RAMs with low cost, reliable 2114s—one at a time. Use two units for total 6550 replacement. Never worry about RAM availability again!

PHB-001	Bare Board	$11.95
PHK-001S	Full Kit (No 2114s)	$17.95
PH-001S	Assembled (No 2114s)	$24.95

($2.50 postage/handling per order)

• 4K MEMORY EXPANSION

Installs easily internally. Uses eight 2114Ls. Write protect provisions. Second expansion needs external +5V power. "Soft-ROM" applications. Bigger programs!

PHB-002	Bare Board	$16.00
PHK-002	Full Kit (No 2114s)	$29.00
PH-002S	Assembled (No 2114s)	$42.00
PH-002	Assembled with 2114s	$89.00

($3.00 postage/handling per order)

• "REAL WORLD" SOFTWARE

For most PET/CBMs. Word Processor, Mailing List, Catalog, Satellite Tracker, Morse Code Keyer, Frequency Counter. Professional results at low cost!

$17–$25

Write for Free Catalog

PROMPT SHIPMENT!
California Residents, add 6% Tax

SATISFACTION GUARANTEED
Foreign Shipping Higher

OPTIMIZED DATA SYSTEMS

Dept. C, P.O. Box 595 – Placentia, California 92670

DEALER INQUIRIES INVITED

PET/CBM Trademark of Commodore Business Machines by pulling and pushing it to and from the data stack. When a command is given to manipulate data, that data had better be on the stack already. To add 1 and 2, we must say 1 2 + so as to place the two numbers on the stack first. This type of notation is called Reverse Polish (or Postfix) notation and seems clumsy at first; but a little practice will make it seem quite natural.

Values in the data stack are usually limited to 16 bit signed integers. That means that numbers can be integers in the range of -32768 to +32767. You can't fit the value of pi into an integer, nor can you represent your annual salary in pennies unless you're unusually poor. And what about arrays? You might have trouble fitting marks for 300 students on the stack. And, if you made it somehow, you'd have a devil of a time getting the particular one you wanted. And we haven't even mentioned how to deal with strings...

Because of such limitations, FORTH quickly leaves its simple form and starts to demand considerable skills of the programmer. He must be able to allocate memory space and set up sets of indirect pointers that will steer him to the particular unit of data he needs. He will need to be able to handle floating point by building special commands; in many cases this feature is at least partly provided by the vendor.

The beginner is faced with a huge vocabulary of commands, most of which he will need to learn. Not everyone will have the patience to slug through this in order to develop competence in FORTH. When he finds he needs to handle indirect pointers and tables, he'll need to have an aptitude for this kind of thing. FORTH demonstrations can be misleading; the language seems to be so easy when a few simple things are shown.

For those who take the time and trouble to develop FORTH competence, the payoff can be high: fast-running code that can be written quickly. But the beginner must realize that it's not all easy sailing; FORTH won't help you along in the same way that BASIC does.

Advocates of structured programming tend to be suspicious of FORTH. Since FORTH encourages to build upward from the detailed code to the total job, it is considered a "bottom up" type of language. Many computer scientists would prefer to see you go the other way: from the top – the big picture – down into increasing levels of details, or "top down" programming.

It's a language that excites many users. For others, it may be tough sledding and too far off the mainstream of small computer activities. Those who are hooked on FORTH become fanatics: they insist that a job is well done only if it's FORTH right. ©

Part I of this three-part machine language monitor for the OSI Superboard appeared in March, 1982, issue #22.

Part II:

A Superboard II Monitor

Frank Cohen
Pacific Palisades, CA

In the March issue of **COMPUTE!** we presented the first part of a fairly complex program to add a sophisticated "monitor" program to the Superboard II. A monitor does nothing more than to peek into the machine's memory and enter, display, move, or store data in the form of hexadecimal bytes.

Stored in the ROM memory on the Superboard is OSI's monitor program. When originally designing the Superboard, a microcomputer called the KIM-1 was selling well in the microcomputer market. The OSI monitor largely resembles the monitor for the KIM-1. The KIM-1 had a six digit display and a hexadecimal (base 16) keypad plus some other keys which had specific functions devoted to each. With the six digit display, there was room to display a two byte address and the contents of that memory location.

There were two modes of operation: the address mode, and data mode. In the address mode, a key pressed was rotated into the current address being displayed. By rotating the key in, the existing address digits are all shifted left one position (the left-most digit was lost) and the new key pressed is put into the right-most digit. The same kind of scheme is used for entering data in the data mode. However, instead of changing the address digits, the contents of that location are changed.

Changing from the data to the address mode (or vice versa) is accomplished by pressing the AD key, or the DA key. The Superboard II uses the period (.) key instead to enter the address mode and the comma (,) key for the data mode. This system works well for the KIM-1 considering that it cost about $175 and did not have a video display or an advanced keyboard as the Superboard does.

Of course, the monitor program for the Superboard only occupies a small fraction of the space that Super-Monitor uses. However, if you start using your Superboard more and more, you normally will learn how to program in machine language. Possibly blocking your move into the wonderful world of machine language is the resident monitor program.

Last month we outlined what the capabilities of the Superboard II's new monitor program should include. Top on the list was the ability to look into memory, a group of locations at a time. Second, we wanted to be able to modify the Superboard's memory and, at the same time, see what we just modified. Third, we want to fill a block of memory with some value. Next, we want to be able to move a whole block of memory from one location to another. Finally, we'll need an intelligent cassette interface routine for storing and retrieving blocks of memory.

Since Super-Monitor is over 500 bytes long, it has been split into sections. Last month's issue presented the listing for a program called HEXDUMP. HEXDUMP was listed first since most of the other routines in Super-Monitor use its subroutines. When looking at the listings of the individual programs, you will find that they are each mini-programs. The start of each listing also tells what other programs (subroutines) are needed to make it work. The logic behind the listing's structure lies in the fact that loading Super-Monitor in its entirety takes about five minutes with the Superboard's slow cassette interface. By loading just the routines that you want, Super-Monitor can be customized.

HEXDUMP fills the screen of the Superboard with data from memory, eight bytes at a time. HEXDUMP, like most of the routines in Super-Monitor, uses a program called Super-Cursor V1.3 (**COMPUTE!** December, 1981, #19, pp. 124-128) to handle its video output. To use Super-Cursor V1.3, a program puts the ASCII character in the CPU's accumulator and executes a jump-to-subroutine (JSR) to the start address of Super-Cursor, 1E40 (Hex). Super-Cursor also is used to clear the screen, address 1EC2 (Hex), and to home the cursor, address 1E80 (Hex). If you don't want to use Super-Cursor, you will have to write your own video output routine. If you want Super-Cursor and Super Monitor you can send a blank cassette and $3 to the address below and I will copy it for you.

The main subroutines from HEXDUMP that the other routines use are called INADR and PLINE. INADR, starting at address 1D93 (Hex), inputs a two byte address from the keyboard and echoes it to the video screen. The resulting address is stored in address 00E7, called ADR, and 00E8. PLINE is used to print a row of eight bytes of data on the screen. The beginning address is located in ADR, 00E7 and 00E8.

INDATA

The first program in this issue is called INDATA. This program is approximately 199 bytes long and allows the user to look into, and modify, any group of memory locations. Entering machine language programs is simple using INDATA. In fact, after writing HEXDUMP, and Super-Cursor V1.3, I used INDATA to enter the other routines. It is fast and efficient.

INDATA shows the programmer a line of eight bytes of data at a time. Preceding the data is the address of the left-most byte of data. A greater-than sign (>) is placed next to the currently "open" memory location. Any hexadecimal key you hit will be rotated into that byte. When you have finished changing the contents of the current memory location, you can move the greater-than sign to the next location (one space right) by pressing the SPACE bar. Or, you can go back to the last location (one space left) by pressing the RUB-OUT key. If you think that you made a mistake just look up at the screen and compare.

If you are at the right-most byte on a row, the next time you hit the space bar the next line of eight bytes will be displayed. The opposite is true for typing a Rub-Out when you are at the left-most byte. When you are finished entering data, pressing the RETURN key will exit the program. In the listing, when you press the RETURN key, the program will go back into OSI's ROM monitor program.

Program 1 is a complete assembly listing of INDATA. As it is listed, it fits right under HEXDUMP on an 8K Superboard II. I do not suggest trying to move INDATA to another part of memory as it uses many absolute addresses which would have to be modified. However, if you don't have an assembler, it is possible to move it. (This is your encouragement to get a more complex system.) If your Superboard has only the original 4K bytes of RAM, I suggest you add some 2114's.

BMOVE

BMOVE is short for Block Move Routine. As the name implies, this routine is set up to move any size block of memory from one location to another. This is especially handy if you have entered a long program and found that you accidentally started at the wrong location. Another application is looking into the ROM's on your Superboard. By telling BMOVE to move the beginning of the BASIC-in-ROM, located at A000, to the memory mapped video area you can see the internal organs of BASIC.

To use BMOVE, you enter the program at location 1BC6 (Hex). The program first asks you for the starting location of the block to be moved by printing "S=" on the screen. Then it asks you for the ending address by printing "E=" on the screen. (No, it is not asking for Einstein's Theory of Relativity.) Finally, BMOVE prompts you to enter the beginning destination address by printing "D=."

BMOVE is very fast. You will find that it can move a block 8K long in about a second. The majority of BMOVE's program listing is devoted to inputting the three addresses. After it has those addresses loaded, BMOVE calculates the last address of the destination. It then proceeds to move the block, byte by byte, from the top down. For every byte it moves, it will decrement the ending address and check to see if it is equal to the starting address. When the two are equal, it will return to OSI's ROM monitor. Again, later, we will modify the program to return to Super-Monitor's main menu routine.

In the third and final installment, next month, the listings will be described and listed. So far we have enough to call this an advanced monitor routine. The three programs, HEXDUMP, INDATA, and BMOVE, allows you to look at, modify, and move, data in very simple steps.

*These routines make extensive calls to HEXDUMP and SUPER-CURSOR V1.3. It also changes SUPER-CURSOR "system variables," such as cursor position. If you want to use INDATA and BMOVE without HEX-DUMP and SUPER-CURSOR, you will need to refer to the listings of SUPER-CURSOR (***COMPUTE!*** *February, 1982, #21) and HEXDUMP (***COMPUTE!*** *December, 1982, #14). Zero page usage:$E7-$ED*

Program 1: INDATA

```
1C56 20 80 1E A9 41 20 40 1E
1C5E A9 3D 20 40 1E 20 96 1D
1C66 A9 00 85 EA A9 3E 8D 61
1C6E 1F 20 80 1E 20 00 1E AE
1C76 CC D0 20 80 1E 86 E2 20
1C7E FB 1E A6 EA 20 FB 1E 20
1C86 FB 1E 20 FB 1E E0 00 F0
1C8E 04 CA 4C 82 1C A5 E7 38
1C96 E9 08 85 E7 B0 02 C6 E8
1C9E A4 EA B1 E7 85 E9 20 BA
1CA6 FF C9 0D D0 08 A9 A0 8D
1CAE 61 1F 4C 43 FE C9 20 D0
1CB6 03 4C D8 1C C9 7F D0 03
1CBE 4C F8 1C 20 F3 1D 8D CF
1CC6 1C A5 E9 0A 0A 0A 0A 18
1CCE 69 00 85 E9 20 15 1D 4C
1CD6 6F 1C 20 15 1D A5 EA C9
1CDE 07 D0 12 A9 00 85 EA A5
1CE6 E7 18 69 07 85 E7 90 02
1CEE E6 E8 4C 6F 1C E6 EA 4C
1CF6 6F 1C 20 15 1D 98 D0 12
1CFE A9 07 85 EA A5 E7 38 E9
1D06 08 85 E7 B0 02 C6 E8 4C
1D0E 6F 1C C6 EA 4C 6F 1C A4
1D16 EA A5 E9 91 E7 60 AA AA
```

Common routines:

1C56	INDATA	Entry point for INDATA program
1C66	BLOOP	Main loop start for INDATA
1C6F	BPCS	Print a line and fix SUPER-CURSOR bug
1C80	SKIP	Positions cursor to current open cell
1C93	CKSP	Fix HEXDUMP bug by adding $08 to ADR
1C9E	OPCELL	Load BYTE with current open cell
1CA4	KEY	Decodes key pressed and jumps to routine
1CC1	ROTIN	Rotates key pressed value into current cell
1CD8	GNCELL	Open next cell
1D15	CLCELL	Close last cell

Program 2: BMOVE

```
1BC6 20 80 1E A9 53 20 40 1E
1BCE A9 3D 20 40 1E 20 96 1D
1BD6 A5 E7 85 EB A5 E8 85 EC
1BDE 20 95 1E 20 AB 1E A9 45
1BE6 20 40 1E A9 3D 20 40 1E
1BEE 20 96 1D A5 E7 85 E9 A5
1BF6 E8 85 EA 20 95 1E 20 AB
1BFE 1E A9 44 20 40 1E A9 3D
1C06 20 40 1E 20 96 1D A5 E9
1C0E 38 E5 EB 85 ED A5 EA E5
1C16 EC 48 A5 ED 18 65 E7 85
1C1E E7 68 65 E8 85 E8 A0 00
1C26 B1 E9 91 E7 A5 EB C5 E9
1C2E D0 09 A5 EC C5 EA D0 03
1C36 4C 43 FE A5 E9 38 E9 01
1C3E 85 E9 B0 02 C6 EA A5 E7
1C46 38 E9 01 85 E7 B0 02 C6
1C4E E8 4C 24 1C EA EA EA
```

Common routines:

1BC6	BMOVE	Inputs starting location of block to be moved
1BDE	INELOC	Inputs ending location of block to be moved
1BF9	INDADR	Inputs destination address of block to be moved
1B0C	CALC	Calculates ending address of destination block
1C24	MOVIT	Moves a byte from EBAD to DBAD
1C2A	CKFIN	Checks to see if we're finished
1C39	NFIN	Decrements two byte registers EBAD and DBAD

©

OSI TRS-80 COLOR-80 OSI

GALAXIAN - 4K - One of the fastest and finest arcade games ever written for the OSI, this one features rows of hard-hitting evasive dogfighting aliens thirsty for your blood. For those who loved (and tired of) Alien Invaders. Specify system — A bargain at $9.95 OSI

LABYRINTH - 8K - This has a display background similar to MINOS as the action takes place in a realistic maze seen from ground level. This is, however, a real time monster hunt as you track down and shoot mobile monsters on foot. Checking out and testing this one was the most fun I've had in years! — $13.95. OSI

THE AARDVARK JOURNAL

FOR OSI USERS — This is a bi-monthly tutorial journal running only articles about OSI systems. Every issue contains programs customized for OSI, tutorials on how to use and modify the system, and reviews of OSI related products. In the last two years we have run articles like these!

1) A tutorial on Machine Code for BASIC programmers.
2) Complete listings of two word processors for BASIC IN ROM machines.
3) Moving the Directory off track 12.
4) Listings for 20 game programs for the OSI.
5) How to write high speed BASIC — and lots more —

Vol. 1 (1980) 6 back issues - $9.00

Vol. 2 (1981) 4 back issues and subscription for 2 additional issues - $9.00.

ADVENTURES!!!

For OSI, TRS-80, and COLOR-80. These Adventures are written in BASIC, are full featured, fast action, full plotted adventures that take 30-50 hours to play. (Adventures are interactive fantasies. It's like reading a book except that you are the main character as you give the computer commands like "Look in the Coffin" and "Light the torch".)

Adventures require 8K on an OSI and 16K on COLOR-80 and TRS-80. They sell for $14.95 each.

ESCAPE FROM MARS (by Rodger Olsen)
This ADVENTURE takes place on the RED PLANT. You'll have to explore a Martian city and deal with possibly hostile aliens to survive this one. A good first adventure.

PYRAMID (by Rodger Olsen)
This is our most challenging ADVENTURE. It is a treasure hunt in a pyramid full of problems. Exciting and tough!

TREK ADVENTURE (by Bob Retelle)
This one takes place aboard a familiar starship. The crew has left for good reasons - but they forgot to take you, and now you are in deep trouble.

DEATH SHIP (by Rodger Olsen)
Our first and original ADVENTURE, this one takes place aboard a cruise ship - but it ain't the Love Boat.

VAMPIRE CASTLE (by Mike Bassman)
This is a contest between you and old Drac - and it's getting a little dark outside. $14.95 each.

OSI NEW-NEW-NEW OSI
TINY COMPILER

The easy way to speed in your programs. The tiny compiler lets you write and debug your program in Basic and then automatically compiles a Machine Code version that runs from 50-150 times faster. The tiny compiler generates relocatable, native, transportable machine code that can be run on any 6502 system.

It does have some limitations. It is memory hungry — 8K is the minimum sized system that can run the Compiler. It also handles only a limited subset of Basic — about 20 keywords including FOR, NEXT, IF THEN, GOSUB, GOTO, RETURN, END, STOP, USR(X), PEEK, POKE, -, =, *, /, , <,> , Variable names A-Z, and Integer Numbers from 0-64K.

TINY COMPILER is written in Basic. It can be modified and augmented by the user. It comes with a 20 page manual.

TINY COMPILER — $19.95 on tape or disk OSI

SUPERDISK II

This disk contains a new BEXEC* that boots up with a numbered directory and which allows creation, deletion and renaming of files without calling other programs. It also contains a slight modification to BASIC to allow 14 character file names.

The disk contains a disk manager that contains a disk packer, a hex/dec calculator and several other utilities.

It also has a full screen editor (in machine code on C2P/C4)) that makes corrections a snap. We'll also toss in renumbering and program search programs — and sell the whole thing for —

SUPERDISK II $29.95 (5¼") OSI

BARE BOARDS FOR OSI C1P

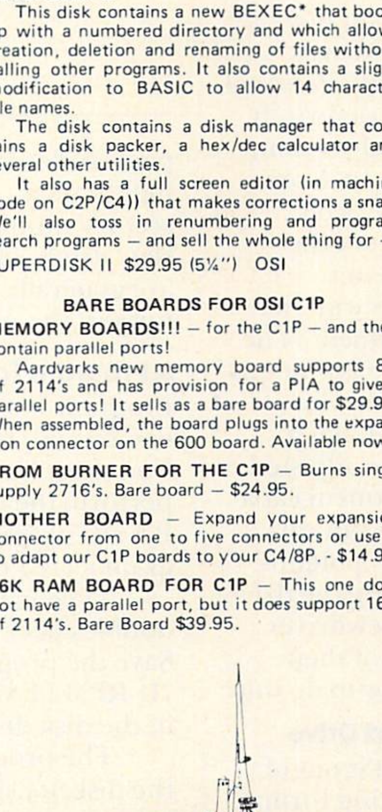
MEMORY BOARDS!!! — for the C1P — and they contain parallel ports!

Aardvarks new memory board supports 8K of 2114's and has provision for a PIA to give a parallel ports! It sells as a bare board for $29.95. When assembled, the board plugs into the expansion connector on the 600 board. Available now!

PROM BURNER FOR THE C1P — Burns single supply 2716's. Bare board — $24.95.

MOTHER BOARD — Expand your expansion connector from one to five connectors or use it to adapt our C1P boards to your C4/8P. - $14.95.

16K RAM BOARD FOR C1P — This one does not have a parallel port, but it does support 16K of 2114's. Bare Board $39.95.



WORD PROCESSING THE EASY WAY— WITH MAXI-PROS

This is a line-oriented word processor designed for the office that doesn't want to send every new girl out for training in how to type a letter.

It has automatic right and left margin justification and lets you vary the width and margins during printing. It has automatic pagination and automatic page numbering. It will print any text single, double or triple spaced and has text centering commands. It will make any number of multiple copies or chain files together to print an entire disk of data at one time.

MAXI-PROS has both global and line edit capability and the polled keyboard versions contain a corrected keyboard routine that make the OSI keyboard decode as a standard typewriter keyboard.

MAXI-PROS also has sophisticated file capabibilities. It can access a file for names and addresses, stop for inputs, and print form letters. It has file merging capabilities so that it can store and combine paragraphs and pages in any order.

Best of all, it is in BASIC (0S65D 51/4" or 8" disk) so that it can be easily adapted to any printer or printing job and so that it can be sold for a measly price.

MAXI-PROS — $39.95. Specify 5¼ or 8" disk.

SUPPORT ROMS FOR BASIC IN ROM MACHINES — C1S/C2S. This ROM adds line edit functions, software selectable scroll windows, bell support, choice of OSI or standard keyboard routines, two callable screen clears, and software support for 32-64 characters per line video. Has one character command to switch model 2 C1P from 24 to 48 character line. When installed in C2 or C4 (C2S) requires installation of additional chip. C1P requires only a jumper change. — $39.95

C1E/C2E similar to above but with extended machine code monitor. — $59.95 OSI

ARCADE GAMES FOR OSI, COLOR-80 AND TRS-80 (8K OSI, 16K TRS-80 AND COLOR-80)

TIMETREK - A REAL TIME, REAL GRAPHICS STARTRECK. See your torpedoes hit and watch your instruments work in real time. No more unrealistic scrolling displays! $14.95.

STARFIGHTER - This one man space war game pits you against spacecruisers, battlewagons, and one man fighters, you have the view from your cockpit window, a real time working instrument panel, and your wits. Another real time goody. $9.95

BATTLEFLEET - This grown up version of Battleship is the toughest thinking game available on OSI or 80 computers. There is no luck involved as you seek out the computers hidden fleet. A topographical toughie. $9.95

QUEST - A NEW IDEA IN ADVENTURE GAMES! Different from all the others, Quest is played on a computer generated mape of Alesia. Your job is to gather men and supplies by combbat, bargaining, exploration of ruins and temples and outright banditry. When your force is strong enough, you attack the Citadel of Moorlock in a life or death battle to the finish. Playable in 2 to 5 hours, this one is different every time.
16K COLOR-80 OR TRS-80 ONLY. $14.95

Please specify system on all orders

This is only a partial listing of what we have to offer. We offer over 120 games, ROMS, and data sheets for OSI systems and many games and utilities for COLOR-80 and TRS-80. Send $1.00 for our catalog.

AARDVARK TECHNICAL SERVICES, LTD.
2352 S. Commerce, Walled Lake, MI 48088
(313) 669-3110

OSI COLOR-80

Getting Your Atari Disk Drive Up To Speed

Bob Christiansen
Vice-President, Quality Software
Reseda, CA

If you have an Atari 810 disk drive that has always worked reliably, then count yourself fortunate. At Quality Software we have spent about half our original investment repairing our Atari drives. Other publishers of software for the Atari Personal Computers have told us of similar experiences. It appears that the 810, at least the original version, was not built to work eight hours a day. In Atari's defense, substantial improvements have been made to the 810 since it first appeared and the reliability of newer models should be better.

One of the most frequent problems with the 810 is that it can get out of speed adjustment. The 810 is supposed to spin diskettes at 288 revolutions per minute (RPM). The hardware, which was actually designed to operate at 300 RPM, has a potentiometer that allows the RPM to be adjusted over a considerable range. Speed adjustment capability is important because many factors can vary the actual speed of the drive. The speed potentiometer is not accessible without removing the drive cover, but anyone who can handle a screwdriver can, with proper care, adjust the speed of their own 810 drive. We will explain later how to do this.

The Symptoms Of An Improperly Tuned Drive

The symptoms exhibited by a drive that is out of adjustment are usually that it starts getting format errors (drive is too fast) or that someone else cannot read a diskette that your drive wrote (drive is too slow). Other reading and writing errors may also occur, but these two symptoms are the most common.

A drive that spins too fast is in danger of improper formatting. It may not write all 18 sectors before completing one revolution. The last sector is written on top of the first, destroying the first. A fast drive will also have trouble writing a sector onto a diskette that was formatted on a slower drive, because it will overwrite the physical space provided on the diskette.

A drive that spins too slowly packs the data so close together that the diskette becomes hard to read, especially by another drive turning at a faster speed. Slower drives can usually read diskettes formatted and written at faster speeds, but the reverse is not true. Thus, if your friend cannot read a disk that you wrote, the most likely event is that his drive is normal and your drive is too slow. If your drive is too slow, you may never know it until you make a diskette and send it to a friend.

The fact that outside tracks have a bigger circumference than inside tracks means that data is packed closer together on the inside tracks (the higher sector numbers). Therefore the higher sectors' numbers are usually the first to fail on a slow drive. These facts about slow and fast drives are generally true about all soft sectored disk drives, not just the Atari 810.

...the 810 ... was not built to work eight hours a day.

A Program To Test RPM

Program 1 is a listing of a BASIC program that you can use to test the speed of your 810 disk drive. You don't have to know how the program works to perform the test, but explanations of both the BASIC program and the machine language program it creates are given later in this article.

Type in the BASIC program, being careful to double check the numbers in the data statements. Save the program by giving it a name like "D:RPMTEST". Then, with any formatted diskette in the disk drive, RUN the program.

The program reads sector one once each time the diskette spins around. It allows the drive to get up to speed, then reads sector one 100 times. This takes about 20 seconds. The elapsed time it takes the drive to turn 100 revolutions is clocked and the RPM is computed and printed out.

At Quality Software, we judge the outcome of the RPM test as follows:

RPM	CONCLUSION
under 285	too slow
285-290	OK, don't adjust
over 290	too fast

The RPMTEST program should be accurate within plus or minus one RPM. If you run the program several times and get results that vary by more than one RPM, it may be due to one or both of the following problems which affect drive

speed:

1. Diskettes that don't turn freely inside their disk jackets significantly slow down the drive. Be sure you make the test with the "loosest" diskette you can find.

2. Drives seem to turn fastest when they are first turned on and slow down slightly as they warm up. We don't know why this is true.

Adjusting Drive Speed

Before you attempt to adjust your drive, we must caution that the operation described here may void any warranty you have for your Atari drive. Even if the drive is out of warranty, Atari does not recommend that users attempt to adjust the speed of their drives. A special speed test diskette and an oscilloscope are necessary, they maintain, to properly set the drive speed.

Neither the author, Quality Software, nor **COMPUTE!** magazine can assume any responsibility for damage caused to your drive while attempting to make a speed adjustment. We do know that hundreds of Atari owners are already adjusting the speed of their drives with no known negative results. Atari does provide an 810 Service Manual for $30 to anyone who insists on doing their own repairs. Write to Atari Personal Computers, 1395 Borregas Avenue, Sunnyvale, CA (attn: Lupe Soto).

If you decide you want to make a speed adjustment, carefully follow the following steps:

1. Be sure you have a clean working environment so that dust, hair, etc. will not get into the disk drive. Turn off the power to the disk drive.

2. You will need a pen knife, a small to medium sized phillips head screwdriver, and a small slot screwdriver.

3. Using the pen knife or similar instrument or strong fingernails, lift off the four plastic stick-on screw hole covers on the top of the drive.

4. Using the phillips screwdriver, remove the four screws that secure the drive cover to the base of the drive.

5. Carefully lift the cover off the drive and set it aside.

6. With the drive facing you, locate the drive speed potentiometer, which is a nylon wheel with a slot in it, in the back of the drive to the left side. We have found two different colors of potentiometers, white and blue.

7. Using the slot screwdriver, turn the tuner in a clockwise direction to slow it down, counterclockwise to speed it up. You will have to experiment until you have arrived at the right speed.

8. Replace the drive cover before testing the drive, but don't screw it back on until you have the drive speed properly adjusted.

The BASIC Program

Looking again at Program 1, we will explain how the BASIC program works. Lines 100 and 110 read in data statements 1000 to 1080, POKEing 73 values into the reserved RAM area starting at memory location 1536 ($600). These 73 bytes comprise a machine language routine that determines the amount of time it takes the drive to rotate 100 times.

Lines 120 and 130 clear the video screen and ask the user to input the drive number. Line 140 pokes the indicated drive number into memory location 1610 ($64A).

Line 150 calls the machine language routine. After returning to the BASIC program, the time it took the drive to rotate 100 times is in memory locations 1611 and 1612 ($64B,$64C). These values are read into A and B by line 160.

The value A is a number ranging from 0 to 255 that is incremented 60 times a second and the value B is a number that is incremented once for every time the value of A reaches 256. Thus line 170 computes the elapsed time in minutes by computing it in sixtieths of a second and dividing by 3600.

Line 180 computes RPM to the nearest unit by dividing 100 (the number of revolutions) by MINUTES. The 0.5 permits proper roundoff. Line 190 displays the result on the video screen.

The Machine Language Program

Program 2 is a listing of the source code for the machine language program and is provided for those familiar with machine language.

The first instruction (line 0010) is required to clean up the stack for reentering BASIC. Lines 0011 to 0014 set the sector number to 1. Lines 0015 to 0017 set up a buffer area at $500 into which the data from sector 1 will be read.

Lines 0018 and 0019 set the drive number to the value specified by the user. Lines 0020 and 0021 specify a read operation. Lines 0022 and 0023 will cause sector 1 to be read five times to ensure that the drive is up to full speed. Lines 0024 and 0026 perform these five reads by calling the resident disk handler ($E453).

Lines 0027 and 0028 request 100 more reads. Lines 0029 and 0031 zero out two of the Atari's timer bytes ($13 and $14). Lines 0032 to 0034 perform 100 reads. Then lines 0035 to 0038 save the elapsed time in ALOC AND BLOC ($64B, $64C). Line 0039 returns control to the BASIC program.

Lines 0040 to 0043 allot space for the variables used by the program.

Program 1.

```
10 REM ********************
20 REM *                  *
30 REM * DISK DRIVE SPEED *
            TEST
40 REM *                  *
50 REM ********************
100 FOR I=0 TO 72:READ A
110 POKE 1536+I,A:NEXT I
120 PRINT CHR$(125);
130 PRINT "DRIVE NUMBER";:
   INPUT DRIVE
140 POKE 1610,DRIVE
150 X=USR(1536)
160 A=PEEK(1611):B=PEEK(16
   12)
170 MINUTES=(256*B+A)/3600
180 RPM=INT(100/MINUTES+0.
   5)
190 PRINT :PRINT :PRINT "R
   PM'S = ";RPM
200 END
1000 DATA 104,169,1,141,10
    ,3,169,0
1010 DATA 141,11,3,141,4,3
    ,169,5
1020 DATA 141,5,3,173,74,6
    ,141,1
1030 DATA 3,169,82,141,2,3
    ,169,5
1040 DATA 141,73,6,32,83,2
    28,206,73
1050 DATA 6,208,248,169,10
    0,141,73,6
1060 DATA 169,0,133,19,133
    ,20,32,83
1070 DATA 228,206,73,6,208
    ,248,165,20
1080 DATA 164,19,141,75,6,
    140,76,6,96
```

TOLL FREE
Subscription
Order Line
800-345-8112
In PA 800-662-2444

Program 2.

```
               0001        PON
               0002 ************
               0003 *
               0004 * TIME 100 DISK
               0005 * REVOLUTIONS
               0006 *
               0007 ************
               0008        ORG   $600
               0009        OBJ   $600
0600 68        0010        PLA
0601 A9 01     0011        LDA   #1
0603 8D 0A 03  0012        STA   $30A
0606 A9 00     0013        LDA   #0
0608 8D 0B 03  0014        STA   $30B
060B 8D 04 03  0015        STA   $304
060E A9 05     0016        LDA   #5
0610 8D 05 03  0017        STA   $305
0613 AD 4A 06  0018        LDA   DRIVE
0616 8D 01 03  0019        STA   $301
0619 A9 52     0020        LDA   #'R
061B 8D 02 03  0021        STA   $302
061E A9 05     0022        LDA   #5
0620 8D 49 06  0023        STA   NREAD
0623 20 53 E4  0024 LOOP   JSR   $E453
0626 CE 49 06  0025        DEC   NREAD
0629 D0 F8     0026        BNE   LOOP
062B A9 64     0027        LDA   #100
062D 8D 49 06  0028        STA   NREAD
0630 A9 00     0029        LDA   #0
0632 85 13     0030        STA   $13
0634 85 14     0031        STA   $14
0636 20 53 E4  0032 LOOPA  JSR   $E453
0639 CE 49 06  0033        DEC   NREAD
063C D0 F8     0034        BNE   LOOPA
063E A5 14     0035        LDA   $14
0640 A4 13     0036        LDY   $13
0642 8D 4B 06  0037        STA   ALOC
0645 8C 4C 06  0038        STY   BLOC
0648 60        0039        RTS
               0040 NREAD  DS    1
               0041 DRIVE  DS    1
               0042 ALOC   DS    1
               0043 BLOC   DS    1

SYMBOL TABLE
  LOOP   0623   LOOPA 0636   NREAD 0649
  DRIVE  064A   ALOC  064B   BLOC  064C
```

©

It is not immediately obvious, but if you know machine language and want to change some aspect of your ROM, you could move the code down into RAM and adjust the JSR's, JMP's, etc. to reflect the new locations. Then, you can modify the previously ROM-frozen program all you want. Here, the Apple II Plus Monitor is augmented by adding Step and Trace functions after moving it into RAM.

Softmon: Restoring Trace And Step To The Apple II Plus Monitor

R. Hiatt
Brock University
St. Catharines, Canada

A monitor trace is a useful tool for debugging machine code programs, in the same way that TRACE works for BASIC. In addition, it is invaluable for deciphering commercial binary routines which tend to be convoluted and to rewrite parts of themselves so that a disassembled listing taken at one point of a run will differ from that taken at another point.

The old Apple II *had* monitor trace, and the Apple II Handbook gives a disassembled listing of Steve Wozniak's monitor, resident at \$F800-\$FFFF, (as is the monitor of the current II Plus). Comparison of the new with old monitor shows relatively few differences, only 200 or so bytes of code. In particular, the subroutine STEP (\$FA43-FAD6, \$FAFD-FB1D) has been changed so that keyboard entry of 'AdrT' or 'AdrS' <CR> simply does an RTS.

The II-Plus monitor, being in ROM, can't be changed, of course, (and perhaps that's just as well). But it can be copied. In fact, it will obligingly copy itself. The procedure is as follows:

Key in:

1. CALL-151 <CR> (puts you into monitor)

2. 4800<F800.FFFFM <CR> (moves, *i.e.*, copies the code from the second address through the third address to a location starting at the first address.)

The move is so fast that it's hard to believe anything has happened, but you really do have a copy of the monitor in RAM, residing at \$4800-4FFF, and *this* can be changed in any way you want. Moreover, it already works. Try it; leave monitor with RESET and execute a CALL 20329. The familiar star appears on the screen and you're in *soft monitor*.

The soft monitor appears to do everything that the resident monitor does. In fact, the resident monitor is still doing all the work. The only part of the soft monitor being used is the directory (now at \$4F65-4FFF) which still calls subroutines in the \$F800-FFFF area. Some changes have to be made. (If you're making changes on your own, it's expedient to make one change right away, and that's the prompt character. For example replace the \$AA at \$4F6A with \$A3. This will produce a "#" instead of a "*". Without this, it's awfully easy to get back into resident monitor without knowing it.)

The changes I was interested in making are done by the BASIC program, Makesoftmon (Program 1). In fact, the program does the whole job, including the original monitor copy, since that subroutine can be called from BASIC after POKEing the appropriate addresses (lines 110-130). A simple loop (lines 160-190) detects the JSR's and JMP's that transfer control within monitor, and changes the high byte of the address from \$Fx to \$4x. A few discrete POKEs are necessary (lines 210-220) to change the high bytes of some indirect addresses and to restore the character table to old monitor form. Finally, the STEP subroutine is read from DATA statements and POKEd.

That completes Softmon. For the user who doesn't want the soft monitor at \$4800-, but somewhere else, Makesoftmon is easy to change. It's just a matter of replacing all \$4x high bytes with the high byte address desired.

Obtaining the STEP and TRACE functions via Softmon carries a fairly high overhead. Ideally, Softmon should call the resident monitor for those routines it handles and carry code only for those things it won't. But that's another project. At the moment, I'm more interested in unravelling some other mysteries using the newly acquired functions.

To use monitor TRACE and STEP with Softmon in memory (\$4800-\$4FFF), either CALL 20329 or CALL-151 and then execute 4F69G.

To execute a *trace,* key in the desired start address (in hex, of course), followed by "T," and then carriage return (CR). For STEP, key in start address followed by "S," CR. To go on to the next instruction, simply key "S," CR. Both trace and step display the disassembled instruction plus the contents of the A, X, Y, S and P registers, and most importantly, *perform* the instruction, so that branches are followed properly and new code is created.

A certain amount of discretion is involved with these. Trace scrolls the information up the CRT about twice as fast as a BASIC LIST. To see what it's doing in detail really requires a printer, and even then it's very easy to go through a ream of paper needlessly if trace encounters loops of any length. I'd be inclined to say the best use of trace would be in debugging your own programs on CRT only, to the extent of insuring that there are no infinite loops, and that the final RTS is correct.

Step has much more potential, the pace is user-controlled. If trapped in a long loop, one can simply step out by keying a step to the alternate branch. Better yet, rather than stepping out explicitly, monitor in the information that will cause the desired branch and then step the location again. This will ensure that subsequent step-traces will be getting correct code.

A cautionary note: since trace and step do perform the instructions, the same caution has to be observed as with any POKE or user-written machine code routine. Switches will be set or reset, *no* permanent damage can be done, but the system can be crashed or DOS can be disabled.

```
10  REM  MAKE SOFTMON
20 D$ =  CHR$ (13) +  CHR$ (4)
30  DEF  FN LB(D) = D - 256 *  INT (D / 256)
40  DEF  FN HB(D) =  INT (D / 256)
50  DEF  FN DC(I) =  PEEK (I) + 256 *  PEEK (I + 1)
60  REM  PAGE ADDRESSES IN $4800-4FFF BY HEX DIGIT
70 S8 = 18432:S9 = 18688:SA = 18944:SB = 19200:SC = 19456:SD = 19712:SE =
   19968:SF = 20224:SG = 20480
80  REM  A1,A2,A4 HIGH & LOW BYTES
90 L1 = 60:H1 = 61:L2 = 62:H2 = 63:L4 = 66:H4 = 67
100 MS = 63488:ME = 65535
110  PRINT "COPYING MONITOR INTO $4800-$4FFF"
120  POKE L1, FN LB(MS): POKE H1, FN HB(MS): POKE L2, FN LB(ME): POKE H2,
     FN HB(ME): POKE L4, FN LB(S8): POKE H4, FN HB(S8)
130  CALL 65068
140  PRINT "CHANGING ADDRESSES FOR JSR'S AND JMP'S"
150  PRINT "CHANGED ADDRESSES ARE LISTED"
160  FOR I = S8 TO SG - 1:P =  PEEK (I): IF P <  > 32 AND P <  > 76 THEN
    190
170 P =  PEEK (I + 2): IF P < 248 THEN 190
180  POKE I + 2,P - 176: PRINT I
190  NEXT
200  PRINT "MAKING A FEW OTHER ADDRESS CHANGES"
210  POKE SA + 235,74: POKE SE + 168,77: POKE SF + 127,79: POKE SF + 191,
    78: POKE SF + 195,79: POKE SF + 207,237
220  POKE SF + 210,236: POKE SF + 233,195: POKE SF + 253,79: POKE SF + 25
    5,74: POKE SF + 106,163
230  PRINT "READING NEW DATA AND POKING"
240  FOR I = SA + 64 TO SA + 214: READ P: POKE I,P: NEXT
250  FOR I = SA + 253 TO SB + 29: READ P: POKE I,P: NEXT
260  FOR I = SE + 194 TO SE + 201: READ P: POKE I,P: NEXT
270  PRINT "NEWMON IS NOW READY": PRINT : PRINT "TO ENTER IT, CALL 20329"
    : PRINT "OR"
280  PRINT "ENTER MONITOR WITH CALL -151": PRINT "AND THEN KEY '4F69G'"
290  INPUT "SAVE SOFTMON ? ";Q$: IF Q$ >  = "Y" THEN  PRINT "SAVING SOFTM
    ON,A$4800,L$800": PRINT D$"BSAVE SOFTMON,A$4800,L$800"
300  END
310  DATA  255,255,255,32,208,72,104,133,44,104,133,45,162,8,189,16
```

```
320  DATA  75,149,60,202,208,248,161,58,
     240,66,164,47,201,32,240,89
330  DATA  201,96,240,69,201,76,240,92,
     201,108,240,89,201,64,240,53
340  DATA  41,31,73,20,201,4,240,2,177,
     58,153,60,0,136,16,248
350  DATA  32,63,79,76,60,0,133,69,104,
     72,10,10,10,48,3,108
360  DATA  254,3,40,32,76,79,104,133,58,
     104,133,59,32,130,72,32
370  DATA  218,74,76,101,79,24,104,133,
     72,104,133,58,104,133,59,165
380  DATA  47,32,86,73,132,59,24,144,20,
     24,32,84,73,170,152,72
390  DATA  138,72,160,2,24,177,58,170,
     136,177,58,134,59,133,58,176
400  DATA  243,165,45,72,165,44,72
410  DATA  24,160,1,177,58,32,86,73,133,
     58,152,56,176,162,32,74
420  DATA  79,56,176,158,234,234,76,11,
     75,76,253,74,193,216,217,208
430  DATA  211,198,52,32,117,78,76,67,74
```

©

Did you ever type NEW and regret it? In theory, your program in memory is wiped out by the word NEW. On the Atari, memory is, in fact, washed clean with zeros. But the Apple and PET/CBM computers just reset some pointers (the program is still in there). Here's how you un-new.

Recovering From NEW On Apple And CBM

Jimmy Stephens
Athens, GA

If you are like me, at least once in your life you have had that sickening feeling that comes with the realization that you have just typed NEW, or maybe turned your computer off, without saving your program. The next thing you probably did was to look for the nearest window to jump out. Unfortunately, if you have just turned off the computer, I can't give you any alternative to jumping out the window. (Hopefully it's a *text window.*) What I can do is to show you how to combat an accidental NEW.

When the NEW command is processed by Apple and CBM computers, the program in memory is not erased. All that happens is that several pointers are reset so that, when the next program is entered, it will write over the old one. If you have entered NEW and have not written any lines to a new program or loaded another program, then your old program is still in memory and can be recovered with a little work.

On both Apple and CBM, NEW destroys only two bytes of the actual program. Fortunately, these two bytes deal with the length of the first line of the program and can be recovered. The first POKE to be made is POKE 2050,8 on the Apple or POKE 1026,4 on CBM. Now, try LISTing the program. Surprise! You should see the first line of the program on the screen.

Your next POKE will be determined by the number of bytes in this first line. Count the number of characters in the line, remembering that reserved BASIC commands are tokenized and take up only one byte. Therefore, count all BASIC commands as one character. Add six to your total and POKE this value into location 2049 on the Apple or location 1025 on CBM.

There is a good chance, because of spaces in the line and such, that this POKE will not be correct the first time. LIST the program again. If it LISTs O.K. then the POKE was right and you can skip to the next paragraph. Otherwise, you will need to try a new value and redo the POKE. Keep trying values around your original total until the program LISTs correctly.

At this point, although it looks alright, *do not run the program.* If you do, the variables will overwrite the program and all your work will have been for nothing. You will have to make POKEs in a pair of locations to reset the pointer for the beginning of the variables. For CBM computers, these locations depend on which ROM revision your system has. For Original ROMs, these locations are 124 and 125, and, for both Upgrade and 4.0, ROMs the locations are 42 and 43. The Apple has a pointer to the start of variable space as well as a pointer to the end of the program. This means that if you have an Apple, you will have to make all remaining POKEs in both of these locations. The first pointer is at 105 and 106 and the second one resides at 175 and 176.

The next POKEs are designed to give your program some room to breathe and are determined by the amount of RAM you have. These POKEs are only temporary and will be changed later. Use the formula below to determine the value of the first POKE.

((Number of Kilobytes of RAM) * 4) -2

This POKE is made in the high byte of the variable pointer(s). Next, POKE 0 into the low byte of the pointer(s). For example: On a 16K Apple, the operation would look like this

```
POKE 105,0 : POKE 106,62 : POKE 175,0 :
POKE 176,62
```

Your program will now run, but, if it uses lots of variables, you will soon get an OUT OF MEMORY message. To fix this, add the following lines to your program:

```
0 I=XXXX
1 I=I+1:IFPEEK(I)<>81THEN1
2 IFPEEK(I+1)<>81THEN1
3 I=I+6:A=INT(I/256):B=I-256*A
4 POKE YY,B:POKE ZZ,A:END
63999 QQ
```

XXXX = 1024 for CBM or 2048 for Apple.
YY = low byte of variable pointer(s).
ZZ = high byte of variable pointer(s). (Two more POKEs for the second pointer will be necessary on the Apple.

When the program is run, it will change the POKEs to the correct value. Obviously, if your program's first line number is less than five or if it has a line numbered 63999, you will have to make some adjustments. Also, you will need to make sure that "QQ" does not appear anywhere else in the program. If it does, you will have to use another two-letter sequence in line 63999 and you will need to adjust the ASCII codes in lines one and two accordingly. ©

There are times when you will write a program that is so large that it will need every last memory cell in your computer. This article shows several ways to reduce the size of your programs. In general, these techniques apply to any computer using BASIC, not just to the VIC-20.

Putting The Squeeze On Your VIC20: Getting The Most Out Of 5000 Bytes

Stanley M. Berlin
Dallas, TX

Five-thousand of almost anything seems like a lot; a Christmas bonus of $5000 would make anyone happy; 5000 jelly beans would be more than even our President could eat; 5000 days is over fourteen years. However, there are other circumstances when a quantity of five-thousand really is not so much. Five-thousand raindrops would probably go unnoticed. The time when five-thousand is a really small quantity is when you are writing a program and have only five-thousand bytes (memory locations) in which to do the job!

I remember working twenty years ago with a 4000 character IBM 1401 Computer and feeling confident that if I had only another twenty-five memory locations I would be able to complete the program. Times have changed during the last twenty years and there are many programmers who now work with *virtual memory systems. [Where the computer can use disk memory as if it were RAM — Ed.]* where there is no such problem as being constrained by the amount of memory. With technology moving as fast as it is, persons working on small microcomputers probably will not have to wait twenty years for a virtual memory-like system. When that time arrives, people will not have to write articles like this one. However, today, if you are writing programs for the Commodore VIC-20 you will have to live with the constraint of having only 5,000 bytes worth of memory in which to do your work.

Anyone who has done any serious programming for the VIC20 knows that it does not take many BASIC statements before you get that dreaded "OUT OF MEMORY" message. Of course, the VIC20 is nice enough to let you know when you turn it on that by the time it gets through allocating 506 bytes for the video mapping, and another 506 bytes for color mapping on the screen, and reserves memory space for such other things such as tape cassette buffers, you have only 3583 bytes of memory in which to store your program.

So, there you are busily entering your new BASIC program and VIC sends you the "OUT OF MEMORY" message. What are your options? You can resign yourself to the fact that, no matter what you do, the program will never fit into memory and abandon your project. Surely, no programmer worth his or her salt would exercise this option! Another option is to run out and purchase a memory expansion unit. This is not too bad a solution except, at the time this is being written, there is no such item available. Even if there were, it is surely a costly solution. The last option is to roll up your sleeves and dig into your program with a finely honed scalpel to perform surgery on it. That is certainly the most challenging option, and it is the purpose of this article to pass on a few points to help you in your efforts.

The items discussed will be from a BASIC programmer's point of view. Technical system information will be avoided except when it is necessary for a clear understanding of the issue. Although these suggestions are aimed at the VIC20 owner and the VIC20 is the computer used to validate the data, much of the information is pertinent to other computers using Microsoft BASIC.

One last point prior to moving to the meat of this article; many of the suggestions presented here are a tradeoff between good program documentation and the amount of memory used. Remark (REM) statements liberally scattered throughout a program provide the roadmap when you are

trying to debug a program.

At this time, most VIC20 owners probably do not have the benefit of a printer, so there is no printed copy of the program on which to make comments regarding the various routines used in a program. If you are going to do any serious programming, a printer makes the task much easier because you can follow the logic and flow of a program from beginning to end without having to enter multiple LIST statements. A disk drive provides a lot of speed and flexibility when you are using a program, but a printer is worth its weight in gold when you are trying to debug a program. If it is necessary to remove REMark statements from a program in order to conserve memory space, it is worthwhile keeping some handwritten notes concerning the program. At a minimum, you should write down the BASIC line numbers of subroutines and major sections of the program so that you will at least have an idea of what area of the program to LIST when you want to look at or change an area of code.

REMs And Blanks

That brings us back to being confronted with the "OUT OF MEMORY" message and the first technique for buying back a few bytes of storage. REMark statements, as important as they are, require memory. They do not provide any function in your program. A REMark statement on a line by itself will require a minimum of six bytes, even if there is no text associated with the REMark. If the REMark contains textual information (as it usually does), add the length of the text to that six bytes.

The quickest way to obtain memory is simply to remove the REMark statements. Remember to write down a notation about the REMark so that the information is at least externally preserved for documentation purposes. One word of caution: if your program contains GOTO or GOSUB statements whose object is a line number containing a REMark that you removed, you will receive an "UNDEFINED STATEMENT" error message. Should you get an "UNDEFINED STATEMENT" error message you will have to figure out from where the REMark was removed and change the GOTO line number to be the line following the original REMark; this can be a long and tedious task if your program contains many statements that GOTO a REMark statement. Try to avoid this situation when you are originally writing your program by not coding GOTO or GOSUB statements which land on a line number containing a REMark.

Another way to buy back a few bytes of memory at the expense of good "internal" documentation is to remove all unnecessary blanks from the program. This makes the program a bit harder to read, but every blank removed is a byte of usable memory. That does not seem to be much, but if you add up the number of unnecessary blank spaces in your program, you will be surprised; besides, no one ever said that this was going to be easy!

The VIC20 makes it easy to remove the blanks with the use of the INST/DEL key, but remember not to remove any blanks from within quotes. Data between quote marks are called *strings* and, if you are displaying them on a television or printer, you undoubtedly need those blanks. For example, if you are displaying the message "PRESS X TO EXIT" you would not want that information displayed as "PRESSXTOEXIT".

The following routine was entered on the VIC20:

```
NEW
100 PRINT "clr"
200 A=2:B=3:C=4:D=5
300 IF A = B AND C = D THEN D = D + 1: GOTO
    400
400 PRINT "FREE=";FRE(X)
```

The results of running this program showed that there were 3465 bytes available; the program occupied 118 bytes (3583 - 3465 = 118). By simply removing the fifteen blanks in statement 300 it became less readable:

```
300 IFA=BANDC=DTHEND=D+1:GOTO400
```

but the results of that run showed 3480 bytes of free space; a 100% return for each blank removed. Finally, you might have observed that I did not remove the blank that separates the line number from the statement. It does not really occupy any memory and is there only for readability; in fact, if you remove that blank, you will find that BASIC will reinsert it when you LIST the statement.

Multiple Statements And Short Variable Names

Closely allied with removing blanks and removing REMark statements is putting multiple statements on a line. Your VIC20 can only display 22 characters on a line, but BASIC will actually accept up to 80 characters. It is possible to have your BASIC statements occupy about three and one half display lines. It is also possible to combine statements on one BASIC line by using the colon separator character. Every line number in your program contains an overhead of five bytes (for technicians: that five bytes consists of two bytes for the line number, two bytes for an internal pointer, and one byte for a delimiter at the end of each statement). You can save four of these five bytes for every statement combined on a line (the colon separator will use one of the five bytes eliminated). For example, instead of coding:

```
100 A=A+1
200 IF A>25 THEN Z% = 0
```

you can save four out of the five byte overhead of the second line by coding:

```
100 A=A+1:IF A>25 THEN Z% = 0
```

However, no suggestions are free of charge and there is also something to watch out for in this instance. You may freely combine statements for up to eighty characters, but it is possible that one of the statements you are combining might be the object of a GOTO or GOSUB statement, in which case you will receive the "UNDEFINED STATEMENT" error message. In the example immediately above, if there were another statement in the program which was "GOTO 200", line number 200 would not be in the program after combining the two lines and you would get the error message. If you had a statement that was "GOTO 100", that would not cause any problems.

Another item to watch for when combining statements is not to combine a line with a preceding line that contains an IF statement. The statement shown above is alright. However, if the two lines were reversed:

```
100 IF A>25 THEN Z% = 0
200 A=A+1
```

you would not be able to combine the lines as:

```
100 IF A>25 THEN Z%=0:A=A+1
```

without altering the meaning of the statement. In this instance, the addition statement would only be executed if A were greater than twenty-five which is not the intent of the original.

The last item that returns a little usable memory at the expense of readability and documentation is the use of short variable names. Although variable names may be up to 255 characters, BASIC uses only the first two characters (plus the $ and % suffixes for string and integer variables respectively). Each character in the variable name occupies a byte wherever it is used; therefore you should limit variable names to two characters and one character would be even more thrifty from a memory-use point of view. Limiting the names to two characters could have a side benefit inasmuch it may eliminate a potential source of programming error. If you had two variables, one named "TAPE" and the other named "TASTE," BASIC would only recognize "TA" as the name and would, in effect, be dealing with a single variable.

Avoiding a technical discussion as to why it is so, it is usually more economical to use constants instead of variables whenever possible. A constant consists of data stored in the BASIC statement itself. Constants may require less memory than variables, especially in cases where the constant is a relatively short string. As the length of the string increases, the amount of savings diminishes because the repetition of the constant also occupies memory.

The following lines each contain a constant:

```
100 A = A + 1
     ("1" is the constant)
100 S$ = D$ + "SUFFIX"
     ("SUFFIX" is the constant)
100 PRINT "TOTAL=";X
     ("TOTAL=" is the constant)
```

An illustration of the savings that can be gained is:

```
100 T$="THIS IS A TEST"
200 PRINT T$
300 PRINT T$
```

occupies 72 bytes of memory, whereas

```
100 PRINT "THIS IS A TEST"
200 PRINT "THIS IS A TEST"
```

only occupies 69 bytes of memory. That is not much of a savings because the string "THIS IS A TEST" is relatively long; if it were shorter, the savings would be more dramatic. The reason for this is that, when data is assigned to a variable, it requires two areas of memory, but a constant requires only one (in the instruction itself).

BASIC is sometimes very shrewd as far as memory management is concerned. BASIC is smart enough to know when you have used a string and will reuse it rather than recreating it again in memory. Thus, if you have coded the statement: PRINT "THIS IS A TEST" and, elsewhere in your program you coded A$="THIS IS A TEST", although memory would be required to contain pointers for the variable "A$", the actual text string "THIS IS A TEST" would not be recreated in memory (except in the instruction itself). The original text string would be pointed to by the variable. This is starting to border on the kind of technical information that this article has tried to avoid, but is interesting enough to pass on.

Don't Avoid Integer Variables

It seems that most people writing BASIC programs never bother with integer variables and, yet, that is where a significant savings in memory can be attained. This is particularly true if the program contains arrays. Consider that, for each element in an array, the number of bytes occupied is as follows:

A string array = 3 bytes plus the length of the string per element
A floating point array = 5 bytes per element.
An integer array = 2 bytes per element.

The contents of many arrays do not require the use of decimal points, but it is easier to code "DIM A(15)" rather than "DIM A%(15)." By using the integer form, you would save 45 bytes of memory. Suppose you were writing a program to deal a deck of cards and you defined an array to keep track of which cards have already been dealt. That fifty-two element array could be a string array, a floating point array, or an integer array. Obviously, there is no need for decimals in this example so the obvious choice would be to use integers. The following program was run three times on the VIC20; each time changing the type of array (making the variable type in statement 400 correspond to the array).

```
100 PRINT "clr"
200 DIM X(100)             <==== First run
    DIM X%(100)            <==== Second run
    DIM X$(100)            <==== Third run
300 FOR Z = 0 TO 99
400 X(Z) = Z               <==== First run
    X%(Z) = Z              <==== Second run
    X$(Z) = CHR$(Z)        <==== Third run
500 NEXT Z
600 PRINT FRE(X)
```

The differences in memory use during these three runs is very dramatic:

The first run used a floating point array and occupied 601 bytes; the second run used the string array and occupied 403 bytes; and the final run, which used an integer array, occupied only 300 bytes. Is it worth the cost of 300 bytes in order to save typing in a "%" each time the variable is used?

Each of the memory conserving measures outlined so far would be relatively easy to implement using the editing capabilities of the VIC20 once a program is written and resident in memory. The last few suggestions are harder to implement and the savings are more indefinite.

If your program contains groupings of instructions that are repeated several times, it would be more memory efficient (and better programming practice) to incorporate those instructions once as a subroutine and GOSUB to them. If such statements are readily identifiable, you can implement this fairly easily with the following three steps:

1. Add a RETURN statement after the first group of statements.
2. Place a GOSUB statement whose object line number is the first line number in the group immediately preceding the group.
3. Add a GOTO statement immediately after the inserted GOSUB statement whose object line number is the statement immediately following the RETURN statement.

Naturally you would delete all other occurrences of the same group of statements and replace them with a GOSUB to the newly created subroutine. This sounds very complicated, but actually is quite easy to implement and is illustrated in the fictitious routine that follows. Assume that the statements starting with line number 650 and ending with line number 710 are repeated several times in the program.

```
640 PRINT "ABC"
650 A = A + 1
660 IF A = 9 THEN 700
670 PRINT "MESSAGE ONE"
680 MC = MC / A
690 GOTO _____
700 PRINT "MESSAGE TWO"
710 MC = MC * A
720 IF Q + 1 = 10
```

You can convert this to a subroutine using the technique described by adding lines 645 and 715 in the following illustration:

```
640 PRINT "ABC"
645 GOSUB 650: GOTO 720  <==== New statement
650 A = A + 1
660 IF A = 9 THEN 700
670 PRINT "MESSAGE ONE"
680 MC = MC / A
690 GOTO _____
700 PRINT "MESSAGE TWO"
710 MC = MC * A
715 RETURN  <==== New statement
720 IF Q + 1 = 10
```

Another almost obvious way to decrease the amount of storage your program uses is simply to reduce the size of messages that you display on the television. For example, if your program displays the word "TOTAL", change it to "TOT". If your program contains cards, instead of spelling out "KING," "QUEEN," and "JACK," simply use a "K," "Q," and "J" respectively.

Another item to investigate is the use of more economical instructions to achieve the same results. Shown below is an example of how you can replace multiple IF statements with a single ON statement. The program does the exact same thing either way you write it, but using the ON statement yields a savings of 62 bytes.

The following statements

```
300 IF A=1 THEN 510
310 IF A=2 THEN 550
320 IF A=3 THEN 600
330 IF A=4 THEN 650
340 IF A=5 THEN 700
350 IF A=6 THEN 750
```

could be replaced with the single statement:

```
300 ON A GOTO 510, 550, 600, 650, 700, 750
```

Consider using FOR/NEXT loops instead of repeating instructions wherever possible. Suppose you wanted to print a vertical line down the center of the screen, you could program it as:

```
300 PRINT TAB (10);"|"
```

```
310 PRINT TAB (10);"I"
320 PRINT TAB (10);"I"
330 PRINT TAB (10);"I"
340 PRINT TAB (10);"I"
350 PRINT TAB (10);"I"
360 PRINT TAB (10);"I"
370 PRINT TAB (10);"I"
380 PRINT TAB (10);"I"
390 PRINT TAB (10);"I"
400 PRINT TAB (10);"I"
```

or alternatively you could code:

```
300 FOR X = 1 TO 11
310 PRINT TAB (10);"I"
320 NEXT X
```

Again, the same results are achieved, but the FOR/NEXT loop yields a savings of 118 bytes!

Overlaying

You can sometimes conserve memory by *overlays*. If your program runs in two separate and distinct phases (that is, one portion of your program completes all its work and then is never executed again) it should be possible to split your program into two sections. Have the last statement in the first section issue the statement "LOAD PHASEII" (assuming your second phase is named "PHASEII"). When the first section completes its job, the last instruction would load the next phase for execution.

This assumes that you have written PHASEII onto the cassette tape immediately after PHASEI. This is called *overlaying;* it lends itself well to a disk-oriented system, but there is no reason not to use it with tape also. You should be aware that any variables used in the first phase will not be available in the second phase. (However, even though this should work, the author has not yet been able to do this successfully.

If you still need memory, you may be able to put some of the data used in your program on a cassette tape and read the data in during program execution. This technique will involve your writing a special program to create the data tape, but, in some instances this can yield substantial memory savings. The premier issue of *Home and Educational COMPUTING!* contained an excellent article by David Malmberg entitled "Custom Characters for the VIC." A program shown in that article lends itself very well to illustrating this memory-saving technique.

That program contained a number of DATA statements and is shown below:

```
170 READ X:IF X > 0 THEN 190
180 FOR X = X TO X+7
182 READ J
184 POKE I,J
186 NEXT
190 GOTO 170
.
.
.
340 DATA 7168,24,24,36,60,102,66,66,0
350 DATA 7176,124,34,34,60,34,34,124,0
360 DATA 7184,126,34,34,32,32,32,112,0
.
.
.
600 DATA 7376,0,0,0,0,0,0,0,0
610 DATA -1
```

You would have to write a program to write the data to cassette tape (and ideally you would write the data immediately after the program you saved that will be using that data). The original program can be easily modified to create the data tape and an example could be:

```
100 OPEN 1,1,2,"DATATAPE"
170 READ X:PRINT#1,X:IF X > 0 THEN 190
180 FOR X = X TO X+7
182 READ J: PRINT#1,J
186 NEXT
190 CLOSE1
200 PRINT "DATA TAPE CREATED"
210 END
340 DATA 7168,24,24,36,60,102,66,66,0
350 DATA 7176,124,34,34,60,34,34,124,0
360 DATA 7184,126,34,34,32,32,32,112,0
.
.
.
600 DATA 7376,0,0,0,0,0,0,0,0
610 DATA -1
```

You would then substitute INPUT#1 statements for READ statements in the original program, but the results would be the same. If you wanted to use the concept in Mr. Malmberg's article in your own program, but needed additional memory, this technique will provide you with a significant amount of memory.

The modified program would be:

```
165 OPEN 1,1,0,"DATATAPE"
170 INPUT#1, X:IF X > 0 THEN 190
180 FOR X = TO X+7
182 INPUT#1,J
184 POKE I,J
186 NEXT
190 GOTO 170
.
.
.
340 REM -NO DATA STATEMENTS-RESULTS IN
350 REM -A SIGNIFICANT REDUCTION IN
         MEMORY
```

Let's assume that you have exercised all the memory-saving procedures outlined and your program still requires additional memory. The last thing to do is carefully investigate the logic of your program. Are there statements that are never executed (perhaps left over from an initial idea that was abandoned)? Naturally, you should remove them. Are there better ways to implement a procedure that might reduce the number of instructions necessary to accomplish some objective? Sometimes being able to buy back just five or ten bytes of memory will allow your program to run.

One final point; there are instances when you have worked for hours on a large, complicated program and you decide to save it on a cassette tape. You type in the command "SAVE MY-PROGRAM", press ENTER, and lo and behold, instead of receiving the message to press the play and record buttons, you instead get the message "OUT OF MEMORY." This is one of the most frustrating events possible when writing a program because you do not want to lose the hours of effort already expended. Try issuing the "SAVE" command with a shorter filename; this usually works. Instead of entering "SAVE MY-PROGRAM", use "SAVE A".

It takes some effort, but using the techniques outlined here could mean the difference between being able to do what you want with your VIC20 or being continuously confronted with an out of memory condition. With a little patience you might be able to find that needle in the haystack. If all else fails, don't give up hope. Remember that your VIC20 has the capability of being expanded to 32K! ©

Considering Space And Time In The Atari

C. Michael Levy and Grant Levy
Gainesville, FL

Would you like to know how to speed up some of your programs, or make them use up less memory? As everyone comes to learn as they gain experience in programming, small and often very subtle changes in even very simple programs can have major consequences. Consider the following four programs. Each has exactly 1000 lines. Each makes two references to variable B, 994 assignments to variable A, contain six GOSUBs and six RETURNs, one POKE, one PEEK, one PRINT and one STOP. They all achieve the same end: they determine that the value of variable A is zero and print the number of clock ticks (each lasting 1/60th sec.) required for execution of the program. They are equivalent, right? *Wrong!*

Programs 1 and 2 each require about 15000 bytes of memory, nearly 66% more than the approximately 9100 bytes required for Programs 3 and 4. The reason for this is the way Atari BASIC handles constants and variables. Each and every reference to a constant (zero, in these programs) consumes seven bytes. This is true even if the statements using those constants are *never* executed in the program (as are lines 2-994 in Program 1 and lines 8-1000 in Program 2). Programs 3 and 4 involve the assignment of a constant to only *one* variable (B in line 1); variable A assumes its values by reference to this value. Each such reference requires only one byte.

So the first lesson is to use constants sparingly, if at all. Develop a schema for assigning variable names to constants so that you will not confuse them with "real" variables. For example, consider D0 = 0, D1 = 1, or C0 = 0, C1 = 1, etc. where the symbol "D" is a mneumonic for *digit* or "C" reminds you that it is a *constant.* Any other combination of letters and/or digits could then represent variables which *vary.*

Speed Differences

While Programs 1 and 2 are identical in memory requirements, and Programs 3 and 4 are also identical to each other in terms of memory, these pairs of programs are vastly different in execution speed. Here we find that Programs 2 and 4 each require only one clock tick for completion. In marked contrast are Programs 1 and 3 which are 42 times slower!

The reason for this enormous discrepancy is the way that Atari BASIC seems to locate subroutines. BASIC is apparently incapable of immediately jumping to the desired line referenced in a GOSUB. Rather, it appears to start from the first line of the program, and sequentially search through the list of lines until it finds what it wants. Thus, in Programs 3 and 4, when it encounters in line 1000 a GOSUB 999, it must begin at line one and look at each of the 998 intervening lines until it reaches 999. There, it encounters a GOSUB 998. Back to the top of the list it goes, fruitlessly examining 997 lines. And so on. BASIC performs the same sequence of steps in Programs 2 and 4, but it obviously has to do fewer of them, since all of the subroutines are near the top of the list of line numbers.

Thus the second lesson, is to place subroutines as close as possible to the beginning of each program. Some of your programs that heretofore seemed to drag on unmercifully could now have more zip.

And, finally, we answer the question that has been bothering you for some time. No, we are not masochists. We did *not* type in 1000 lines of code in order to perform these benchmark tests. Instead, a one-line program was written to create a 1000-line skeleton for Programs 1 and 2:

```
10 OPEN #2,8,0,"D:PROGRAM : FOR J = 1 TO 1000
   : ? #2,J; " A = 0" : NEXT J : CLOSE #2
```

Then PROGRAM was ENTERed, the appropriate minor changes made to only seven lines, and then RUN. The same procedure was followed for

Program 1.

```
   1 POKE 20,0 : B = 0 : GOSUB 1000 : B = PEEK(20)
     : ? B : STOP
   2 A = 0
   3 A = 0
   4 A = 0
   5 A = 0
   6 A = 0
   7 A = 0
   8 A = 0
   9 A = 0
     etc.
 995 A = 0 : RETURN
 996 GOSUB 995 : RETURN
 997 GOSUB 996 : RETURN
 998 GOSUB 997 : RETURN
 999 GOSUB 998 : RETURN
1000 GOSUB 999 : RETURN
```

Programs 3 and 4; the sole change was that "A=0" was written as "A=A".

Program 2.

```
1 POKE 20,0 : B=0 : GOSUB 7
  : B=PEEK(20)
  : ? B : STOP
2 A=0 : RETURN
3 GOSUB 2 : RETURN
4 GOSUB 3 : RETURN
5 GOSUB 4 : RETURN
6 GOSUB 5 : RETURN
7 GOSUB 6 : RETURN
8 A=0
9 A=0
  etc.
995 A=0
996 A=0
997 A=0
998 A=0
999 A=0
1000 A=0
```

Program 3.

```
1 POKE 20,0 : B=0 : GOSUB
  1000 : B=PEEK(20)
  : ? B : STOP
2 A=A
3 A=A
4 A=A
5 A=A
6 A=A
7 A=A
8 A=A
9 A=A
  etc.
995 A=A : RETURN
996 GOSUB 995 : RETURN
997 GOSUB 996 : RETURN
998 GOSUB 997 : RETURN
999 GOSUB 998 : RETURN
1000 GOSUB 999 : RETURN
```

Program 4.

```
1 POKE 20,0 : B=0 : GOSUB 7
  : B=PEEK(20)
  : ? B : STOP
2 A=A : RETURN
3 GOSUB 2 : RETURN
4 GOSUB 3 : RETURN
5 GOSUB 4 : RETURN
6 GOSUB 5 : RETURN
7 GOSUB 6 : RETURN
8 A=A
9 A=A
  etc.
995 A=A
996 A=A
997 A=A
998 A=A
999 A=A
1000 A=A
```

©

Micro Wholesale

BEST PRICES

VIC-20 $248

5 VIC Programs $19.95
States & Capitols, Jackpot, Countries & Capitols, Black Jack, and Hangman.

COMMODORE EQUIPMENT	
VIC Printer	$ 320
VIC Disk	$ 467
PET 4016	$ 789
PET 4032	$ 952
CBM 8032	$1087
Super PET	$1636
Datasette	$ 63
Single Disk	$ 541
4040 Disk	$ 983
8050 Disk	$1333
4022 Printer	$ 613
8023P Printer	$ 768
8300P Daisy Printer	$1732

Xerox 820	$2447
Atari 400	$ 347
Atari 800	$ 697
Atari Disk	$ 447
Box of 10 Elephant Diskettes No. 1 S.S. S.D. Soft Sector	$ 21
Anadex 9501 Printer 200CPS Matrix-Graphics	$1197
Smith Corona Daisy Printer Letter Quality, Parallel or Serial	$ 787
Comet I, 124 CPS Matrix Printer Bidir. Parallel or Serl-Lmtd Qty	$ 302
Epson MX70	$ 344
Epson MX80	$ 444
Epson MX80FT	$ 547

Other Brands Carried: NEC, ALTOS, NORTH STAR, DIABLO, QUME OKIDATA, TELEVIDEO, AMDEK, MAXELL, SCOTCH, DYSAN, CITOH, ZENITH and More.

Ask For Catalog: Specify Commodore, Atari, Apple, North Star, or Peripherals.

ICS Micro Wholesale
276 N. University, P.O. Box 1243
Provo, Utah 84603 (801) 373-2901

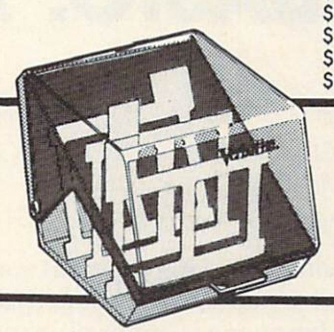
Prices are for prepayment by certified check, add 3% for Bank Cards. Personal checks accepted, but will delay shipment. When possible, all items shipped UPS, freight collect. All sales subject to availability and stock.

SPRING DISKOUNT SALE

DUST COVERS for ATARI

Protect your investment! Made of fabric (not cheap vinyl), these dust covers will more than pay for themselves in preventing problems associated with dirt and grime. All seams are machine sewn for durability. Colors are tan with chocolate brown trim.

Cat. No. 3818	Atari 800	$10.95
Cat. No. 3819	Atari 400	$8.95
Cat. No. 3820	Atari 410	$7.95
Cat. No. 3821	Atari 810	$7.95
Cat. No. 3822	Atari 850	$8.95
Cat. No. 3823	Atari 825	$8.95



DISK PROTECTOR CASES

- Holds 50 diskettes
- Hi-impact smoked plastic
- Desktop–perfect for home / or office

Cat. No. 2956 $23.00

SHADOW HAWK ONE

HORIZON SIMULATIONS

The Galactic Empire has just conquered the Noble Free Space Confederation... except for your remote outpost on a moon at the outer limits. Employing the unmatched speed ofyour warship, SHAWDOW HAWK ONE, you prey on the Empire's Merchant Fleet to capture enemy raw materials which can be bartered for better weaponry, shielding, missiles, etc. Be Alert! The Empire Wants you.

Cat. No. 3397 Atari, 48K, disk $49.95

TT#5
PLAYER/MISSILE GRAPHICS

SANTA CRUZ

This is one of the best of the tutorials from Santa Cruz Educational Software. P/M Graphics is what sets the Atari computer a cut above the rest when it comes to graphics. Learn how to create a simple shape called a player and you're on your way. This tutorial is loaded with 25 examples to create programs ranging from a complete business application to a small game.

Cat. No. 3400	Atari, 32K, cass	$29.95
Cat. No. 3401	Atari, 32K, disk	$29.95

VERBATIM DATALIFE DISKETTES

- For ATARI, APPLE II, TRS-80, etc.
- Single sided, double density
- 40 track tested
- Built-in hub rings

Cat. No. 1147 (box of 10) $28.00

HOW TO ORDER

Write or phone. Pay by check, M/C, VISA, or COD (add $1.50 for COD).
(800) 423-5387
(213) 886-9200

Offer Expires May 31, 1982
Mention this ad and we pay shipping (UPS ground only).
HW Electronics 19511 Business Center Dr. Dept. G5
Northridge, CA 91324

WHEN IN SOUTHERN CALIFORNIA, VISIT OUR RETAIL STORES

HW ELECTRONICS

19511 Business Center Dr.
Northridge, CA 91324

2301 Artesia Blvd.
Redondo Beach, CA

Extensibility means the ability to extend a computer language by adding new, customized commands. The & symbol has been used to extend Apple's BASIC, but it has drawbacks. Here's an alternative approach.

Modifying Apple's Floating Point BASIC: An & Interpreter Without the &

H. Cem Kaner and John R. Vokey
Department of Psychology
McMaster University
Hamilton, Ontario

Any computer language has some features which are clumsy and lacks others that you wish you had. If the language is generally satisfactory, you can either put up with these drawbacks or, in some cases, you can "patch" the language, adding new commands or changing old ones to meet your requirements. Applesoft, Apple's floating point BASIC, is quite easy to patch, and there is a growing collection of documentation to help the programmer along. A very direct and popular approach to patching the language uses the &.

& Interpreter

The presence of this character in a program forces a jump out of BASIC, and can force a jump to the subroutine you have written to handle a new command. As the number of available patches has grown, a further feature has been added, an ampersand interpreter. In this case, the & forces a jump to a program (the &-interpreter) which first determines which new command is being signalled, and then branches to the subroutine appropriate to that command. The &-interpreter provided by Mottala is a fine example of this kind of program. For us, this ability to interpret a wide range of user-defined commands is a very significant enhancement to Applesoft. Unfortunately, any ampersand interpreter has a drawback – that the & itself.

The problem arises if the new command flagged by the & involves a symbol Applesoft would readily treat as a command without the &. An example of this is Smith's *& GOTO* command (**COMPUTE!**, May, 1981, #12). This modification to GOTO allows the use of labels instead of line numbers in GOTO statements. For example, if X = 1000, then & GOTO X sends program control to line 1000. Forget that &, though, and Applesoft branches to line 0, no matter what X holds.

This quirk of Applesoft (no fault of Smith's) caused us no end of debugging problems, and we set out to find some way to avoid the new & HEADACHE command the Apple seemed determined to send to us. We found two solutions to the problem. first, use of the & should be restricted to flagging commands which would be gibberish to Applesoft in the absence of a preceding &. Forgetting & still causes the program to crash, but it crashes at the appropriate line number, so the error can be easily found and fixed. Second, when Applesoft commands are themselves modified, the modifying subroutine should be executed whenever that command is encountered, without requiring the presence of the & to signal something new. In this article we will show this second solution can be implemented, on all types of Apple II systems.

It is usually relatively easy to change a BASIC command if Applesoft resides in RAM memory. This occurs in one of two ways: either Applesoft has been loaded onto a 16K memory expansion card, or Ramcard, or Applesoft IIa has been loaded, usually from cassette tape, into the Apple's RAM memory. Ramcard Applesoft is identical to the Applesoft stored in ROM on the Apple II PLUS and on Apple's ROM Applesoft Board. Applesoft IIa is different from these, and we will save discussion of it until later.

If you do use Ramcard Applesoft, then the & approach to modification of BASIC commands is unnecessarily slow along with being very space inefficient. Our labs use the Apple Language System, which includes a Ramcard. By modifying the GOTO code directly, we save 60 of the 67 bytes required for Smith's labelled GOTO/GOSUB program, and do everything his program does, except that our version does not suffer from the one minor error in his (and any ROM Applesoft) version of that routine (details later). Unfortunately, our experience with Apple's rendition of PASCAL (which should not be taken as generalizing to other versions of PASCAL), was quite negative. Partially based on this, other labs in the Department in which we work decided to save their money, and did not buy the Language System or any other RAM memory expansion board. Thus, our programs would not run on their machines, and their &-laden programs would not run on ours. To maintain compatibility of systems, while avoiding the headaches inherent in &-flagged command modifications, we were forced to develop an interpreter for ROM Applesoft which works in much the same way as an &-interpreter would, except that it does not require the &.

How The Applesoft Interpreter Works

Rather than storing the literal characters of BASIC commands, and of many other key words and symbols, Applesoft represents these internally in a tokenized format. Each key word is replaced by a number, between $80 and $EA (see the *Applesoft Reference Manual*, p. 121), and can thus be stored in a single byte of memory. This is an extremely space efficient storage system. However, Applesoft now requires an interpreter to decode these tokens, in order to act on the commands, or to evaluate the functions which they represent. All key words are first tokenized, then interpreted, whether the machine is in Immediate Mode, responding to commands as you type them in, or in Deferred Mode, running a program. Our interpreter mimics the Applesoft interpreter. For this reason, and also because we think it would be helpful for anyone writing modifications to Applesoft, we will describe the behavior of the Applesoft interpreter in some detail.

The Flow Of Control

The Applesoft interpreter starts at location $D805. We cannot list this copyrighted routine here, but you can see it for yourself if you jump to the MONITOR (via CALL -151) and then type in "D805L". Having the routine in front of you may clarify the flow of control described below.

This program begins, at $D805, by determining whether the TRACE command is in effect (flagged by the contents of $F2). If so, and if a Deferred Mode program is being run (flagged by contents of $76), then, before each command is executed, the "#" sign is printed, followed by the line number. The first location following this printout is $D81D, which is branched to directly if the printing is not to be done.

At $D81D, the CHRGET routine ($B1-$C8) is called. This subroutine fetches the next character of the program and sets the Zero flag if that character signals an "end of line," that is, if the character is a carriage return or a colon, ":". The routine then calls, via a JSR, the actual interpretation subroutine, which starts at $D828. This returns immediately, via the RTS at $D857, if an end of line is encountered. Otherwise, the program will return from this subroutine later, in a more indirect way. When this subroutine is returned from, the interpreter exits by jumping to a routine called NEWSTT (NEW STaTement) at $D7D2, which will execute the next program statement, falling back into this interpreter in the process.

If CHRGET did not find an end of line, the $D828 subroutine expects to find a command of some sort. If the character fetched by CHRGET is not a token, the interpreter assumes the programmer intended a LET command (eg X = 100) and jumps to the LET subroutine at $DA46. The interpreter determines whether it has a token by subtracting $80, the value of the smallest token, from the Accumulator (A), which holds the character. If A is still positive, we have a token. This token may represent a command ($80 through $BF), or it may be some non-command key word ($C0 through $EQ). Since we have already subtracted $80 from A, we have a command only if A is less than $40 ($40 + $80 = $C0), which is checked by a CMP (compare) instruction. If A is not less than $40, we do not have a command, which is what should be here, so the interpreter jumps to $D846, thence to $DEC9, which produces a "? SYNTAX ERROR".

The Command Table

If we are dealing with a command, the next job of the interpreter is to determine where to go to execute it. There is an address table, beginning at $D000, (Applesoft IIa: $0800) which contains this information. In this table, the starting address of every command, less 1, is stored in order of magnitude of the command's token. Thus the address of END, whose token is $80, is stored first, from $D000 to $D001. FOR's token is next, and the address of the FOR routine, less 1, is stored from $D002 to $D003. Since $80 was subtracted from A, A now stores a number between $00 and $3F. Double this number, by rotating A left, add it to $D000, and you get the location of the two byte address of the command.

The addition is accomplished by indexing $D001 and $D000 with register Y, after Y is loaded with the doubled contents of A. The command's address, less 1, is then pushed onto the stack. When the next RTS is encountered, the program will "return" control to the last address on the stack, after adding 1 to that address. Thus, the next RTS we encounter will force a jump to the correct starting address of the command to be executed. The actual location of the interpreter's RTS is hidden. The final instruction of the interpreter is a JMP, rather than a JSR, to CHRGET, which will fetch the first character following the command. The RTS from CHRGET is the one which takes us to the command itself.

Note that the next address on the stack is the address of the routine which called this interpreter. We have already seen what happens when this one is returned to: the interpretation process stops and the program jumps to NEWSTT. As soon as the RTS at the end of the command we will execute is encountered, the program will effectively branch to NEWSTT.

What Happens When Applesoft Runs Into An &

The & symbol is tokenized in the same way that BASIC commands are tokenized, so, even though & is not a proper BASIC "command" (see the *Applesoft Reference Manual),* the interpreter will treat it as if it were one. The & address in the token lookup table is $03F5 (less 1). That is, the CHRGET routine jumped to by the interpreter will return to location $3F5 when it has fetched the next character following the &. At $3F5, there are three free locations, which typically contain an instruction to jump to some other address, say $0300, where the actual &-interpreter begins. The &-interpreter will then typically examine the contents of the accumulator, which holds the latest character fetched by CHRGET, and will operate on these in much the same manner as the Applesoft interpreter did for the character it got from CHRGET. There are various ways to accomplish this, but the effect will be to eventually find yet another address (the start of the command flagged by the &) and to force a JMP to that address. The return at the end of the subroutine jumped to will be a return back to the Applesoft Interpreter and, thence, to NEWSTT, as described above.

How To Get Rid Of The & in ROM Applesoft

Our goal is to write a command token interpreter which will handle modified Applesoft commands. We are not worried about things flagged by the & which would not otherwise be treated as commands. We could extend the routine below so that it would handle these, but, given the implementation, (i.e. as a patch to CHRGET, which is very frequently called), this would noticeably slow down Applesoft, so we do not recommend it. Further, we could extend the interpreter so that it would handle tokens which are not commands, allowing modification of functions. Again, we do not recommend this. Applesoft provides a USR function with the explicit intention of allowing user defined functions. Our experience with functions is that they are not really modified at all. Rather, they are replaced by something quite different, which the user considers better.

In this case, we feel that the appropriate vehicle for replacement is the one deliberately built into Applesoft, i.e. USR. An &-based alternative, or an alternative using this routine, would be inappropriate as well as quite clumsy. Given these limited objectives, along with a desire (if only to ease our memory burden) to mimic the Applesoft token interpretation approach, the modification to ROM Applesoft in order to allow compatability with RAM Applesoft without requiring ampersands to flag modified commands is surprisingly straightforward.

Nearly all of ROM Applesoft resides in ROM and so cannot be changed. One important routine, CHRGET, does not reside in ROM. CHRGET is loaded into memory whenever Applesoft is initialized (e.g. by the FP command if you have a disk). This is exactly the routine called by the Applesoft interpreter whenever it wants a new command. We modify CHRGET so that it jumps to a routine we call NEWGET, located at $300, which will replace CHRGET. Along with doing everything CHRGET used to do, this routine checks whether the character it fetches is a token in a list of modified command tokens. If so, it executes that command before returning to the Applesoft interpeter.

All properly written Applesoft commands and programs will leave the TeXT PoinTeR (TXTPTR) pointing to an end of line following the command, once the command has been executed. User written commands must conform to this as well. If they do, then, when NEWGET finally returns control to BASIC, it will pass the Applesoft interpreter an "end of line." This forces, as we have seen above, a branch to NEWSTT, which is just what we want in order to avoid confusion in Applesoft over what should be executed next.

Further Notes

Here are a few further notes on the ROM Applesoft token interpreter subroutine listed at the end of this article, which may not otherwise be clear from the discussion above.

1. The CHRGET routine conceptually divides into two subsections. The first increments TXTPTR, to point to the next character. The next section is often called CHRGOT, and this is the routine which actually fetches the contents of the location pointed to by TXTPTR and sets up various internal flags. By loading JMP $300 into the first three bytes of CHRGET, we effectively destroy the 6-byte segment of code which increments TXTPTR, and thus have to repeat this as the first 6 bytes of NEWGET. In the process, we free the three page 0 locations, $B4, $B5, $B6, which follow the JMP $300. These are used as temporary locations by NEWGET. The CHRGOT routine is completely unaffected by the jump, so we can still use it to actually fetch the character and to set the appropriate flags (eg. Zero).

2. CHRGET has no effect on registers X and Y. We will use both in searching the token table. If we do not return the original values of X and Y when we return from NEWGET, we will induce errors in the many routines which call CHRGET and which assume that this call will not affect these registers.

3. Variable FLAG checks whether or not a command is currently being executed. If FLAG is 0, we are not in the midst of handling a command. If FLAG is not 0, then the user has typed in something like GOTO GOTO, which is a syntax error. Since Applesoft syntax is very clear on this point – all commands must be separated by ends of line (colon or carriage return) – we return "? SYNTAX ERROR" if we find a command token while executing a command. Since BASIC is not in direct control of program execution at this point, this program must do its own error checking.

4. Our lookup table differs in format from the one at $D000. We also store jump locations, less 1, and will get to these via an RTS, as the Applesoft interpreter does. However our table also holds the target tokens themselves. The first and second locations of CMDTBL (CoMmanD TaBLe) hold the low and high bytes of the address of the subroutine which will execute the command whose token is stored in the third location. Similarly for the fourth and fifth locations (addresses) and the sixth location (token), and so on. The command table is stored in this version of the program at location $0354. There is no need whatever to store it here. You can put it in any convenient place in memory, as long as you change the four places in the program which refer to the starting address of CMDTBL (before JMPGOT and after GOTONE).

5. End of CMDTBL is indicated by a zero value for the target token. We load the token into register Y at $31D in order to test for end of table. If Y is zero, the command token is not one we are looking for, so we exit.

The Labelled GOTO/GOSUB Example

1. Comments on the patch.

We use Smith's routine as an example partially because it was published in **COMPUTE!**, so you may be familiar with it, and partially because we have found it quite useful. The routine has been completely rewritten in three ways, once for ROM Applesoft, once for RAMCARD Applesoft, and once for Applesoft IIa, or TAPERAM. A more complete discussion of the logic of the routine is in Smith's article.

The effect of the patch is as follows: Taking X to mean any arithmetic expression or variable, (X may, but need not, be a literal number), then if the value of X is 1000, GOTO X will be treated by Applesoft in the same way as GOTO 1000. Similarly, GOSUB X will be treated as GOSUB 1000. Thus labels can be defined for subroutines (as all rational programming languages, including nearly all assemblers, allow) and for GOTO statements (reminiscent of FORTRAN's ASSIGN statement). If X is a *real* number, it is rounded down to the nearest integer, so be wary of arithmetic expressions.

This patch does not affect the behavior of ON...GOTO and ON...GOSUB, so these must still use line numbers rather than labels. However, you can replace these computed GOTO's in your code by computed GOTO's of a very different type, which are reminiscent of PASCAL's CASE handling. As an example, at the start of your program you might DIMension a matrix SELECT(20) and assign the values of 20 different line numbers to the values of SELECT(I). To GOTO these lines, you can compute the value of I, and then GOTO SELECT(I). With decent commenting on the intention and conditions of each choice of SELECT's line numbers (which is best done at the place in the program that the line numbers are actually assigned to SELECT's elements,) your program will probably be much more readable than one with an equally commented ON...GOTO statement. This is our experience with these two different forms of computed GOTO and GOSUB, and we have stopped using ON...GOTO completely.

Spaghetti Structure

A different method of implementing a computed GOTO is ideal for making your program structure resemble a plate of spaghetti. If you change the value of X at various points in the program and repeatedly use X in GOTO X or GOSUB X statements, then, if you succeed in debugging your program, you will be able to amaze your friends and neighbors with your ability to produce unintelligible, yet functional, code. The labelled GOTO/GOSUB facility as presented here can be used to dramatically increase the readability of your program, or it can be abused to degrade the structure of your program. We strongly recommend that you assign line numbers to specific variables at the start of the program, use informative names for those variables, and never change their values once assigned.

2. The actual patches

The ROM Applesoft patch is given in the appended listing, directly after the lookup table. The program is virtually the same as Smith's, with a few more comments. Note that whether you use this program or Smith's, IF...THEN X, IF...GOTO X and IF...THEN GOTO X will not work properly. We have tried and tried, but cannot fix this flaw in a program of reasonable length. A statement of the form IF...THEN: GOTO X (or GOSUB X) will work correctly. There must be a colon between the THEN and the GOTO or the GOSUB.

The RAM Applesoft patch is much simpler. For either RAM Applesoft version, you do not need the subinterpreter. Instead, modify GOTO directly. GOTO is a subroutine of GOSUB, so this modifies both. Change the JSR $DA0C (the LINe number GETting subroutine) at $D93E (Ramcard), or the equivalent command at $1140 for Taperam to JSR $300 (or wherever you wish to store this patch). This changes the first command of GOTO, forcing it to treat the patch as its line number getting subroutine. The patch consists of:

JSR FRMEVL
JSR GETADR
RTS.

The Ramcard locations of these routines are $DD7B for FRMEVL, which evaluates the expression following the GOTO or GOSUB and deposits this in Applesoft's floating point accumulator, FAC, and $E752 for GETADR, which moves the contents of FAC to locations $50 and $51, where GOTO expects to find the line number to which to go. The locations for TAPERAM are $157E, and $1F49 for the two routines. That's all there is to it. Seven bytes of new code and, to top it off, IF... GOTO X works (IF...THEN X will not), as do IF...THEN GOTO X and IF...THEN GOSUB X though, to preserve program portability, you will probably want to include the colon, entering only IF...THEN: GOTO (or GOSUB) statements when using labels rather than literal line numbers.

To use this routine, the Applesoft CHRGET routine at $B1 must be modified. This may be accomplished from the monitor by writing B1:4C 00 03NB6:00 or by calling the initialization patch (included below) from BASIC.

```
0300              ;EQUATES:
0300
0300              SYNERR =$DEC9          ;GENERATE SYNTAX ERROR
0300              TXTPTR =$B8
0300              CHRGOT =$B7
0300              XTEMP  =$B4
0300              YTEMP  =$B5
0300              FLAG   =$B6            ;FLAG COMMAND IN PROGRESS
0300
0300              ;THE NEW CHRGET ROUTINE:
0300
0300 E6B8         NEWGET INC TXTPTR       ;INCREMENT LOW BYTE
0302 D002                BNE GETCHR      ;GET NEXT CHARACTER
0304 E6B9                INC TXTPTR+1    ;INCREMENT HIGH BYTE
0306 20B700       GETCHR JSR CHRGOT       ;GET CHARACTER
0309
0309              ;THE SUB-INTERPRETER:
0309
0309 C9C0                CMP #$C0        ;IS IT A NON-COMMAND TOKEN?
030B B025                BCS OUT         ;YES, GO
030D 86B4                STX XTEMP       ;ELSE, SAVE X
030F 84B5                STY YTEMP       ;AND Y
0311 A8                  TAY             ;TEST N FLAG
0312 1016                BPL JMPGOT      ;NOT A TOKEN?, GO
0314 A4B6                LDY FLAG        ;COMMAND IN PROGRESS?
0316 D01D                BNE ERROUT      ;YES, ERROR
0318 A2FF                LDX #$FF        ;NO, SET X FOR TABLE LOOKUP
031A E8           NXTCMD INX              ;POINT X AT NEXT COMMAND TOKEN
031B E8                  INX
031C E8                  INX
031D BC5403              LDY CMDTBL,X    ;GET TOKEN
0320 F008                BEQ JMPGOT      ;END OF TABLE? GO
0322 DD5403              CMP CMDTBL,X    ;ELSE, COMPARE TO TABLE TOKEN
0325 D0F3                BNE NXTCMD      ;NO MATCH, GET NEXT COMMAND
0327 203803              JSR GOTONE      ;ELSE SET UP FORCED JUMP
032A A200         JMPGOT LDX #0
032C 86B6                STX FLAG        ;CLEAR FLAG
032E A6B4                LDX XTEMP       ;RECOVER X
0330 A4B5                LDY YTEMP       ;RECOVER Y
```

```
0332 4CB700 OUT     JMP CHRGOT        ;GET CHAR AT TXTPTR
0335                ;                  AND RETURN TO APPLESOFT
0335
0335 4CC9DE ERROUT JMP SYNERR         ;DO SYNTAX ERROR
0338
0338                ;SET UP FORCED JMP VIA RTS AND SET FLAG
0338
0338 85B6   GOTONE STA FLAG           ;FLAG COMMAND IN PROGRESS
033A BD5303         LDA CMDTBL-1,X ;GET HIGH BYTE
033D 48             PHA               ;AND DEPOSIT ON STACK
033E BD5203         LDA CMDTBL-2,X ;GET LOW BYTE
0341 48             PHA               ;AND DEPOSIT ON STACK
0342 60             RTS               ;EXECUTE USER PATCH
0343
0343                ;INITIALIZATION PATCH:
0343                ;A CALL 835 FROM BASIC WILL INITIALIZE
0343                ;THE ROM SUB-INTERPRETER.

0343
0343 A94C   INIT    LDA #$4C          ;LOAD A 'JMP' AND
0345 85B1           STA $B1           ;STORE AT CHRGET
0347 A900           LDA #<NEWGET      ;LOW BYTE OF INTERPRETER
0349 85B2           STA $B2
034B A903           LDA #>NEWGET      ;HIGH BYTE
034D 85B3           STA $B3
034F A900           LDA #0            ;CLEAR THE
0351 85B6           STA FLAG          ;COMMAND IN PROGESS FLAG.
0353 60             RTS               ;RETURN TO BASIC
0354
0354                ;COMMAND TABLE:
0354                ;COMMANDS AND THEIR PATCH ADDRESSES
0354                ;ARE STORED TOGETHER IN LOW-BYTE, HIGH BYTE,
0354                ;COMMAND TOKEN ORDER.  THE LAST THREE BYTES
0354                ;OF THE TABLE MUST BE ZEROS.
0354
0354 5C03   CMDTBL .WOR GOSUB-1       ;GOSUB PATCH ADDRESS
0356 B0             .BYT $B0          ;GOSUB TOKEN
0357 7603           .WOR GOTO-1       ;GOTO PATCH ADDRESS
0359 AB             .BYT $AB          ;GOTO TOKEN
035A 00             .BYT 0,0,0        ;END OF TABLE
035B 00
035C 00
035D
035D                ;AS AN EXAMPLE OF USING THE ROM TOKEN INTERPRETER,
035D                ;WE INCLUDE A LISTING OF A PATCH THAT PROVIDES
035D                ;LABELLED GOSUBS AND GOTOS IN APPLESOFT BASIC.

035D                ;THIS CODE IS FROM M.R. SMITH (COMPUTE, 12, 1981).
035D                ;FOR THE GOSUB PATCH, IT IS EFFECTIVELY A
035D                ;RELOCATION OF THE INITIAL PORTION OF THE
035D                ;APPLESOFT GOSUB CODE.  THIS ENABLES A
035D                ;MODIFICATION OF THE SECTION OF CODE THAT
035D                ;JUMPS TO THE APPLESOFT GOTO CODE, WHERE
035D                ;THE EFFECTIVE CHANGE IS MADE.
035D
035D                ;APPLESOFT POINTERS AND ROUTINES:
035D                CURLIN =$75          ;CURRENT LINE NUMBER
035D                NGOSUB =$D7D2        ;NORMAL GOSUB
```

```
035D        NGOTO  =$D941
    ;NORMAL GOTO
035D        FRMEVL =$DD7B
    ;EVALUATE EXPRESSION AT TXTPTR
035D        STACK  =$D3D6
    ;CHECK ON STACK POINTER
035D        GETADR =$E752
    ;TRANSFER FAC TO LINNUM
035D
035D               ;GOSUB PATCH:
035D
035D A903   GOSUB  LDA #3
    ;NORMAL GOSUB RELOCATED
035F 20D6D3              JSR STACK
    ;FROM $D921
0362 A5B9                LDA TXTPTR+1
    ;STORE TXTPTR
0364 48                  PHA
    ;ON STACK
0365 A5B8                LDA TXTPTR
0367 48                  PHA
0368 A576                LDA CURLIN+1
    ;STORE CURRENT LINE NUMBER
036A 48                  PHA
    ;ON STACK
036B A575                LDA CURLIN
036D 48                  PHA
036E A9B0                LDA #$B0
    ;MARK GOSUB
0370 48                  PHA
    ;ON STACK
0371 207703              JSR GOTO
    ;DO A MODIFIED GOTO
0374 4CD2D7              JMP NGOSUB
    ;FINISH NORMAL GOSUB
0377
0377               ;GOTO PATCH:
0377               ;THIS IS WHERE THE EFFECTIVE CHANGE OCCURS.
0377               ;BY REDIRECTING THE EVALUATION OF WHAT FOLLOWS
0377               ;THE GOTO OR GOSUB TOKEN TO FRMEVL (WHICH
0377               ;EVALUATES THE EXPRESSION FOLLOWING THE TOKEN),
0377               ;AND THEN TRANSFERING THE RESULT FROM FAC
0377               ;(THE FLOATING POINT ACCUMULATOR) TO LINNUM
0377               ;(WHERE THE REMAINDER OF THE ACTUAL GOTO
0377               ;ROUTINE EXPECTS TO FIND THE LINE NUMBER),
0377               ;LABEL AND EXPRESSION EVALUATION FOLLOWING
0377               ;GOTO OR GOSUB IS EFFECTED.
0377
0377 200003 GOTO   JSR NEWGET      ;GET NEXT CHARACTER
037A 207BDD        JSR FRMEVL      ;EVALUATE EXPRESSION
037D 2052E7        JSR GETADR      ;TRANSFER FAC TO LINNUM
0380 4C41D9        JMP NGOTO       ;FINISH NORMAL GOTO
0383
0383               .END            ;END ASSEMBLY
```

©

"Garbage collection" refers to the long delays which can occur while the computer rearranges strings. This article shows how to avoid these delays when you are working with significant numbers of strings. The technique here is most useful for PET/CBM owners who have older machines. Newer machines with BASIC 4.0 avoid these problems.

Screen Input On The PET

Elizabeth Deal
Malvern, PA

Using the information placed on the screen as a source of *input* seems like a contradiction in terms. Why should one bother inputting something when one already knows what it is? I stumbled upon one good reason and worked with it to a happy ending: no garbage collection delays.

This article runs through a series of small experimments. Both tests and conclusions are based on work in the Upgrade PET. Users of original ROMs and pre-Fat 40 BASIC 4, 40 and 80 column systems are invited to try the tests. I suspect that the results will be the same. However, simple as it is, I just don't know how the whole thing will behave in another PET. The concept of null input might be handled differently and, together with POKEing two system locations, it might crash the PET. This might mean it will need to be reset. There is no way of telling until you join the fun.

Screen input, as described in this article, is most valuable in the systems prior to 4.0 BASIC as it shows yet another way of minimizing character string handling in our quest towards a garbageless PET. My method is limited, however, to only certain applications. The entire problem was explained and a more general procedure was proposed by Jim Butterfield in **COMPUTE!**, September, 1981, #10. BASIC 4 users, of course, don't suffer from these problems.

The reason I became interested in screen input is twofold. First, the suggestion appears in the POWER™ manual, and it made me curious about the method's utility. [*POWER is a chip which adds several commands to BASIC. It is sold by Professional Software.*] Secondly, I have a very nice disk utility which displays all the sorts of data about the contents of the floppy. The program carefully displays that information by use of ordinary PRINT statements (no reverse field, no cursor controls, nothing, just plain letters). I needed to lift this data, put it into variables, and use it for other things, not excluding a sorted listing. It seemed like a simple task, until a seven minute long garbage collection zapped me during testing.

The garbage collection occurred because, although the program originally concatenated (added) pieces of strings by use of PRINT statements in a loop, as in

```
PRINT#4,CHR$(X+48);
```

I had to change the code to

```
V$(3,I)=V$(3,I)+CHR$(X+48)
```

So the concatenation would run its course eight times before I could touch the value. (Putting each character into an array is out of the question). This invited trouble. It didn't really happen, as the number of strings wasn't that large. But, when I let the program do the work several hundred times while simulating a small PET and a large disk drive, the PET got clogged up with strings and produced an annoying, slow motion spectacle, eventually ending in the interminable pause for housekeeping. That did it.

Clearly, there are many approaches to such a simple task, but the challenging aspect in this case was that the screen already contained neatly formatted data in the exact form I wanted. If I could only, somehow, pick up these fields and stuff them into the elements of a string array V$(record,field) I'd be home free. I couldn't resist trying the screen input idea. It just seemed right for the task.

Testing

Before actually using a new scheme in a program it pays to understand how the process really works, remembering Murphy's Law: "nothing is ever as simple as it looks." Let's run through experiments that seem to provide several definitive answers.

A note to non-Upgrade BASIC users: I don't know what these tests will do to you. Several conversion values are provided, but if your system works differently, it is up to you to discover it. The amount of typing is insignificant until the very end. By then you will know enough to avoid serious trouble.

(1) Type a line: 1050 OPEN3,3:INPUT#3:CLOSE3. Clear the screen and RUN it. The only redeeming feature of this result is that we haven't killed the PET. The error tells us that some bad values have landed in the OS area. Give up or think. In the remaining exercises you do not need to clear the screen. As a matter of fact, it's more enlightening if you don't.

(2) Add these lines:

```
100 PRINT"ONE";:GOSUB1000
200 ?:?:PRINT A$,LEN(A$);ASC(A$+CHR$(0));
    VAL(A$)
210 END
```

```
1000 :
1500 RETURN
```

And RUN it. Well, it doesn't do anything. A$ is a useless string, a long chain of blanks, not exactly our intention.

(3) Let's position the cursor over the string. Move it back to "0" on "ONE" by inserting three left cursors within the quotes after "E". RUN it. There is hope, but note the unacceptable length of the string.

(4) We can limit PET's wish for characters by printing a comma after the "ONE". But that's the last thing I need in my neat display. Let's convince the PET, instead, that its screen width is not 40 characters (or 80 in 80xx), but only three. Butterfield's memory map indicates that information is held in 213. I don't know what it is in Original PETs, those maps were never printed in **COMPUTE!**. (Perhaps an equivalent of 213 does not exist. If anyone would like to contribute an accurate cross-reference type map for all these systems, please send one in). Type two lines:

```
1020 POKE213,3    to squeeze the screen width
1080 POKE213,39   to restore it back.
                  (PET counts from zero)
```

This works. Incidentally, line 1080 may not be needed, but out of overabundance of caution I'll keep it. PET's self preservation instinct seems to restore 213 for subsequent screen work. Query whether 80-column window facility might be useful.

(5) We're not out of the woods yet. What will happen if "ONE" were followed by a string of blanks, a typical sight in a left adjusted alphabetic information? Modify two lines to be:

```
100 ?LEFT$("ONE"+"10 sp",10)+"10 left";:
    GOSUB1000
1020 POKE213,10
```

RUN it. Trailing blanks are handled correctly.

(6) What about leading blanks you might see in a right-justified number? Change one line:

```
100 ?RIGHT$("10 sp"+"472",10)+"10 left";:
GOSUB1000
```

and RUN it. Oops! The value is intact, but PET stripped the leading spaces, as it always does in INPUT. This result may be satisfactory for many applications, too sloppy for others. Let's handle it.

(7) One fix might involve printing a phony character in the first position. But then, if we wanted to use the value in a numeric variable, we'd have to strip the character. Instead, we'll use a harmless quote by typing 100 PRINT CHR$(34) + RIGHT$ ... etc. and increase the screen width to 11 in line 1020. Unfortunately this places a character on the screen. Unavoidable. RUN it. Not bad. The leading blanks are there, the value is all right, but the original string is useless with the in-quote cursors.

In fact, any formatting command on the same line, i.e. prior to going into 1000, will print this way. TAB(x) will yield x right cursors. That's the way PET is built. We could POKE 205 to disable the quote mode, but that gets messy in some applications. We could close the quote, but that increases the space penalty on the line by another character. A less limiting way is needed.

(8) Remember that all we really want is to move the cursor left. PET doesn't care how. The memory map indicates the current cursor position is held in 198 (Original 226?). To move the cursor into the first position we'll put a zero there. In line 100 delete 10 left cursors and its surrounding quotes. Type 1005 POKE198,0 to put cursor in the first position, and 1040 POKE 198,11 so that a subsequent field prints in a correct position. RUN it. It really works now. We can handle leading and trailing blanks and anything in between. Right?

(9) Not quite. How about a null string? It can happen. If a condition is true you might be printing a "*", if false you'll print nothing and use TAB to position to the next item, or skip line if it is at the end. Our 1000 subroutine can't predict the future, and will attempt to process a null string with all its inherent troubles. Let's see what happens. Type 100 PRINT"";:GOSUB1000 and don't type, but imagine, 110 PRINT TAB(!0)"whatever". RUN it. Very nasty. There is actually more wrong than the eye can see. If the PET is alive you have just learned, for instance, the easiest way of causing active files to vanish into thin air (look at the table in $0251-026F on the Upgrade and 4.0 systems). Tragic results, to say the least.

(10) A quote may help. Type 100 PRINT CHR$(34) + "";:GOSUB1000. Once again, this gives a correct result, but subsequent TAB or cursor characters will cause trouble. This was not very important with numbers. It's vital that it doesn't happen here.

(11) While we correct this mishap, let's make the routine a bit more general for further testing. Type NEW and the code in lines 10-260, adjusting 198, 213 and screen width of 39 to your system. P1 is the first position of the field. P2 is the final position after a string, null or otherwise, is printed. You can try the code on various things within the quotes in line 30. When done, type two quotes there (" ") and convince yourself that it can't work. Then remove REM from line 190 and try it. This works. Replace REM in 190 and remove REM in 200. This also works with some space penalty.

Either method should work for a particular application – the choice is a function of further use for the field. Lines 190 or 200 simply check if the cursor has moved. If not, the subroutine detects a null string and makes the needed adjustments by faking a different screen width. In case of 200, it must also reset 198, since the PRINT statement

ASERT yourself . . . with CFI's new Database Retrieval System

WHO CAN USE ASERT?

libraries
personnel departments
dating services

OFFICIALLY APPROVED BY COMMODORE

schools
employment agencies
accountants

ANY BUSINESS THAT KEEPS RECORDS CAN USE ASERT TO:

- Create up to 21 fields per record
- Restructure fields at any time
- Sort on any field at any time
- Use FREE-TEXT area for comments
- Create up to 90 searchwords
- Search & retrieve on any combination of 90 searchwords
- Search with MUST HAVE, MAY NOT HAVE and OPTIONAL operators
- Print out hardcopy including labels
- Output to any word processor
- Compile summary statistics
- Maintain 1900 records per disk with "virtual" 5K record length

ASERT — Aid for Search & Retrieval of Text — $495 complete

For the 8032 CBM and 8050 disk drive — Commodore Approved Software

OTHER CFI SOFTWARE

Federal Income Tax Preparation System*
Personal Tax Calculator*
Emergency Control Program*
VIC Animation Tutorial

*Distributed for CFI under the Commodore label

ALL CFI SOFTWARE AVAILABLE

from your local Commodore dealer
or direct from CFI

CFI . . . *Computer Solutions*, 201 West 92 St., New York, NY 10025

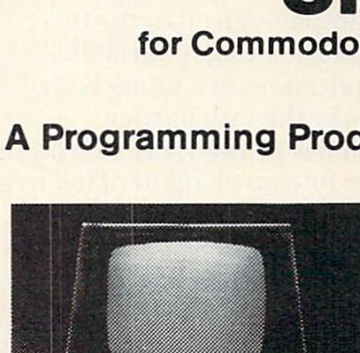
SM-KIT

for Commodore Computers

A Programming Productivity Tool

The SM-KIT is a collection of machine language firmware programming and test aids for BASIC programmers. SM-KIT is a 4K ROM (twice the normal capacity) which you simply insert in a single ROM socket on any BASIC 4 CBM/PET–either 80 column or 40 column. Includes both programming aids and disk handling commands.

ERROR DETECTION: the SM-KIT automatically indicates the erroneous line and statement for any BASIC program error.
LINE NUMBERING: the SM-KIT automatically numbers BASIC statements until you turn the function off.
SCREEN OUTPUT: the commands FIND, DUMP, TRACE and DIRECTORY display on the CRT while you hold the RETURN key (display pauses when the key is released). Continuous output is selected with shift-lock.
OUTPUT CONTROL to DISK or PRINTER: in addition to displaying on the CRT, you can direct output to either disk or printer.
HARDCOPY: allows screen displays to be either printed or stored on disk.
FIND: searches all or any part of a program for text or command strings or variable names. Either exact search or wild card search supported.
RENUMBER: the SM-KIT can renumber all or any part of a program. The selective renumbering allows you to move blocks of code within your program.
VARIABLE DUMP: displays the contents of floating point, integer, and string variables (both simple and array). Can display all variables or any selected variables.
TRACE: SM-KIT can trace program execution either continuously or step by step starting with any line number. Selected program variables can be displayed while tracing.
DISK COMMANDS: as in DOS Support (Universal Wedge), the "shorthand" versions of disk commands may be used for displaying disk directory, initializing, copying, scratching files, load and run, etc.
LOAD: SM-KIT can load all or part of BASIC or machine language programs. It can append to a program in memory, overwrite any part of a program, load starting with any absolute memory location, and load without changing variable pointers.
MERGE: allows merging all or any part of a program on disk with a program in memory.
SAVE and VERIFY: SM-KIT provides one step program save and verification. It also allows you to save any part of a program, or any address range.



ONLY $40

A 4K ROM with both programming and disk handling aids.

Developed by (and available in Europe from) SM Softwareverbund-Microcomputer GmbH, Scherbaumstrasse 29, 8000 Munchen 83, Germany

252 Bethlehem Pike
Colmar, PA 18915
215-822-7727 **A B Computers**

WRITE FOR CATALOG.
Add $1.25 per order for shipping. We pay balance of UPS surface charges on all prepaid orders. Prices listed are on cash discount basis. Regular prices slightly higher.

changes it. Unless Murphy has other plans, this completes the tests.

You should be able to guess now what would happen if the PRINT statement contained cursor controls, forced quotes, commas, colons, a clear screen command, or reverse fields (both at the beginning and the end of the space) within the quotes. All the information needed to answer the questions is here and in the known facts of an ordinary INPUT statement. If you can't guess you may want to try these things by repeating step 11. We don't need those answers for the simple application below, but we may need them for other tasks.

The Problem Solved

Since we now understand some of the PET's rules in this game we can try a slightly more ambitious project: correctly picking up multifield records, our original intent. A routine in lines 510-1060 is used. By way of illustration, we use unformatted data from DATA lines. Two PRINT statements format the data on the screen, avoiding garbage collection problems in the string construction. (The strings are actually constructed in high memory, but that does not contribute to the garbage.) PRINT C$ does not format the data, therefore a null string will result on output/input, a slight complication if you want to leave the program intact. The shifted-space method is used. I leave it as an exercise to the reader to try null strings elsewhere on a line, while positioning subsequent fields correctly. An example is given in the two REM lines, which introduce another variable. Admittedly these complications are unnecessary, they are included to make the task harder.

As each part of a line is printed, the subroutine in 1010 places its image into an element of an array A$(line,field). Note that in some applications you may not even have to go after each field. You may wish to pick up one, two, three, or all four fields in a single procedure by placing GOSUB1010 where you want it.

There are no new concepts in the code. The key variables are:

A$(I,J)	**destination array**
I	**record or screen line number**
J	**field number, counted in subroutine from left to right**
P(J-1)	**cursor position before the field is printed**
P(J)	**cursor position after the field is printed**
W	**screen width varying with P(J)**
WS	**system screen width address where we adjust screen size**
CP	**system address of cursor position on a line**
PS	**forces tabbing on a null string if such must exist. PS default is 1.**
P	**flags when we're dealing with a null string.**

The two P arrays could have been coded as single variables. They are provided in array form, for we may need those values for something else.

When you run this demo program, you will see that the method works and is not all that complicated. Here, we have purposely set up certain roadblocks to see how far we can push the PET.

There are more uses of screen input, just as there are more aspects to the method. We just scratched the surface. But once we have the basics we can go on to bigger things.

Screen input is useful for working with strings without the usual penalty. I thank Jim Butterfield for hinting at this in the POWER documentation and, more importantly, for offering a valuable warning about trouble spots.

```
10 REM SCREEN INPUT  ELIZABETH DEAL
20 P1=POS(0)
30 PRINT"ANYTHING";:GOSUB180
40 PRINT"'":PRINT
50 PRINTA$,LEN(A$)ASC(A$+CHR$(0))
60 END
180 P2=POS(0):P=0
190 :REM IFP2=P1THENP=1
200 :REM IFP2=P1THENP=1:PRINTCHR$(1
    60);
210 POKE198,P1
220 POKE213,P2+P
230 OPEN3,3:INPUT#3,A$:CLOSE3
240 POKE213,39
250 POKE198,P2+P
260 RETURN
500 :
510 W=39:WS=213:CP=198
520 R=3:NV=5:DIM A$(R,NV),P(NV)
530 BL$="      ":PF$=CHR$(160):PS=1
540 FORI=1TOR:J=0:READ A$,B,C$
550 : REM PRINT"";:IFI=2THENPRINT"*
    ";
560 : REM PS=2:GOSUB1010
570 PRINT LEFT$(A$+BL$,8);:GOSUB1010
580 PRINT PF$+RIGHT$(BL$+STR$(B),6)
    +" ";:GOSUB1010
590 PRINT C$;:PS=1:GOSUB1010
600 PRINT:NEXTI:PRINT
610 FORI=1TOR:FORJ=1TONV
620 PRINTA$(I,J);
630 NEXTJ:PRINT:NEXTI:PRINT:END
640 DATA FIRST,12345,TEXT  1
650 DATA SECOND,12,TEXT 11
660 DATA THIRD,,,
1000 :
1010 J=J+1:P(J)=POS(0):P=0
1020 IFP(J)=P(J-1)THENP=1:PRINT PF$;
1030 POKECP,P(J-1):POKEWS,P(J)+P*PS
1040 P(J)=P(J)+P*PS
1050 OPEN3,3:INPUT#3,A$(I,J):CLOSE3
1060 POKEWS,W:POKECP,P(J):RETURN
```

©

the retailer™

cash register program/inventory management on the Commodore CBM

The **retailer**™ contains the following programs:

Cash Register
- Records sales transactions
- Prints register tape or invoice
- Automatically computes discounts and/or sales tax
- Computes change due
- Prints daily sales summaries

Inventory Manager
- Creates and maintains complete information on every item in inventory (over 3000 items)
- Adds, changes, deletes inventory on command
- Automatically updates inventory from register transactions
- Maintains stock balance, records dollar amounts sold as well as cost of items sold
- Prints price tags

Report Program
Reports produced:
- Listing of:
  - All Inventory items
  - Items sold in past week
  - Items on order
  - Items on hand by season
  - Items with zero stock balance
- Dollar listing by stock category of:
  - Weekly sales
  - Cost of weekly sales
  - Merchandise on hand
  - Merchandise on order
  - Merchandise at mark down
  - % sales to total sales
  - % stock to total stock
  - % gross profit
  - % stock at mark down
- Daily summary of sales by month

available now $650.00

Dealer inquiries invited

RETCOM Systems, Inc., 61-B Mountain Blvd., Warren, New Jersey 07060 (201) 561-3112

Extra Colors For Atari Through Artifacting

Bill Mohn
Danville, CA

How can you get up to four colors in GRAPHICS 8? The technique is called "artifacting"; it makes use of an otherwise unintended characteristic of a color display tube.

The highest resolution mode for the Atari is GRAPHICS 8. This mode includes 51,200 pixels (picture elements) arranged in a rectangular area extending 320 units horizontally and 160 units vertically. Unlike modes 2 through 7, GRAPHICS 8 normally allows only one color at two brightnesses. The background hue and luminance is determined by Color Register 2. "SETCOLOR 2,hue,lum" establishes Color Register 2. If left unset, it defaults to dark blue.

Color Register 1, which is set by SETCOLOR 1,hue,lum is used only to determine the luminance of graphics points. The hue of graphics points is determined by Color Register 2. A point is put on the screen by BASIC's PLOT or DRAWTO commands. To be effective, a COLOR 1 must preceed the first PLOT.

In order to understand artifacting, look closely at a color television screen. You will see closely spaced vertical stripes of red, green and blue phosphor elements. If these are illuminated equally, a white picture is produced. Unequal illumination produces all of the other colors.

GRAPHICS 7 Versus 8

A pixel in GRAPHICS 8 is the size of one half of one set of three color phosphor elements. If Color Register 1 luminance is relatively high and a graphics point is plotted at an odd valued horizontal coordinate, only the blue phosphor will be lighted. If a shape is drawn, taking care to use only odd values for X, the entire shape will be blue.

On the other hand, if only even values of X are chosen, the red and green phosphors will be selected. The proportion of these will depend on the background hue (Color Register 2) and the resulting "even-only" figure can range from red through brown to green. If both even and odd points are plotted, the background hue will result at a luminance specified by Color Register 1. Of course if neither even nor odd points are plotted, the result will be background hue and luminance. An interesting side effect occurs when an "even figure" overlaps an "odd figure" – the area of overlap will be clearly visible.

GRAPHICS 7 has pixels four times as large as GRAPHICS 8. That is, they are twice as wide and twice as high. Since artifacting requires skipping half the points in the horizontal direction, the resolution in that direction with GRAPHICS 8 approximately equals that of GRAPHICS 7. However, the *vertical* resolution of GRAPHICS 8 is twice that of GRAPHICS 7.

The program here is a simple demonstration of these principles. It first uses GRAPHICS 8 to draw three overlapping disks, the upper left using odd points only, the upper right using even points only, and the lower using all points. Note that the lower disk does have better horizontal resolution than the other two. After a delay, the program plots three similar figures in GRAPHICS 7. You will see that these disks have horizontal resolution equal to and vertical resolution worse than the previous upper disks.

You may explore the variety of colors possible with artifacting by changing the two SETCOLOR statements. Statements 4000 through 4120 may be inserted to cause all 960 possible SETCOLOR combinations to be displayed. This will step through all combinations as long as START is depressed.

Notes On The Sample Program

1000-1200	**Make an array of Y-values corresponding to X-values for circles.**
1300-1500	**Setup for the first display in GRAPHICS 8.**
1600-1900	**Draw upper-left figure using odd points only.**
2000-2300	**Draw upper-right figure using even points only.**
3000-3300	**Draw lower figure using even and odd points.**
3400	**A delay loop to allow viewing the first display.**
5000-5600	**Setup and draw first figure in GRAPHICS 7.**
5700-7300	**Draw second and third figures.**
9000-9100	**Another delay loop followed by a return to repeat.**
4000-4120	**Three nested loops stepping through all possible hues and luminances.**
4060	**The "arrow" is entered by typing ESC, CTRL, CLEAR**
4090	**This loops until either START, SELECT, or OPTION is depressed.**

```
100 REM ******************************
200 REM *                            *
300 REM * DEMONSTRATION OF           *
400 REM * ARTIFACTING IN GRAPHICS 8  *
500 REM *                            *
600 REM * BY BILL MOHN               *
700 REM *                            *
```

```
800 REM *                          *
900 REM ***************************
1000 DIM Y(79)
1050 ? "SHORT DELAY WHILE CALCULATING"
1060 ? "COORDINATES FOR CIRCLE....."
1100 FOR I=1 TO 79
1200 Y(I)=SQR(1600-(40-I)*(40-I)):NEXT I

1300 GRAPHICS 8:COLOR 1
1400 SETCOLOR 1,0,14:SETCOLOR 2,10,0
1500 ? "GRAPHICS MODE 8 WITH 'ARTIFACTIN
G'"
1600 FOR X=1 TO 79 STEP 2
1700 PLOT 80+X,45-Y(X)
1800 DRAWTO 80+X,45+Y(X)
1900 NEXT X
2000 FOR X=2 TO 78 STEP 2
2100 PLOT 150+X,45-Y(X)
2200 DRAWTO 150+X,45+Y(X)
2300 NEXT X
3000 FOR X=1 TO 79
3100 PLOT 120+X,110-Y(X)
3200 DRAWTO 120+X,110+Y(X)
3300 NEXT X
3400 FOR DLY=1 TO 1000:NEXT DLY
4000 REM INSERT FOR AUTOMATIC SEQUENCING

4010 FOR HUE=0 TO 15
4020 FOR LUM2=0 TO 14 STEP 2
4030 SETCOLOR 2,HUE,LUM2
4040 FOR LUM1=0 TO 14 STEP 2
4050 SETCOLOR 1,0,LUM1
4060 ? "}";"SETCOLOR 2,";HUE;",";LUM2
4070 ? "SETCOLOR 1,0,";LUM1
4080 FOR DLY=1 TO 50:NEXT DLY
4090 IF PEEK(53279)=7 THEN 4090
4100 NEXT LUM1
4110 NEXT LUM2
4120 NEXT HUE
5000 GRAPHICS 7
5100 ? "GRAPHICS MODE 7"
5200 COLOR 1
5300 FOR X=1 TO 79 STEP 2
5400 PLOT 40+X/2,23-Y(X)/2
5500 DRAWTO 40+X/2,23+Y(X)/2
5600 NEXT X
5700 COLOR 2
6000 FOR X=1 TO 79 STEP 2
6100 PLOT 75+X/2,23-Y(X)/2
6200 DRAWTO 75+X/2,23+Y(X)/2
6300 NEXT X
6400 COLOR 3
7000 FOR X=1 TO 79 STEP 2
7100 PLOT 60+X/2,55-Y(X)/2
7200 DRAWTO 60+X/2,55+Y(X)/2
7300 NEXT X
9000 FOR DLY=1 TO 1000:NEXT DLY
9100 GOTO 1300
```

©

Insight: Atari

Bill Wilkinson
Optimized Systems Software
Cupertino, CA

The major program for this month is perhaps the most exciting one to appear in this column to date. We will take advantage of Atari's modular software construction to define a set of *soft keys*, a concept that is marketed for real $$$ on some machines. The most obvious use of soft keys is in writing BASIC programs. Even with the abbreviations allowed by BASIC, wouldn't it be convenient to be able to use a single keystroke to get a disk directory listing? And, of course, when programming in assembly language there are certain character combinations that are repeated often enough to justify the use of soft keys (e.g., "),Y" or ".BYTE").

The techniques presented in the soft key program include how to "steal" the system's default I/O devices and adapt them for your own purposes. It might be worth your while to re-read my column on adding the "M:" driver, (**COMPUTE!**, January, 1982, #20, pg. 120) since I will be assuming your knowledge of some of the points made there.

For the BASIC user, the soft keys can be made truly "soft" – even to the point of allowing a running BASIC program to change the definition of what a soft key means. And, of course, there will be the usual set of tidbits for those who don't feel up to tackling the soft keys project.

An Announcement

As most of you probably know, magazine articles and columns are written months before they actually appear. As I write this in mid-February, my company (Optimized Systems Software, Inc.) and **COMPUTE!** are frantically engaged in getting a new book ready for publication. *Inside Atari DOS* will presumably have made its appearance by the time you read this. Now, for the first time, Atari users will have access to the listing of the File Manager System of Atari DOS 2.0S, the current version of Atari DOS. Besides the listings, there is a complete description of each major subroutine, complete with entry and exit parameters and error conditions.

The book is not complete in and of itself: you would still need Atari's listings of the OS ROMs and DUP to have access to all of the DOS secrets. But this book will tie together many loose ends.

Let me leave you with one caveat (don't I always?): the book assumes that the reader has at least a working knowledge of 6502 assembly language. The book is of most value if you would like to see how such a complex organism as a DOS is built.

The terminology "soft" keys refers to keyboard keys that may change "meaning" as desired by the user. The Atari keyboard keys which we will make "soft" include the characters "control-A" through "control-Z" (that is, what are normally graphic characters produced by holding down the control key while hitting one of the letter keys).

The keys will be made soft in a very flexible fashion: each of the 26 keystrokes may be defined in such a way that entering one of them will "fool" the Atari OS (and hence BASIC, etc.) into thinking that a sequence of one or more ordinary keys have been depressed. The phrase "one or more" is literal: there is no effective limit on the number of characters a soft key may represent.

As this program is written, there are some limitations. Only characters with an ATASCII value of 1 to 127 decimal ($01 to $7F hex) may be placed in the string. The CR (RETURN or End-Of-Line) character is thus not permitted (since its value is $9B hex, 155 decimal). Since lack of CR seems to me to be a major flaw, each zero ($00) byte in the string is converted so that OS sees a CR instead.

The reason that only the values from 0 to 127 are acceptable is that a byte with its most significant bit (MSB, $80 or 128) turned on is our signal that this byte is to be the last character in the soft key string. Obviously, you can rewrite this part, if you desire, so that some other means is used to designate the end of string (a preceding length byte or trailing zero byte are obvious alternatives). However, the method chosen is simple and seems adequate for most purposes.

For the BASIC user, the soft keys can be made truly 'soft'...

One more note before we get into implementation details: since there are times when you might really want the graphics character "hidden" by a soft key, I have designed an "escape" sequence. Pressing "control-comma" (normally the graphics heart character) signals to the soft key routine that the next character is *not* to be translated. Thus, even the heart may be generated by pressing control-comma twice.

The Nitty-Gritty

Program 1 shows the complete source of Easykey, the program which implements all the features mentioned above. The program is composed of five primary parts.

The first part, with line numbers in the 2000-2999 range, is used to hook the routine into Atari's OS. First, we search the Handler Table looking for the E: device. When we find it, we hold onto its address and put the address of our replacement driver in the table instead. Recall that the address in HATABS must be the address of the Handler Routine Table. We copy the current table (presumably the Atari default table, from ROM) to our NEWETBL (new E. table) and replace the entry point for the get-a-single-character routine with the address of our new routine (NEWEGETCH) less one (always required, see commentary on the M: driver in my January column). We then change LOMEM so BASIC won't wipe us out and exit.

The second part of this process is the new get-a-single-character routine for the screen editor (E:) device. Most of this code is copied directly from the OS ROMs, the only exceptions being the branches to locations in ROM and the call to the keyboard get-a-character routine (KGETCH).

The third part, NEWKGETCH (NEW Keyboard GET single CHaracter routine), is the heart of this whole process. Here is where individual keystrokes are actually interpreted from their hardware codes and characteristics to a more palatable ATASCII code. But here, also, is where the system is vulnerable to our machinations. Since nothing "downstream" of the keyboard handler (e.g., the rest of the E: driver, CIO, BASIC, etc.) knows what happened at the physical keyboard level, the calling routines will believe the keyboard handler no matter what it tells them.

Actually, NEWKGETCH is fairly simple. First it checks to see if it is already processing a soft key. If so, it simply hands the caller the next key of the soft key's string. If not, it goes and gets a real key from the keyboard. If that key is a heart, it simply gets the next real keyboard key and passes that back to the caller (our "escape" clause). Otherwise, if the keyboard key was not control-A through Control-Z, then the key is returned to the caller unchanged.

If the keyboard key was one of the definable soft keys, its value is used to index into a table of soft key string pointers. Here one last validity check is made: if the string pointer is zero, the keyboard key is returned to the caller and no soft key string handling occurs. If a string pointer is encountered, its address is placed in KPTR and used to access all the characters in the string.

Note that zero bytes are translated to $9B (CR) characters and that any character with its most significant bit on terminates the string (by zeroing the high order byte of KPTR).

The fourth part of this routine is simply the above-mentioned table of soft key string pointers. Note that we take advantage of the fact that the assembler places zeroes into .WORDs (or .BYTEs) which are undefined.

The last part, of course, consists of the actual strings. Note the flexibility here. Control-D (label SD), for example, includes the modified CR character ($00) as also its last character by simply turning on the MSBit, producing a code of $80.

Soft Keys: Using Them From BASIC

To conserve space and to show the flexibility of the soft key system, I included strings for only three keys: control-D, which causes a disk directory listing, control-S for "SETCOLOR," and control-P for "PRINT#." The simplest way to add more soft key strings is to put them into the source given (with labels "SA" through "SZ," as appropriate) and assemble the whole thing at an address appropriate for your system.

But you can add or change soft key strings dynamically from BASIC via POKEs, etc. Note that Easykey, as given here, reserves over 450 bytes for soft strings. Note also the addresses of the labels "STABLE" and "STRINGS." If you assemble your own copy of Easykey, be sure and note the addresses of these two labels and convert them to decimal if you intend to use dynamic soft keys.

...if you control these seven pointers, you control BASIC...

Program 2, the BASIC program, is a sample which will allow you to redefine all 26 soft keys to any string you like and then save the resulting definitions in a disk file for later use. Study the technique, and you should be able to produce any kind of soft keys you might want. The program fragment (Program 3) will allow the reloading of predefined sets produced by the previous program.

Inside Atari BASIC: Part 4

This month we will feature a short discussion of the various "tables" used by Atari BASIC and how to find them. Some of this material is well covered by some of the articles in *COMPUTE!'s First Book of Atari,* so if you are too impatient to wait for next month you can run out and buy the book. Next month we will begin to use the information we discover this month to "fool" BASIC into letting us do things it was never designed for. We begin:

When an Atari BASIC program is SAVEd to disk or cassette, there are only 14 bytes of zero page written out along with the main tables and program. These 14 bytes consist of seven two-byte pointers (in the traditional low-byte, high-byte form) which tell BASIC where everything is in the particular program being SAVEd (or later being LOADed). All the other important zero page locations (and there are over 50 of them) are regenerated and/or recalculated by BASIC anytime you type NEW or SAVE (or, for some locations, RUN, GOTO, etc.). The implication is that, if you control these seven pointers, you control BASIC so let's examine their names and functions. Table 1 gives a summary thereof.

The first thing you may note about this table is that some of the locations (indicated by asterisks) have duplicate labels. If you examine the mnemonic meanings, you will probably see why: the pointer can mean different things in different contexts. For example, the space pointed to by location $80 (decimal 128) is used for different purposes, depending on whether BASIC is currently working on entering a new line (it uses OUTBUFF) or executing an expression within a program (when it uses ARGOPS).

You might also note that I provided a list of more than seven pointers. The locations $8E and $90 are not SAVEd and reLOADed because they are always dependent on the current state of the program (i.e., whether it is RUNning, whether it has executed a DIM statement, etc.). They are included here for completeness: aside from the zero page locations (and the $600 page locations with BASIC A+), these pointers completely define BASIC's usage of the Atari computer's memory space. So now let's go into detail about what each of these pointers is used for.

Table 1: BASIC's Critical Zero-Page Pointers

Location Hex	Decimal	Mnemonic Label:	Which means:
80	128	LOMEN	pointer to LOw MEMory limit
80	128 *	ARGOPS	ARGument/OPerator Stack
80	128 *	OUTBUFF	syntax OUTput BUFFer
82	130	VNTP	Variable Name Table Pointer
84	132	VNTD	Variable Name Table Dummy end
86	134	VVTP	Variable Value Table Pointer
88	136 *	STMTAB	STateMent TABle
8A	138	STMCUR	CURrent STateMent pointer
8C	140	STARP	STring/ARray Pointer
--	---		
8E	142	ENDSTAR	END STring/ARray space
8E	142 *	RUNSTK	RUNtime STacK pointer
90	144	TOPRSTK	TOP of Runtime STacK space
90	144 *	MEMTOP	pointer MEMory TOP limit

We already noted that ARGOPS is used in expression evaluation. That is, whenever BASIC sees any kind of expression to be evaluated [e.g., 3*A+B or SIN(30) or 2^(LOG(4/EXP(Y*Z^3))-1/(Z^2.5+ATN(Z)) or even 1.25], it must put intermediate results and/or operators on a "stack." ARGOPS points to a 256 byte area reserved for both the argument stack and the operator stack. (What actually happens in Execute Expression is extremely complex and far beyond the scope of this article.) Since expression evaluation and program entry cannot occur at the same time, OUTBUFF shares this same 256 byte space. When a program line is entered, BASIC checks it for syntax and converts it to internal tokens, placing these tokens temporarily into this 256 byte buffer (before moving them into the appropriate place in the program, depending on the line number). Again, this process is complex, but the results have been documented here in prior columns and in such places as *De Re Atari* and *COMPUTE!'s First Book of Atari*.

VNTP and VNTD point to the Variable Name Table. In Atari BASIC and BASIC A+, only the first occurrence of a name causes an entry to be added to this table. Within the tokenized program, the name(s) are replaced by a "variable number" which refers to the name's position within the name table. The names are simply placed in this table one after the other, with no intervening bytes, and the end of a name is signaled by turning on the significant bit ($80, 128 decimal) of its last character. Note that the dollar sign on the end of a string name and the left parenthesis on the end of an array name *are* included in this table.

VVTP and ENDVVT define the limits of the Variable Value Table. Aside from the actual tokenized BASIC program, this is probably the most interesting of the tables. Each variable occupies eight bytes in this table, so the variable number token need only be shifted left three times to index to the proper location herein. In Part 5 of this series, we will delve into this table in depth, finding many ways to fool BASIC, but there is no room in this issue for more on the subject.

STMTAB defines the beginning of the tokenized program; and, since there is no proper label to refer to, we may consider that STARP defines the end of same. Again, I refer to previous parts of this series for details on the structure of tokenized lines. STMCUR is interesting because it normally points to the actual line currently being executed. This would be one way of implementing special "statements" in Atari BASIC; a USR call would cause the subroutine to use STMCUR to examine the rest of the line for variables, etc. But my comments on ease of use, etc., from last month still apply: I don't think this is really practical.

STARP is the last of the seven pointers that are SAVEd and LOADed. Actually, it is included only to point to the end of the program area. The string/array space is *not* SAVEd or LOADed (but see Al Baker's article on "Atari Tape Data Files" in *COMPUTE!'s First Book of Atari* for some tricky techniques which I may expand on in future columns). STARP and ENDSTAR define the limits of the string/array space. Atari BASIC is different from Microsoft-style BASICs in that arrays and strings are allocated from this space in the order they are DIMensioned and are not moved around relative to each other after that. Thus, if you code "DIM A$(100),B(3,3)" then you can use ADR(A$) + 100 as the address of B(0,0).

Finally, there is the run-time stack, defined by RUNSTK and TOPRSTK. When a GOSUB or FOR (or WHILE in BASIC A+) is encountered, the current "address" (consisting of the line number and statement offset within the line) must be "pushed" onto a stack to wait for the corresponding RETURN or NEXT (or ENDWHILE) to "pop" it off, so that the loop or mainline routine may continue where it left off. This stack thus expands and contracts as necessary while a program is running. Again, full details of how the stack is accessed can't be discussed here. In any case, the mechanism is relatively simple for GOSUB and WHILE; only FOR...NEXT presents some interesting problems.

Before we leave this topic for this month, we should note that when a program is SAVEd all seven pointers are "relativized" to zero. That is, each pointer has the value of LOMEM (which is also the first pointer) subtracted from it. Then when the program is LOADed, the current value of LOMEM is added to each pointer, thus allowing self-relocating BASIC programs. A side effect of this process is that the first pointer is thus always zero (actually two bytes of zero), and BASIC uses this fact as a self-check when LOADing: it assumes that any file which does *not* start with two zero bytes cannot be a BASIC SAVEd program.

Tidbit #1: Structured Programs

An often desirable construct within properly structured programs is this one:

1. IF <expression> THEN <procedure-1> ELSE <procedure-2>

Since BASIC doesn't support procedures, we will modify this to the more familiar-looking form:

2. IF <expression> THEN GOSUB <line-1> ELSE GOSUB <line-2> or, using BASIC A+,

3. IF <expression> : GOSUB <line-1>ELSE : GOSUB <line-2> : ENDIF

But still, the Atari BASIC programmer cannot use either of these forms. Take heart! There is a solution which is a logical replacement for 2., above:

4. ON <logical-expression> + 1 GOSUB <line-1>,<line-2>

Note that there is a subtle difference: where the IF allowed "expression," we now require "logical-expression." The reason is fairly obvious if you recall that a logical expression in Atari BASIC (e.g., A<B or B>=0 or A$<>B$) always evaluates to a one (true) or zero (false). By adding one (the "+1" in 4.) to a logical expression's value, we have a value of either one or two, something which ON...GOSUB is quite happy with since it GOSUBs to line-1 if the value is one and line-2 if the value is two.

If you really do have an "expression" to replace (e.g., IF A THEN...), simply change it into a logical expression by comparing it to zero, thus:

```
IF A THEN ...
becomes
IF (A<>0) THEN ...
which becomes
ON (A<>0) + 1 THEN ...
```

...we are going to let Atari's CLEAR-SCREEN character do all the work for us.

P.S.: If you want some structuring, but not too much, notice that the GOSUBs in 2. and 4. may be changed to GOTOs with similar effects.

Tidbit #2: A Bug in DOS 2.0S

DOS 2.0S and OS/A+ have an improvement which allows much faster disk reads and writes. When DOS detects that a large data transfer is about to take place, it drops into what is called *Burst I/O Mode.* However, when a file is opened for update (OPEN #1,12,...), burst I/O should not take place. DOS handles update writes correctly, but will often blow it on update reads. The following two, one-byte patches may be made and then DOS should be re-written to the disk (with INIT under OS/A+, with menu option "H" under Atari DOS 2.0S). Caution: do *not* apply these patches to any other versions of DOS!

from BASIC:	from DEBUG:
POKE 2596,144	C A24<90
	C AD5<IF
POKE 2773,31	('C' is the Change command in BUG.)

Tidbit #3: Clearing Memory (Revisited)

My thanks to Jerry White for permission to share his ideas on this with you. This concept is actually the result of a series of coincidences. Coincidence #1: a zero byte in screen memory is displayed as a space on the screen (not true on most machines, where $20–decimal 32–is the space character). Coincidence #2: the Atari CLEAR-SCREEN character (SHIFT-CLEAR or CTRL-CLEAR) is not subjected to most of the cursor range checks that other characters must go through. Coincidence #3: the code to clear the screen doesn't just clear one line 24 times (as does, for example, the Apple II's code); instead it simply starts at what it thinks is the lowest address being displayed and continues to the top of memory.

By now, it should be obvious that we are going to let the Atari's CLEAR-SCREEN character do all the work for us. The only thing we must do is fool it into believing that the "screen" is where we want it and is the size we want it.

CLEAR-SCREEN starts clearing at the location pointed to by $58 (88 decimal) and continues until one-byte short of the page pointed to by $6A (106 decimal). That is, it always stops clearing at location $xxFF, where xx is one less than the contents of $6A. So our memory clear program fragment looks something like this:

```
  LOWADDR = ???? : REM the lowest address to clear
  HIADDR = ????  : REM the highest address to clear
                   !! must end on xxFF boundary !!
* SVLOW1 = PEEK(88) : SVLOW2 = PEEK(89)
  SVHI   = PEEK(106)
  POKE 106,INT( (HIADDR + 1) / 256 )
  TEMP = INT(LOWADDR/256) : POKE 89,TEMP
  POKE 88,LOWADDR-256*TEMP
  PRINT CHR$(125); : REM this does the actual clear
  POKE 106,SVHI
* POKE 88,SVLOW1 : POKE 89,SVLOW2
```

Some *cautions* are in order (as usual): 1) The screen editor thinks that it really has cleared the screen and homed the cursor. For safety's sake, it is probably best to follow that code fragment with either a GRAPHICS statement or a real screen clear. 2) Since you can only specify the high (ending) address to the nearest page boundary, you have to be careful you aren't wiping something else out.

For once, though, *caution* number (1) has a good side effect. If you follow the program fragment given above with a GRAPHICS statement, then locations 88 and 89 are going to get recalculated anyway! So the lines marked with asterisks may be omitted in such cases.

P.S.: If you have BASIC A+, there is a much easier method, related to the way strings may be cleared. Given that you know LOWADDR and HIADDR, as in the fragment given above, you may clear the area via the following:

poke lowaddr,0 : move lowaddr,lowaddr + 1, hiaddr-lowaddr (And, wouldn't you know it, another *caution:* the system gets *very* unhappy if hiaddr = lowaddr.)

Next month, we will teach you how to have your Atari take over the entire Bell telephone system. All you need is 37 billion dollars, 25000 miles of #0000 gauge copper wire, and a toothpick. *Caution:* if you try this you musssssssssssssssssttttttt tabbbbbbbbbbbbeeeeee suuuuuurrrrrr.........asdf asetasyghxvnbaer6q23uqerngt1357 etaoin shrdlu

Program 1.

```
ØØØØ           Ø99Ø        .PAGE "        EQUATES"
               1ØØØ ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
               1Ø1Ø ;
               1Ø2Ø ; EASYKEY --
               1Ø3Ø ;   A program to ease repetitive typing
               1Ø4Ø ;   when using Atari BASIC, etc.
               1Ø5Ø ;
               1Ø6Ø ; Written by Bill Wilkinson
               1Ø7Ø ;             Optimized Systems Software
               1Ø8Ø ;
               1Ø9Ø ; for the May, 1982, issue of COMPUTE!
               11ØØ ;
               111Ø ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
               112Ø ;
               113Ø ;
               114Ø ; Equates to subroutines, etc., located
               115Ø ;   in the Atari OS ROMs
               116Ø ;
               117Ø ; CAUTION: these equates are for the
               118Ø ;   revision 'A' ROMs.
               119Ø ;
FCB3           12ØØ SWAP   =    $FCB3
FA88           121Ø ERANGE =    $FA88
ØØ6B           122Ø BUFCNT =    $6B
ØØ54           123Ø ROWCRS =    $54
ØØ55           124Ø COLCRS =    $55
ØØ6C           125Ø BUFSTR =    $6C
F6E2           126Ø KGETCH =    $F6E2
ØØ4C           127Ø DSTAT  =    $4C
Ø2FB           128Ø ATACHR =    $2FB
ØØ9B           129Ø CR     =    $9B
F66E           13ØØ EGETC2 =    $F66E
F67C           131Ø EGETC3 =    $F67C
ØØ63           132Ø LOGCOL =    $63
F9ØA           133Ø BELL   =    $F9ØA
F6AD           134Ø DOSS   =    $F6AD
F634           135Ø RETUR1 =    $F634
               136Ø ;
               137Ø ;
Ø31A           138Ø HATABS =    $31A
Ø2E7           139Ø SYSTEMLOMEM = $2E7
ØØØØ           149Ø        .PAGE
               15ØØ ;
               151Ø ; EQUATES UNIQUE TO THIS ROUTINE
               152Ø ;
ØØ22           153Ø QUOTE  =    $22       ; The " character
ØØØØ           154Ø NUL    =    Ø         ; A nul...which becomes a CR
               155Ø ;
ØØE6           156Ø ZTEMP1 =    $E6       ; shared with fltg pt routines
               157Ø ;
ØØFF           158Ø LOW    =    $FF
Ø1ØØ           159Ø HIGH   =    $1ØØ
ØØ8Ø           16ØØ KQUIT  =    $8Ø       ;MSBit says quit
               18ØØ ;
               181Ø ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
               182Ø ;
               183Ø ; CHOOSE THIS ORIGIN TO FIT YOUR SYSTEM !!
               184Ø ;
               185Ø ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
               186Ø ;
1FØØ           19ØØ ORIGIN =    $1FØØ
22ØØ           191Ø NEWLOMEM =  ORIGIN+$3ØØ
ØØØØ           192Ø        *=   ORIGIN
1FØØ           199Ø        .PAGE "        Hooking the driver into OS"
               2ØØØ ;
               2Ø1Ø ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
               2Ø2Ø ;
               2Ø3Ø ; The 'HOOKUP' routine --
               2Ø4Ø ;
               2Ø5Ø ;   this portion of the code simply hooks
               2Ø6Ø ;   our replacement driver into the E:
               2Ø7Ø ;   handler vector table.
               2Ø8Ø ;
               2Ø9Ø ;
               21ØØ ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
               211Ø ;
               212Ø HOOKUP
1FØØ A2ØØ      213Ø        LDX  #Ø          ;start at beginning
               214Ø LOOKLOOP
1FØ2 BD1AØ3    215Ø        LDA  HATABS,X
1FØ5 C945      216Ø        CMP  #'E         ; looking for the E: driver
1FØ7 FØØ7      217Ø        BEQ  EFND        ;got it
1FØ9 E8        218Ø        INX
1FØA E8        219Ø        INX
1FØB E8        22ØØ        INX              ; to next name
1FØC DØF4      221Ø        BNE  LOOKLOOP    ; keep looking
1FØE FØFØ      222Ø        BEQ  HOOKUP      ; actually, this is fatal error
               223Ø ;
               224Ø ; found the E driver
               225Ø EFND
1F1Ø BD1BØ3    226Ø        LDA  HATABS+1,X ; get LSB of addr
1F13 85E6      227Ø        STA  ZTEMP1
1F15 BD1CØ3    228Ø        LDA  HATABS+2,X ; and MSB
1F18 85E7      229Ø        STA  ZTEMP1+1
1F1A A9C7      23ØØ        LDA  #NEWETBL&LOW
1F1C 9D1BØ3    231Ø        STA  HATABS+1,X ; replace with our table
1F1F A91F      232Ø        LDA  #NEWETBL/HIGH
1F21 9D1CØ3    233Ø        STA  HATABS+2,X
               234Ø ;
1F24 AØØF      235Ø        LDY  #15         ; all bytes of table
               236Ø EMVLP
1F26 B1E6      237Ø        LDA  (ZTEMP1),Y ; get a byte of old tbl
1F28 99C71F    238Ø        STA  NEWETBL,Y ; to our table
1F2B 88        239Ø        DEY
1F2C 1ØF8      24ØØ        BPL  EMVLP       ; and do more
               241Ø ;
               242Ø ; E: handler vector table is moved
               243Ø ;
1F2E A942      244Ø        LDA  #NEWEGETCH-1&LOW
1F3Ø 8DCB1F    245Ø        STA  NEWETBLGC ; change the get character ptr
1F33 A91F      246Ø        LDA  #NEWEGETCH-1/HIGH
1F35 8DCC1F    247Ø        STA  NEWETBLGC+1
               248Ø ;
               25ØØ ; tables, etc. are corrected
               251Ø ;
               252Ø ; move lomem
               253Ø ;
1F38 A9ØØ      254Ø        LDA  #NEWLOMEM&LOW
1F3A 8DE7Ø2    255Ø        STA  SYSTEMLOMEM
1F3D A922      256Ø        LDA  #NEWLOMEM/HIGH
1F3F 8DE8Ø2    257Ø        STA  SYSTEMLOMEM+1
1F42 6Ø        258Ø        RTS
1F43           299Ø        .PAGE "        The replacement E: driver"
               3ØØØ ;
               3Ø1Ø ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
               3Ø2Ø ;
               3Ø3Ø ; Begin the actual E: replacement routine
               3Ø4Ø ;
               3Ø5Ø ; This routine replaces E:'s get-a-character
               3Ø6Ø ;   entry point.
               3Ø7Ø ;
               3Ø8Ø ;
               3Ø9Ø ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
               31ØØ ;
               311Ø NEWEGETCH
1F43 2ØB3FC    312Ø        JSR  SWAP        ;<< lines with comments starting
1F46 2Ø88FA    313Ø        JSR  ERANGE      ;<< with "<<" are copied from
1F49 A56B      314Ø        LDA  BUFCNT      ;<< without change from ROM
               315Ø ;
1F4B DØ26      316Ø        BNE  GOEGETC3    ; was just EGETC3
               317Ø ;
1F4D A554      319Ø        LDA  ROWCRS      ;<<
```

```
1F4F 856C    3200          STA  BUFSTR    ;<<
1F51 A555    3210          LDA  COLCRS    ;<<
1F53 856D    3220          STA  BUFSTR+1  ;<<
             3230 EGETC1
1F55 20791F  3240          JSR  NEWKGETCH ; What we really wanted to replace
1F58 844C    3250          STY  DSTAT     ;<<
1F5A ADFB02  3260          LDA  ATACHR    ;<<
1F5D C99B    3270          CMP  #CR       ;<<
             3280 ;
1F5F F015    3290          BEQ  GOEGETC2  ; was just EGETC2
             3300 ;
1F61 20ADF6  3310          JSR  DOSS      ;<<
1F64 20B3FC  3320          JSR  SWAP      ;<<
1F67 A563    3330          LDA  LOGCOL    ;<<
1F69 C971    3340          CMP  #113      ;<<
1F6B D003    3350          BNE  EGETC6    ;<<
1F6D 200AF9  3360          JSR  BELL      ;<<
             3370 EGETC6
1F70 4C551F  3380          JMP  EGETC1    ;<<
             3390 ;
             3400 GOEGETC3
1F73 4C7CF6  3410          JMP  EGETC3    ;becuz branch can't get there
             3420 ;
             3430 GOEGETC2
1F76 4C6EF6  3440          JMP  EGETC2    ;becuz branch can't get there
             3450 ;
1F79         3990          .PAGE "        The replacement KGETCH routine"
             4000 ;
             4010 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
             4020 ;
             4030 ; This is the routine that replaces the
             4040 ;   ROM-based KGETCH (Keyboard GET Character)
             4050 ;   routine.  This routine is designed
             4060 ;   especially for NEWEGETCH, but could be
             4070 ;   called in place of KGETCH if the user
             4080 ;   wished.
             4090 ;
             4100 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
             4110 ;
             4120 NEWKGETCH
1F79 ADC61F  4130          LDA  KPTR+1    ; msb of ptr to multikey string
1F7C F024    4140          BEQ  NOSTRING  ; but no string, so normal KGETCH
1F7E 85E7    4150          STA  ZTEMP1+1  ; need to put it in zero page
1F80 ADC51F  4160          LDA  KPTR      ; ditto the LSB
1F83 85E6    4170          STA  ZTEMP1
1F85 EEC51F  4180          INC  KPTR      ; bump ptr for next time
1F88 D003    4190          BNE  NEWK1
1F8A EEC61F  4200          INC  KPTR+1    ; MSB also if needed
             4210 NEWK1
             4220 ;
1F8D A000    4230          LDY  #0
1F8F B1E6    4240          LDA  (ZTEMP1),Y ; get next char in string
1F91 1003    4250          BPL  NEWK2     ; last char has MSB on
1F93 8CC61F  4260          STY  KPTR+1    ; so reset pointer & flag
             4270 NEWK2
1F96 297F    4280          AND  #$7F      ; ensure only 7 bits
1F98 D002    4290          BNE  NEWK3     ; except...
1F9A A99B    4300          LDA  #CR       ; null becomes CR
             4310 NEWK3
1F9C 8DFB02  4320          STA  ATACHR    ; this is what KGETCH does
1F9F 4C34F6  4330 GORETURN JMP RETUR1     ; seems silly, but it works
             4340 ;
             4350 ; no string waiting for us...get a real key
             4360 ;
             4370 NOSTRING
1FA2 20E2F6  4380          JSR  KGETCH    ; a real key
1FA5 844C    4390          STY  DSTAT     ; why??? just in case
1FA7 C900    4400          CMP  #0        ; control-comma, the heart
1FA9 F017    4410          BEQ  REALCTL   ; handled special
1FAB C91B    4420          CMP  #27       ; one more than control-Z
1FAD B0F0    4430          BCS  GORETURN  ; a regular key
             4440 ;
             4450 ; if here, we have control-A through control-Z
             4460 ;
1FAF 0A      4470          ASL  A
1FB0 A8      4480          TAY            ; doubled value used as index
1FB1 B9FE1F  4490          LDA  STABLE-2,Y ; -2 becuz control-A is
1FB4 8DC51F  4500          STA  KPTR      ; value of 1 instead of zero
1FB7 B9FF1F  4510          LDA  STABLE-1,Y
1FBA 8DC61F  4520          STA  KPTR+1    ; MSB of addr of string
1FBD F0E0    4530          BEQ  GORETURN  ; oops...no string for this key
             4540 ;
1FBF 4C791F  4550          JMP  NEWKGETCH ; and pass back first char of string
             4560 ;
             4570 ;
             4580 ; special handling for control-comma !!!!
             4590 ;
             4600 REALCTL
1FC2 4CE2F6  4610          JMP  KGETCH    ; it is used as an escape key
             4620 ;                       ; to allow real control chars !
             4630 ;
             4900 ;
             4910 ; MISCELLANEOUS RAM USAGE
             4920 ;
1FC5 0000    4930 KPTR   .WORD 0          ;master ptr to next char
             4940 ;
1FC7 0000    4950 NEWETBL .WORD 0,0       ;to be filled in
1FC9 0000
1FCB 0000    4960 NEWETBLGC .WORD 0,0,0,0,0,0
1FCD 0000
1FCF 0000
1FD1 0000
1FD3 0000
1FD5 0000
             4970 ;
1FD7         4990          .PAGE "        The pointer table"
             5000 ;
             5010 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
             5020 ;
             5030 ; The strings and String TABLE
             5040 ;
             5050 ; Each string in use has its address placed
             5060 ;   in the corresponding location in STABLE
             5070 ; Any control character which should not be
             5080 ;   translated should have $0000 as its
             5090 ;   string address in STABLE.
             5100 ;
             5110 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
             5120 ;   (first, get on even boundary)
1FD7         5130          *=   ORIGIN+$100
             5140 STABLE
2000 0000    5150          .WORD SA,SB,SC,SD,SE,SF,SG,SH,SI
2002 0000
2004 0000
2006 3420
2008 0000
200A 0000
200C 0000
200E 0000
2010 0000
}***ERROR -  5-UNDEFINED
2012 0000    5160          .WORD SJ,SK,SL,SM,SN,SO,SP,SQ,SR
2014 0000
2016 0000
2018 0000
201A 0000
201C 0000
201E 7F20
2020 0000
2022 0000
}***ERROR -  5-UNDEFINED
2024 8620    5170          .WORD SS,ST,SU,SV,SW,SX,SY,SZ
2026 0000
2028 0000
```

```
202A 0000
202C 0000
202E 0000
2030 0000
2032 0000
]***ERROR - 5-UNDEFINED
2034         5990        .PAGE "        The soft key strings"
             6000 ;
             6010 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
             6020 ;
             6030 ; The actual strings
             6040 ;
             6050 ; we need supply strings only for
             6060 ; the desired softkeys
             6070 ;
             6080 ; Note that inverse video is not
             6090 ; allowed, except that the last
             6100 ; character of a string has its
             6110 ; MSBit on, same as inverse video
             6120 ;
             6130 ; Note that zero bytes ('NUL') get
             6140 ; converted to RETURN ($9B) characters
             6150 ;
             6160 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
             6170 ;
             6200 SD .BYTE "CLOSE #7:OPEN #7,6,0,"
             6210    .BYTE QUOTE,"D:*.*",QUOTE
             6220    .BYTE ":FOR I=0 TO 256:GET #7,I"
             6230    .BYTE ":PRINT CHR$(I);:NEXT I"
             6240    .BYTE NUL+KQUIT
             6250 ;
             6260 SP .BYTE "PRINT ",KQUIT+'#
             6270 ;
             6280 SS .BYTE "SETCOLO",KQUIT+'R
             6290 ;
             6300 ; Note the methods of getting quoted strings
             6310 ; in .BYTE statements and turning on the
             6320 ; MSBits of a character
             7000 ;
             7010 ; finally, we set up for LOAD-AND-GO
             7020 ;
             7030      *= $2E0
             7040      .WORD  HOOKUP
             7050 ;
             9999    .END
```

Program 2.

```
1000 REM **********************************
1010 REM *
1020 REM * EASYLOAD --
1030 REM *       A BASIC PROGRAM THAT ALLOWS
1040 REM *       THE USER TO MAKE HIS/HER OWN
1050 REM *       SET OF "SOFTKEYS" FOR USE
1060 REM *       WITH THE "EASYKEY" PROGRAM
1070 REM *
1080 REM **********************************
1090 REM
1100 REM :BEFORE RUNNING THIS PROGRAM, BE SURE
1110 REM :THAT "EASYKEY" HAS BEEN LOADED AND
1120 REM :RUN.  YOU MAY VERIFY THIS BY CHECKING
1130 REM :THE VALUE OF PEEK(744) -- IT SHOULD
1140 REM :MATCH THE NEWLOMEM PAGE VALUE IN
1150 REM :THE LISTING OF EASYKEY
1160 REM :[  PEEK(744) = 34 IF EASYKEY IS
1170 REM :ASSEMBLED AS GIVEN HERE ]
1180 REM
1190 REM == FIRST, SET UP ADDRESSES, ETC. ==
1200 STABLE=8192:REM STABLE = $2000
1210 STRINGS=STABLE+(2*26)
1220 REM
1230 DIM KEY$(130)
1240 REM
1250 REM FIRST, CLEAR OUT OLD SOFTKEY DEFINITIONS
1260 REM
1270 FOR ADDR=STABLE TO STRINGS-1:POKE ADDR,0:NEXT ADDR
1280 REM
1290 PRINT CHR$(125);
1300 PRINT "When prompted, enter a softkey string"
1310 PRINT "for the given control-key."
1320 PRINT
1330 PRINT "Remember: no inverse keys and use"
1340 PRINT "control-comma (the heart) to request"
1350 PRINT "a RETURN key code.  Actual use of the"
1360 PRINT "Return key terminates the string for"
1370 PRINT "the given softkey."
1380 PRINT :PRINT "[just RETURN will undefine that key]"
1390 PRINT :PRINT
1400 ADDR=STRINGS:REM WHERE WE START STORING SOFTKEY STRINGS
2000 REM MAIN LOOP
2010 FOR KEY=0 TO 25
2020   PRINT "ConTRoL-";CHR$(65+KEY);:INPUT KEY$
2030   KLEN=LEN(KEY$):IF  NOT KLEN THEN 2290
2040   KEY$(KLEN)=CHR$(ASC(KEY$(KLEN))+128):REM SET MSB OF LAST CHARACTER
2050   IF ADDR+KLEN>=STABLE+512 THEN PRINT "too much defined!";CHR$(253):END
2070   REM SET ADDRESS OF STRING IN STABLE
2080   TEMP=INT(ADDR/256):POKE STABLE+KEY+KEY+1,TEMP
2090   POKE STABLE+KEY+KEY,ADDR-256*TEMP
2100   FOR KPT=1 TO KLEN
2110     POKE ADDR,ASC(KEY$(KPT))
2120     ADDR=ADDR+1
2130   NEXT KPT
2200   REM END OF PROCESSING FOR THAT KEY
2290   NEXT KEY
3000 REM **********************************
3010 REM
3020 REM NOW WE ALLOW YOU TO SAVE THIS NEWLY DEFINED SET ON DISK
3030 REM
3040 REM SIMPLY HIT BREAK AT THE QUESTION IF YOU DON'T WANT TO SAVE
3050 REM
3100 PRINT :PRINT :PRINT "What set of softkeys should we save"
3110 PRINT "  this definition under (1-999) ";:INPUT SET
3120 IF SET<1 OR SET>999 OR SET<>INT(SET) THEN 3100
3150 KEY$="D:SOFTKEY.":KEY$(LEN(KEY$)+1)=STR$(SET)
3200 OPEN #1,8,0,KEY$
3210 FOR KPT=STABLE TO ADDR
3220   PUT #1,PEEK(KPT)
3230   NEXT KPT
3250 CLOSE #1
3280 PRINT :PRINT "==== normal end ===="
3290 END
```

Program 3.

```
30000 REM **********************************
30010 REM
30020 REM THIS PROGRAM OR PROGRAM FRAGMENT IS
30030 REM INTENDED TO BE USED TO RELOAD ONE
30040 REM THE SETS OF SOFTKEYS CREATED BY
30050 REM "EASYLOAD".
30060 REM
30100 STABLE=8192:REM STABLE=$2000
30110 DIM NAME$(20)
30200 PRINT "What set of softkeys should be"
30210 PRINT "  reloaded from disk (1-999) ";:INPUT SET
30220 NAME$="D:SOFTKEY.":NAME$(LEN(NAME$)+1)=STR$(SET)
30250 OPEN #1,4,0,NAME$
30290 TRAP 30400
30300 FOR KPT=STABLE TO STABLE+512
30310   GET #1,KEY:POKE KPT,KEY
30320   NEXT KPT
30400 CLOSE #1
30500 REM COULD RETURN FROM SUBROUTINE, IF DESIRED
```

©

For people who would find it very difficult to enter commands or use the keyboard, the program below prints messages (or BASIC commands) on screen and allows the user to move a pointer to the desired word and enter it into the computer. It can accept input from either the numeric keypad or from a joystick. It is designed to work on a PET with the Graphics keyboard, but the central idea could be adapted to any computer.

Handicapped Programming

Hilton B. Souther
Lynchburg, VA

This program allows a person to build messages on the screen or could even be used to program. The input is made by pressing a button on a joystick or by pressing keyboard numeric pad no. 5.

When the program first comes up, it gives you an option of moving a pointer with the numeric pad or using the joystick; from then on, it responds based on the input. The program is set up for the joystick on the high order bits on the user port. If there is no joystick, all commands come from the numeric pad.

Most of the PET commands, except seldom used ones such as CMD, are displayed on the screen. A pointer will be shown opposite the various commands or BASIC words and the pointer can be moved up, down, right, or left until it is opposite the desired word. The user then pushes the button or the no. 5, depending on the mode of operation, and the word that is being indicated by the pointer is printed at the bottom of the screen. After the item is printed, the pointer is again activated for further entry. At the bottom of the screen where the message is being built, a vertical pointer indicates the next location for the next character location or word location. If one wishes to enter information not shown in the listing of BASIC words, put the pointer opposite the word "SCREEN" and enter. The word will print and then erase; a pointer at the top of the screen will then activate and will be pointing at the alphabet and numbers located across the top of the screen. The up arrow is used for the pointer; it can only move left or right; it will wrap around at the end. When the pointer is beneath the desired character, push the no. 5 key or button and the single character will be added to the message. To change the top of the screen to special characters, enter the greater than or less than sign. The line will change, and then you can enter the special characters. To return to the words, enter the @ symbol and the pointer will be back at the words.

When the message is completed or you wish to print on a printer at Device No. 4, enter the % symbol; the message or basic statement will go to the printer. The program will clear the array, present the screen again and allow the next message or statement to be entered. To prevent the screen from scrolling, there is a limitation of 240 characters. The program checks for the limit and, if reached, prints the message and continues.

The program presently goes to a printer; however, it could be changed to write to a disk or to the cassette, or even to push out to a modem and communicate with another computer.

There is sound with each movement of the pointers using the CB2 convention. All the computer needs for this program to function is the ability to close switches so a handicapped person could make it work. Of course, the necessary switches would have to be used in place of the joystick. I did not mention that to enter blanks, enter the shaded space. To delete characters, enter the right bracket and the message will decrease by one character. The program will run on an 8K machine, Original or Upgrade ROMs; however, the REMARKS will have to be deleted first. If you have a wedge in the 8K and try to load, it won't fit. After you load it on the 8K, you can't save unless you delete one line since you will get an out-of-memory message.

I have not stated in this article all of the line numbers and their functions. I think the REMARKS do a pretty good job of that. The sort for sorting the words is the SHELL sort. I read all of the variables first to count them and then dimension the array that size. The program could be changed by using new data statements. It is presently set up for 64 words on the screen in four rows. The program could be changed to calculate the number of words being used and adjust the various locations of the pokes to accommodate the different values. I did not try that since I was trying to accomplish something that might be useful for someone who cannot access the computer any other way.

If anyone wishes a copy of the program, I will make a copy from the original for a cost of $3.00. Please enclose a tape and SASE mailer.

Hilton B. Souther
115 Windingway Rd.
Lynchburg, VA 24502

```
16 BEG=1:REM HOUSEKEEPING
18 GOTO112
20 DI=1:PRINT"{HOME}{15 DOWN}":REM
    [HM][DN15]
24 GB=3:POKETR,0:ONDIGOSUB32,220
28 POKEK4,16:POKEK5,15:POKEK6,51:P
    OKEK5,0:POKEK4,0:POKEK6,0
30 GOTO24
32 POKEI,ASC(">"):FORJ=1TO10:NEXT:
    POKEE6,0
33 S9=0:FORJ=I+1TOI+9:IFPEEK(J)=32
    ANDPEEK(J+1)=32THENS9=J-1:
    GOTO37
36 POKEJ,PEEK(J)OR128:NEXT
37 IFPEEK(TR)=255THEN37
38 GOTO40
39 FORJ=I+1TOS9:POKEJ,PEEK(J)-128:
    NEXT:RETURN
40 IFPEEK(TR)=E4THENGOSUB39:POKEI,
    32:GOSUB88:I=I-40:RETURN
44 IFPEEK(TR)=E5THENGOSUB39:POKEI,
    32:GOSUB94:I=I+40:RETURN
48 IFPEEK(TR)=E3THENGOSUB39:POKEI,
    32:GOSUB100:I=I+10:RETURN
52 IFPEEK(TR)=E2THENGOSUB39:POKEI,
    32:GOSUB106:I=I-10:RETURN
54 IFPEEK(TR)=E1THEN64
60 RETURN
64 GOSUB39:FORJ=I+1TOS9:FR=PEEK(J)
    :IFPEEK(J)=32THENCT=CT+1:I
    FCT=2THEN80
68 IFPEEK(J)<>32THENCT=0
72 POKELO,FR:POKELN,32:LN=LN+1:POK
    ELN,30:LO=LO+1:E$=E$+CHR$(
    PEEK(J)+64)
74 NEXT:IFCT=0THENPOKEL0,32:POKELN
    ,32:LN=LN+1:POKELN,30:LO=L
    O+1:E$=E$+" "
78 IFLEFT$(E$,3)="SCR"THENGOSUB394
    :RETURN
80 REM ADD TO STRING AND MOVE THE ~
    POINTER TO THE NEXT POSITI
    ON
82 IFLEN(C$)+LEN(E$)=>240THENPT=10
    1:GOTO316
84 C$=C$+E$:E$="":RETURN
88 IFI=T3ORI=T2ORI=T1ORI=TTHENI=I+
    K-T+40
90 RETURN
94 IFI=K3ORI=K2ORI=K1ORI=KTHENI=I-
    (K-T)-40
96 RETURN
100 IF((I-8)+10)/40=INT((I+10)/40)T
    HENI=I-40
102 RETURN
106 IF((I-8)/40)=INT((I-8)/40)THENI
    =I+40
108 RETURN
112 PRINT"{CLEAR}{03 DOWN}"TAB(10)"
    {REV}READING VARIABLES":PR
    INT:REM[CL] [DN3]
114 K=33448:K1=K+10:K2=K+20:K3=K+30
    :BEG=1:W=0:K4=59467:K5=K4-
    1:K6=K4-3:K7=K4-8
116 READW$:IFW$<>"9999"THENW=W+1:GO
    TO116
118 I=32848:T=I:T1=T+10:T2=T+20:T3=
    T+30:DIMW$(W):LO=33488:LN=
    LO+40
122 RESTORE:FORP=0TOW-1:READW$(P):P
    RINT".";:W$(P)=W$(P)+" ":N
    EXT:P=0:GOTO138
128 DATAPEEK(,INPUT,COS(,GET,ASC(,E
    XP(,ATN(,LOG(,RND(,SGN(,SQ
    R(,CHR$(,LEFT$(
130 DATALEN(,MID$(,RIGHT$(,STR$(,VA
    L(,POS(,TAB(,FRE(,SYS,TI$,
    TI,USR,"DEF FN"
132 DATACLR,SAVE,CONT,LIST,LOAD,NEW
    ,RUN,VERIFY,DIM,ON,GOSUB,G
    OTO,IF,THEN,ABS(
134 DATAREM,RESTORE,RETURN,STOP,WAI
    T,SIN(,PRINT,SCREEN,AND,CL
    OSE,FOR,INT(,NEXT
136 DATANOT,OPEN,OR,POKE,READ,SPC(,
    STEP,TO,END,DATA,9999
138 TA=W-1:PRINT:PRINTTAB(52)"{REV}
    SORTING":PRINT
140 TC=TA
142 TC=INT(TC/2):IFTC>=1THEN146
144 GOTO172
146 REM SORT THE LIST
148 FORU=0TOTC:FORT9=UTOTA-TCSTEPTC
    :T0=T9:T$=W$(T9+TC)
150 IFT$>=W$(T0)THEN154
152 W$(T0+TC)=W$(T0):T0=T0-TC:IFT0=
    >1THEN150
154 W$(T0+TC)=T$:PRINT"'";:NEXTT9,U
    :GOTO142
```

```
162 P=Ø:PRINT"{HOME}{Ø2 DOWN}";:FOR
    Y=ØTO3:FORY1=ØTO15:REM[HM]
     [DN2]
164 PRINTTAB(Y*1Ø+1);W$(P):P=P+1:IF
    P>W+1THEN17Ø
168 NEXTY1:PRINT"{HOME}{Ø2 DOWN}";:
    NEXTY:PRINT:REM [HM] [DN2]

17Ø GOTO2Ø
172 CR$=CHR$(13):REM SET CARRAIAGE ~
    CONTROL
176 M$="{REV}4{OFF} FOR LEFT {REV}6
    {OFF} FOR RIGHT {REV}8{OFF
    OFF} UP AND {REV}2{OFF} DO
    WN":M1$="THE NUMBER {REV}5
    "
18Ø S=328Ø8:TR=515:E1=34:E2=42:E3=4
    1:E4=5Ø:E5=18:E6=525
182 INPUT"{CLEAR}{Ø2 DOWN}USING A J
    OYSTICK {REV}Y{OFF}ES OR {
    REV}N{OFF}O";J$:IFPEEK(5ØØ
    Ø3)THENTR=151:E6=158
186 IFJ$="Y"THENTR=59471:E1=63:E2=2
    23:E3=239:E4=127:E5=191
188 IFJ$="Y"THENM$="JOYSTICK":M1$="
    {REV}RED{OFF} BUTTON"
192 GOTO322
196 IFS=328Ø7THENPOKE328Ø8,32:S=S+4
    Ø:POKES,3Ø:RETURN
2ØØ POKES+1,32:POKES,3Ø:RETURN
2Ø4 IFS=32848THENPOKE32847,32:S=S-4
    Ø:POKES,3Ø:RETURN
2Ø8 POKES-1,32:POKES,3Ø:RETURN
212 POKEK7,Ø
216 GOSUB22Ø:RETURN
22Ø POKES-4Ø,PEEK(S-4Ø)OR128
221 IFPEEK(TR)=255THEN221
224 GOSUB3Ø4:GOSUB227:IFPEEK(TR)=E1
    THENPT=PEEK(S-4Ø):GOSUB266
    :GOSUB23Ø
226 GOTO228
227 IFPEEK(S-4Ø)>127THENPOKES-4Ø,PE
    EK(S-4Ø)-128:RETURN
228 POKEE6,Ø:RETURN
23Ø REM
232 TB=LEN(C$):IFLEN(C$)=ØTHENRETUR
    N
234 IFPT=ØTHENRETURN
238 IFASC(RIGHT$(C$,1))=1Ø1THENC$=L
    EFT$(C$,LEN(C$)-1):GOTO288

242 IFPT=6ØORPT=62THEN262
244 IFPT=29ANDLEN(C$)=ØTHENRETURN
246 IFPT=29ANDLEN(C$)=1THENPOKELN,3
    2:POKELO-1,32:GOSUB398:RET
    URN
25Ø IFPT=29THENPOKELN,32:LN=LN-1:PO
    KELN,3Ø:POKELO-1,32:LO=LO-
    1:GOSUB4ØØ
252 IFPT<>29ANDPT<>32THENPOKELO,PT:
    POKELN,32:LN=LN+1:LO=LO+1:
    POKELN,3Ø
256 IFPT=32THENPOKELO,23Ø:LO=LO+1:P
    OKELN,32:LN=LN+1:POKELN,3Ø

262 FORU=1TODL:NEXT:RETURN
266 IFPT=ØTHENDI=1:POKES,32:RETURN
27Ø IFPT=62THENGOSUB388:GOSUB392:PO
    KES,3Ø:RETURN
272 IFPT=6ØTHENGOSUB388:GOSUB39Ø:PO
    KES,3Ø:RETURN
276 IFPT=1Ø2THENPT=32:C$=C$+CHR$(PT
    ):RETURN
278 IFPT<>94ANDPT<>224ANDPT<>29THEN
    C$=C$+CHR$(PT+64):GOTO28Ø
28Ø IFPT=94THENC$=C$+CHR$(255)
284 RETURN
288 POKEE6,Ø:PRINT:PRINT"{CLEAR}{Ø7
     DOWN}YOUR MESSAGE IS":PRI
    NTC$:REM[CL] [DN7]
292 ML=1:OPEN4,4:CMD4:PRINTC$
296 PRINT#4:CLOSE4
3ØØ IFML=1THENLO=LO-LEN(C$):LN=LO+4
    Ø:L=Ø:P=Ø:C$="":PRINT"{CLE
    CLEAR}":P=Ø:GOTO372
3Ø4 IFPEEK(TR)=E2THENGOSUB227:S=S-1
    :GOSUB196
3Ø8 IFPEEK(TR)=E3THENGOSUB227:S=S+1
    :GOSUB2Ø4
31Ø RETURN
314 PRINT
316 PRINT"{CLEAR}   YOUR MESSAGE LE
    NGTH HAS REACHED THE  MAXI
    MUM FOR THIS ONE, ";
318 PRINT" PRINTING AND THENCONTINU
    ING.":GOSUB262
32Ø DL=2ØØØ:GOSUB262:ML=1:GOTO288
322 PRINT"{CLEAR}{Ø3 DOWN}":REM[CL]
    [DN3]
324 PRINT"YOU WILL BE PRESENTED WIT
    H A LIST OF "
326 PRINT"WORDS, OPPOSITE THE WORDS
     WILL BE A >
328 PRINT"TO MOVE THE > PUSH THE
33Ø PRINTM$
332 PRINT"WHEN THE > POINTS TO THE ~
    PHRASE
334 PRINT"YOU WISH THEN PUSH "M1$
336 PRINT"TO CHANGE THE > FROM THE ~
    PHRASES TO
338 PRINT"CHARACTERS, PUSH "
34Ø IFE5<>191THENPRINTRIGHT$(M1$,9)
    "{OFF} WHEN OPPOSITE
342 IFE5=191THENPRINTCR$;M1$"{OFF} ~
    WHEN OPPOSITE
344 PRINT"THE WORD SCREEN AND THEN ~
    YOU WILL HAVE
```

```
346 PRINT"THIS POINTER ^ AT THE TOP
     OF THE SCREEN
348 PRINT"ONLY MOVE LEFT OR RIGHT W
    ITH THIS ^
350 PRINT"WHEN UNDER THE DESIRED CH
    ARACTER PUSH
352 PRINTM1$
354 PRINT"TO CHANGE THE LINE DISPLA
    Y AT THE TOP
356 PRINT"ENTER THE'<' OR THE '>' W
    HEN THE ^ IS   {REV}UNDER{
    OFF} THEM
358 PRINT"TO  DELETE A CHARACTER, E
    NTER THE ']'
360 PRINT"TO ENTER A SPACE ENTER TH
    E &.
362 PRINT"TO PRINT OUT ENTER THE '%
    ' SYMBOL
364 PRINT"TO RETURN TO THE WORDS EN
    TER THE '@'    SYMBOL. ALL
     ITEMS POINTED TO
365 PRINT"WILL BE IN {REV}REVERSE F
    IELD{OFF}"
366 PRINT"TO START PUSH ";M1$
368 IFPEEK(TR)<>E1THEN368
370 FORU=1TO500:NEXT
372 PRINT"{CLEAR}";
374 A1$="<.,;:!#$'&\()+-=?@%&]<
376 A$=">ABCDEFGHIJKLMNOPQRSTUVWXYZ
    0123456789@]&":PRINTA1$;
378 IFBEG=1THENDIMA%(LEN(A$)),A1%(L
    EN(A1$)+1):TY=32768
380 IFBEG=1THENFORJA=TYTOTY+LEN(A1$
    ):A1%(JA-TY)=PEEK(JA):NEXT
    :A1%(JA-TY-1)=34
382 IFBEG=1THENPRINT"{CLEAR}";A$:FO
    RJA=TYTOTY+LEN(A$):A%(JA-T
    Y)=PEEK(JA):NEXT
384 IFBEG=1THENBEG=2:GOTO162
386 PRINT:GOTO162
388 PRINT"{HOME}";:FORU6=1TO40:POKE
    U6+32767,32:NEXT:RETURN
390 FORU6=TYTOTY+LEN(A$):POKEU6,A%(
    U6-TY):NEXT:RETURN
392 FORU6=TYTOTY+LEN(A1$):POKEU6,A1
    %(U6-TY):NEXT:RETURN
394 DI=2:POKEI,32:POKES,30:FORY9=1T
    OLEN(E$):POKELO,32:LO=LO-1
    :NEXT
396 POKELO,32:POKELN,32:LN=LN-LEN(E
    $):POKELN,30:E$="":RETURN
398 LO=LO-1:LN=LO+40:POKELN,30:C$="
    ":RETURN
400 C$=LEFT$(C$,LEN(C$)-1):RETURN
```

©

We have ways...

to make it talk!

Give your computer the magic of speech with an ECHO™ Speech Synthesizer!

There are four new models to choose from: the ECHO 80 (TRS-80 models I & III), ECHO 100 (S-100), the ECHO GP (general-purpose, serial/parallel) and the ECHO SBC (single board computer). These join the popular ECHO][(Apple).

ECHO Speech Synthesizers use LPC synthesis pioneered by Texas Instruments, combined with a phoneme-based operating system to provide any vocabulary you desire while using a minimum of your computer's memory.

The TEXTALKER Speech Generator, standard with all ECHOs except the SBC, translates normal English text to speech automatically.

Simple commands allow you to select from:

- 63 different pitch levels
- Words spoken monotonically or with intonation
- Fast or slow speech output
- Entire words pronounced or spelled letter by letter
- Different volumes
- Spoken punctuation if desired

Speech applications are virtually unlimited, ranging from education to games to aiding the handicapped. The flexibility and low price of ECHO Speech Synthesizers make them the logical choice when adding speech to your system.

For more information see your dealer or contact:

SEC **STREET ELECTRONICS CORPORATION**
3152 E. La Palma Avenue, Suite D
Anaheim, CA 92806 Telephone: 714/632-9950

New Products

ComputerTown International To Offer New Services

ComputerTown, USA!, a grass-roots computer literacy network sponsored by the National Science Foundation, offers guidance and support to anyone who wishes to learn about computers on a grass-roots, community-based level. Under its NSF grant, CTUSA! has fostered ComputerTown projects across the US and abroad, and is currently completing a book-length "Implementation Package" outlining the details of starting and conducting a ComputerTown project.

Funded until October, 1983, CTUSA! has proven so effective that an on-going project, ComputerTown International (CTI) has now come to life. The purpose of CTI is to build on the knowledge and resources gained by ComputerTown, USA! and to keep the communication channels open after the NSF funding ends. CTUSA! and CTI will both operate from the offices of People's Computer Company in Menlo Park, California, and most of the staff will overlap.

Three initial services are planned by ComputerTown International;

1) *A Speakers' List* will be developed and made available to interested groups or individuals. This list will be comprised of qualified individuals willing to speak on computer literacy and related subjects at locally organized seminars and events.

2) CTI is organizing *Two Teams of Specialists* – teachers and consultants, who are expert in the theory and practice of public access computer literacy and informal education. While these services will be offered on a fee basis, this will not decrease the free support and advice offered under the NSF grant, but will enable CTI to provide a depth of assistance and instruction impossible on a complimentary basis.

3) *The ComputerTown Bulletin* has proven an invaluable vehicle of communication among the international network of people and organizations involved in computer literacy. As of March, 1982, the *Bulletin* will no longer be funded by the National Science Foundation. In order to continue publication, ComputerTown International seeks individuals or businesses willing to sponsor one issue of the *Bulletin.* In exchange, the *Bulletin* will devote two pages to news and information about the sponsor's products and services and will acknowledge the sponsorship in print. Sponsors are encouraged to print extra copies for their own use as mailing and handout materials. Six thousand bi-monthly issues of each sponsored *Bulletin* will continue to be offered free of charge to anyone with an interest in computer literacy.

For further information about ComputerTown International and its services, write:

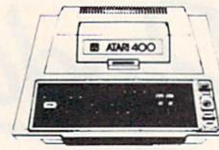
ComputerTown International
P.O. Box E
Menlo Park, CA 94025

Time Bomb Develops New Language Skills

A new educational game, Time Bomb, has been released by

ATARI HOME COMPUTERS

ATARI



ATARI 800

16K . . . $679
32K . . . $749
48K . . . $819

ATARI 400

16K. . . . $329
32K. . . . $478
48K. . . . $555

410 Recorder	$76.00
810 Disc Drive	$449.00
822 Printer	$269.00
825 Printer	$629.00
830 Modem	$159.00
820 Printer	$269.00
850 Interface	$169.00
New DOS 2 System	$29.00

PACKAGES

481 Entertainer	$83.00
482 Educator	$130.00
483 Programmer	$57.00
484 Communicator	$344.00

ATARI HOME COMPUTER PROGRAMS

Home Office

CX404 ATARI Word Processor	$119.00
CX8102 Calculator	$29.00
CX412 Dow Jones Investment Evaluator	$99.00
CX4109 Graph It, Joystick optional	$17.00
CX4104 Mailing List	$20.00
CX4115 Mortgage & Loan Analysis	$13.00
CX406 Personal Financial Management System	$59.00
CX4103 Statistics 1	$20.00
CX8107 Stock Analysis	$20.00
CXL4015 TeleLink 1	$23.00

Home Study

CX4101 An Invitation to Programming 1	$20.00
CX4106 An Invitation to Programming 2	$23.00
CX4107 Biorhythm	$13.00
Conversational Languages (ea.)	$46.00
CX4121 Energy Czar	$13.00
CX4114 European Countries & Capitals	$13.00
CX4108 Hangman, Joystick optional	$13.00
CX4102 Kingdom	$13.00
CXL4007 Music Composer	$47.00
CX4123 Scram, uses joystick	$20.00
CX4112 States & Capitals	$13.00
CX4110 Touch Typing	$20.00

Home Entertainment

PAC MAN	$35.00
Centipede	$35.00
Caverns of Mars	$32.00
CXL4013 Asteroids	$35.00
CXL4004 Basketball	$27.00
CX4105 Blackjack	$13.00
CXL4009 Computer Chess	$33.00
CXL4012 Missile Command	$35.00
CXL4008 Space Invaders	$35.00
CXL4011 Star Raiders	$42.00
CXL4006 Super Breakout	$33.00
CXL4010 3-D Tic-Tac-Toe	$27.00
CXL4005 Video Easel	$24.00

Programming Languages and Aids

CXL4003 Assembler Editor	$47.00
CXL4002 ATARI BASIC	$47.00
CX8126 ATARI Microsoft BASIC	$70.00
CXL4018 PILOT	$72.00
CX405 PILOT	$105.00

CX30 Paddle	$18.00
CX40 Joy Stick	$18.00
CX853 16K RAM	$89.00
Microtek 16K RAM	$75.00
Microtek 32K RAM	$159.00
Ramdisk (128K)	$539.00
Inteck 48K Board	$249.00
One year extended warranty	**$50.00**

THIRD PARTY PROGRAMS

ATARI Program Exchange:

Eastern Front '41	$25.50
Avalanche	$15.50
Outlaw	$15.50
747 Landing Simulation	$15.50
Babel	$15.50
Dog Daze	$15.50
Downhill	$15.50
Attack!	$15.50
Blackjack-Casino	$15.50
Reversi II	$15.50
Domination	$15.50
Solitare	$15.50
Disk Fixer	$15.50
Supersort	$15.50
Data Management	$15.50
Chameleon	$15.50
Instedit	$15.50
Insomnia	$15.50
My First Alphabet	$25.50
Mapware	$18.00
Number Blast	$11.50
Family Cash Flow	$15.50
Weekly Planner	$15.50
Bowler's Data Base	$13.00
Banner Generator	$11.50

Crystal Software	
Bermuda Fantasy	$26.00
Beneath Pyramids	$20.00
Galactic Quest	$26.00
House of Usher	$20.00
Forgotten Island	$26.00
Haunted Palace	$33.00
Compumax (Acct. Rec., Gen. Ledger, (Inventory, Payroll, ea.)	$110.00
Visicalc	$169.00
Letterperfect (Word Processor)	$109.00
Ricochet	$14.50
Crush Crumble & Chomp (cassette or disk)	$24.00
Star Warrior (cassette or disk)	$29.00
Rescue at Rigel (cassette or disk)	$24.00
Datestones (cassette or disk)	$16.00
Invasion Orion (cassette or disk)	$18.50
Mission Asteriod	$22.00
MouskATTACK	$31.00
The Next Step	$34.00
Softporn	$27.00
Wizzard & Princess	$29.00
K-BYTE Krazy Shoot Out (ROM)	$39.00
Protector (Disk 32K)	$32.00
Jaw Breaker (on line disk)	$27.00
Ghost Hunter (cassette)	$24.00
Ghost Hunter (disk)	$30.00

PRINTERS

Centronics 739-3	$619.00
Centronics 739-1	$519.00
Diablo 630 Special	$1799.00
Epson	
MX70	$359.00
MX80	$469.00
MX80FT	Call
MX100	Call
NEC	
8023	$549.00
7730	Call
7720	Call
7710	Call
Okidata	
82A	$499.00
83A	$769.00
84	$1129.00
Citoh Starwriter	
F10-40 CPS	$1469.00
F10-55 CPS	Call
Paper Tiger	
445G	$699.00
460G	$899.00
560G	$1129.00
Talley	
8024-7	$1399.00
8024-L	$1629.00
MPC Apple Parallel Board & Cable	$69.00

XEROX®

Xerox 820	
System I 5¼"	$2450.00
System II 8"	$2950.00
CPM 5¼"	$169.00
Word Processing	$429.00
Super Calc	$269.00

Texas Instruments

TI-99/4A $299

PHA 2100 R F Modulator	$43.00
PHP 1600 Telephone Coupler	$179.00
PHP 1700 RS-232 Accessories Interface	$179.00
PHP 1800 Disk Drive Controller	$239.00
PHP 1850 Disk Memory Drive	$389.00
PHP 2200 Memory Expansion (32K RAM)	$319.00
PHP 1100 Wired Remote Controllers(Pair)	$31.00
32K Expansion	$329.00
PHP Printer Solid State	$319.00
PHM 3006 Home Financial Decisions	$26.00
PHM 3013 Personal Record Keeping	$43.00
PHD 5001 Mailing List	$60.00
PHD 5021 Checkbook Manager	$18.00
PHM 3008 Video Chess	$60.00
PHM 3010 Physical Fitness	$26.00
PHM 3009 Football	$26.00
PHM 3018 Video Games I	$26.00
PHM 3024 Indoor Soccer	$26.00
PHM 3025 Mind Challengers	$22.00
PHM 3031 The Attack	$35.00
PHM 3032 Blasto	$22.00
PHM 3033 Blackjack and Poker	$22.00
PHM 3034 Hustle	$22.00
PHM 3036 Zero Zap	$18.00
PHM 3037 Hangman	$18.00
PHM 3038 Connect Four	$13.00
PHM 3039 Yahtzee	$22.00
Tombstone City 21st Century	$34.00
Munch Man	$34.00
T I INVADERS	$34.00
CAR WARS	$34.00

All items subject to availability and price change.

computer mail order west

CALL TOLL FREE **800-648-3311**

IN NEVADA, CALL (702) 588-5654

P.O. BOX 6689, STATE LINE, NEVADA 89449

Franco
Habla
Espanol

Program Design, the Greenwich, Connecticut firm that specializes in the design, development, and marketing of educational and game software for microcomputers.

Time Bomb is a version of the pen-and-pencil game Hangman. The computer chooses at random a secret word from a list of hundreds of words. The player is told the number of letters in the secret word, then asks the computer if certain letters appear in the word. After the computer shows if the letters are present, the player tries to guess the word. Every wrong guess shortens the fuse attached to a large bomb. The player must guess the secret word before the bomb goes off.

Some of the secret words are well known to all. Others are less common, or consist of unusual letter combinations that make guessing them difficult. There are two separate games, each with their own word lists. In addition, the disk version allows the user to add his or her own word lists.

Time Bomb was written by Dr. Dean Victor, author of Mini-crossword, AstroQuotes, and several other PDI programs. The program uses high resolution and player/missile graphics, and presents a challenge to both children and adults.

Time Bomb is available for use on Atari 400/800 computers with a memory of at least 16K. The cassette version retails for $16.95, the disk version for $23.95.

Program Design, Inc.
11 Idar Court
Greenwich, CT 06830
(203)661-8799 John Victor
(203)792-8382 Laurie Hall

Atari Sponsors Summer Computer Camps

Atari, Inc., will conduct eight camp sessions this summer, two in each of four locations, for 10 to 18-year olds interested in computers.

The camp sessions will each last four weeks. Atari Computer Camp sessions will begin in late June or early July, and will be conducted on school and university campuses in the northeast, southeast, midwest and west. Day-to-day operation of the camps will be handled by Specialty Camps, Inc., an organization with some 25 years of experience in running theme and traditional camps. Atari is designing their own curriculum for the camps under the direction of Robert A. Kohn who has been involved with computers and education for the past 15 years. They will recruit and train their own instructors, many of whom will be professional educators.

While the formal instruction sessions will last for two hours each day, all of the computers and software will be available to campers during their free time. The daily schedule will also include traditional summer camp activities.

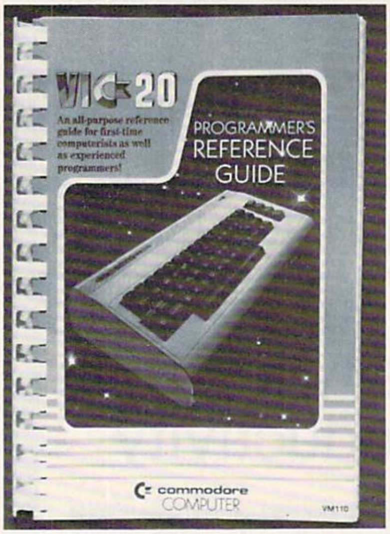
Teaching sessions will be limited in size, with one instructor for every five to six campers. There will be one computer for every two campers, since it has been Atari's observation that computer learning is enhanced when people work together on computer projects.

Equipment used will be Atari 400 and 800 Home Computers.

Atari, Inc.
1265 Borregas Ave.
P.O. Box 427
Sunnyvale, CA 94086

VIC 20 Programmers Reference Guide

The new VIC 20 *Programmers Reference Guide* is now available from Commodore Business Machines, Inc. Designed for use by first-time computerists as well as experienced programmers, the Programmers Reference Guide sells for $16.95 and provides complete information about the programming of Commodore's VIC 20 home computer. Nearly 300 pages, the



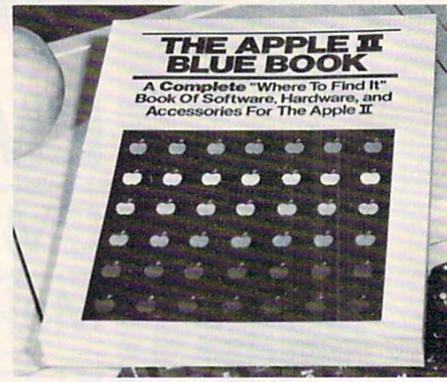
guide includes illustrations, instructions, charts and programs, as well as a schematic of the VIC 20.

Commodore Business Machines, Inc.
Computers Systems Division
681 Moore Road
King of Prussia, PA 19406
(215)337-7100

New Blue Book Available For The Apple Computer

WIDL Video, Chicago, publisher of the Apple Directories, has released the new 2nd edition of *The Apple II Blue Book.* The Blue Book is a master directory of software, hardware, peripherals, and information for the Apple II Computer. It gives Apple users a complete "where to find it" guide to available software and also includes a directory of hardware, boards, peripherals, and accessories. The meaty, 400 page Blue Book is loaded with useful information including over 5,000 software and hardware listings and more than 750 software and hardware producers.

The Software Section of the Blue Book contains program



listings for every application and features business, games, and educational software. Also included are special interest sections featuring word processing programs, graphics software, and data base management systems for the serious Apple user.

The Hardware section of the Blue Book contains new devices that interface with the Apple II Computer. There are manufacturers listed along with product information photos, price

ECLECTIC SYSTEMS CORPORATION
Order TOLL FREE 1-800-527-3135

ECLECTIC SYSTEMS

P.O. Box 1166 16260 Midway Road
Addison, TX 75001 (214) 661-1370

Authorized Commodore service center

Repair of the complete line of Commodore products

In a hurry? Check our modular exchange program

HARDWARE:	
CBM 8032 Computer, 80 Column	$1095
CBM 8050 Disk Drive	1340
CBM 4032 Computer, 40 Column	995
CBM 4040 Disk Drive	995
CBM 4022 Printer	649
CBM VIC 20 Computer	263
CBM VIC 1515 Printer	355
CBM 8300P (DIABLO)	1925
CBM VS100 Cassette	68
PET to IEEE Cable	33
IEEE to IEEE Cable	39

SOFTWARE:	
OZZ	$299
Wordcraft 80	299
The Manager (Data Base)	240
Wordpro 5+	329
Wordpro 4+	329
VISIC ALC	175
MUMPS for SuperPET	299
CMS Accounting System	call
Assembler Development Package	77
BASF Diskette, Box of 10	30

ANS MUMPS Programmer's Reference Manual	$17.50
MUMPS Pocket Guide	2.00
Computer Programming in ANS MUMPS	17.50
RS232 Interface for Commodore VIC 20	39.95
EIO has: two serial asynchronous RS232 ports, two parallel ports with handshaking, one shift register, two 16 bit timers, and room for two optional buffer IC's.	188.00
EIO-C Alternate character generator ROM board for screen display allows you to display characters or graphics of your choice. Alternate characters are soft selectable. EIO board required.	Call for price
EIO-TX Terminal ROM for EIO firmware to turn your CBM into a communications terminal. Store and transmit from disk to remote host and terminal. EIO board required.	49.95
EIO-RS232 Cable for EIO board.	28.00

BOOKS:	
Commodore Software Encyclopedia 2nd edition	9.95
The PET Revealed	19.95
Library of PET Subroutines	19.95
PET Interfacing	16.95
PET Basic	12.95
PET and IEEE 488 Buss	15.00

OTHER:	
EPROM Burner for CBM "burns 2716, 2732 & 2532"	89.95
Software for EPROM Burner on CBM	15.95
D.C. Hayes Smart Modem	249.00

ESC-100 RS232 interface manual selector —switch select between device A, B, or C; switching 25 conductors. Requires no input power and uses receptacle type DB25S. 140.00

ESC-120 M43 TTY-EIA/CURRENT LOOP interface unit—converts TTL level signals from the M43 to EIA or 20ma outputs. Convert EIA or 20ma inputs back to TTL level signals for the M43. Single printed circuit mounts inside the M43. Comes with interface cable. 65.00

ESC-140 RS232 standard interface cable kit. Build your own cable.
7 conductor 16.00
12 conductor 19.95

ESC-150 The Electronic Switch/Poller allows a single computer serial port to selectively communicate with an assortment of peripheral devices. As many as seven serial asynchronous and six parallel devices may be selected. This concept effectively creates a simplified multi-drop, polled environment. The computer maintains control of the network and may seize any of the peripherals by transmitting a simple escape sequence to the "ESC-150". 488.00

ESC-170 RICKETYMETEr-Indispensable hand-held tool for trouble-shooting, checkout and installation of RS232 type serial line communications systems. 69.95

ESC-180 RICKETYTRAp is a small hand-held device which when interposed in an EIA/RS232 line can monitor and trap on any specific ASCII, binary, or control character. The RICKETYTRAp can also monitor and trap on any specific range of characters bit selectable. This device can also check for type of parity being sent and proper framing. The RICKETYTRAp is switch selectable for 1 or 2 stop bits and can operate at speeds of 150 to 9600 baud. Call for price

ESC-200 PREDITOR is a small microprocessor with ROM firmware, RAM, and 8 RS232 asynchronous serial input/output ports. The PREDITOR is available in either stand-alone, self-contained, or rack-mounted models. The PREDITOR can be programmed by firmware to function as a prompter and editor in a distributed processing network, transforming 4 dumb asynchronous terminals into 4 intelligent terminals communicating with 4 computer asynchronous input/output ports. Call for price

ESC-861M The DATA CONCENTRATOR accepts data from a multitude of inputs—printers, CRTs, parallel ports, status lines, etc.—then transmits the composite (multiplexed) data stream down a high speed synchronous serial line to a remote DATA CONCENTRATOR. The remote DATA CONCENTRATOR separates (demuxes) the composite data into its original form to be transmitted to the computer ports. Alternately, multiple computer port data is multiplexed, transmitted down the high speed synchronous serial line, demuxed, and finally dispatched to the individual terminals. 850.00

Order TOLL FREE 1+800-527-3135

10 AM to 4 PM CDT Monday through Friday

Texas residents call 1+214-661-1370

VISA, MASTER CHARGE, MONEY ORDERS, AND C.O.D. "Certified Check" accepted.
Units in stock shipped within 24 hours, F.O.B. Dallas, Texas.
All equipment shipped with manufacturer's warranty.

Residents of Texas, Louisiana, Oklahoma City and Tulsa, Oklahoma must add applicable taxes.

Be sure to write to the address at the top of this ad for more information. Dealer inquiries welcome.

information, and complete descriptions.

In addition, the Blue Book has a Resource Section that lists reference manuals, publications, newsletters, Apple user groups, clubs, time sharing systems, and more.

The suggested retail price of the new 2nd edition Apple Blue Book is $24.95 and is available from most computer shops, bookstores, or direct from:

WIDL Video
5245 W. Diversey
Chicago, IL 60639
(312)622-9606

Scott Adams' Graphic Adventure Series

Scott Adams' Adventure series is being released with high-resolution graphics for the Apple II, adding a new dimension to the game. The graphics are compressed, and drawn using a special palette of over 100 colors. In addition, the new programs support the Votrax Type 'N Talk voice synthesizer, giving a full-color Adventure that talks.

Adventureland, the first adventure in the series, is available now. You will wander through an enchanged world trying to recover 13 lost treasures, encountering wild animals, magical beings, and many other perils and puzzles. It retails for $29.95.

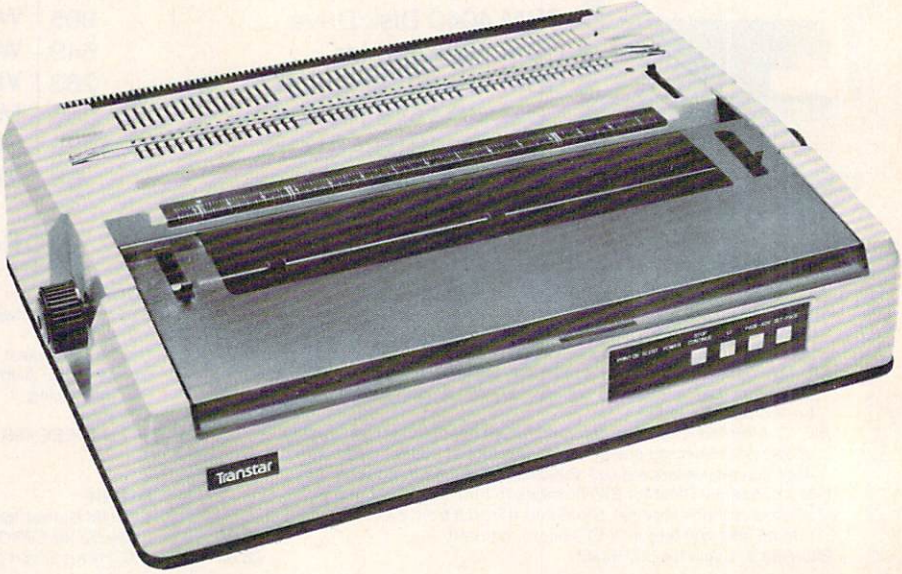
Adventure International
Dept. G, Box 3435
Longwood, FL 32750

Daisy Wheel Printer From Transtar

The new Transtar 140 daisy wheel printer combines 40 cps (38 Shannon) letter quality performance and reliability with a list price of $1695. The new serial printer, built to specification by one of Japan's largest printer manufacturers, is Diablo code compatible for plug-and-go use with Magic Wand and Wordstar. Ribbons and printwheels are also industry standard. Transtar offers a 6-month end-user warranty. Transtar's low profile package is only 6 inches high, and shipping weight is under 50 pounds for UPS delivery. The Transtar 140 printer is available now from:

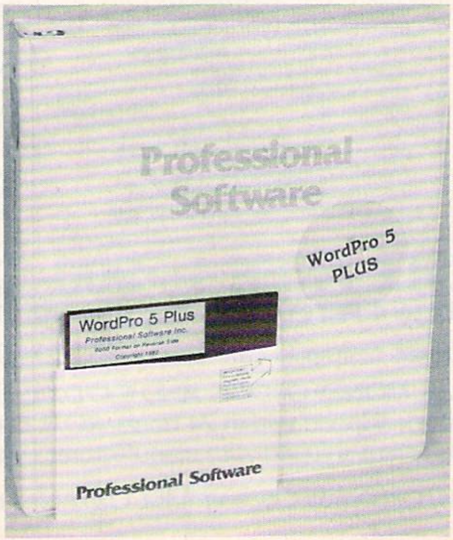
Micro Distributors
11794 Parklawn Drive
Rockville, MD 20852
(800-638-6621)

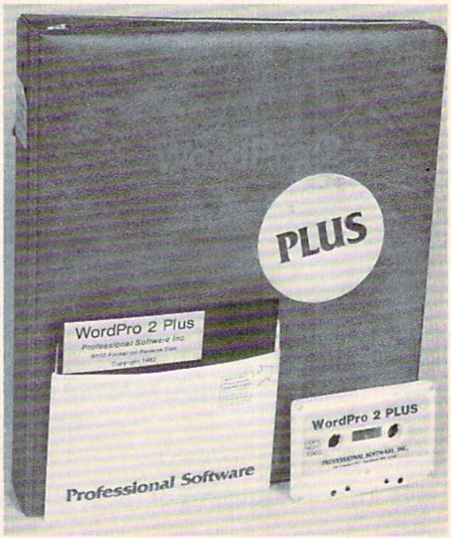
Sigma Distributing
2110-116th Avenue N.E.
Bellevue, WA
(800-426-1412)



Professional Software Releases Two More WordPro Packages

WordPro 5 Plus transforms your Commodore 8096 computer into an easy to use, commercial level word processing system. It creates a system with separate text areas to make word processing easier. Multi-user capability (up to 8 workstations) is available via the addition of a multi-user interface device. It operates with many business oriented programs





including DataPlus, Professional Software's new Information Management program. WordPro 5 Plus is designed for use on Commodore's 8032 computer with Commodore's 64K Memory Expansion Board installed. Any CBM Disk Drive may be used for document storage and any properly interfaced ASCII letter quality or dot matrix printer may be used.

WordPro 5 Plus will be available from Professional Software dealers beginning in January 1982 and will retail for approximately $450.00.

WordPro 2 Plus is compatible with almost any CBM computer available. It requires a minimum of 16K and is sold complete with both cassette and diskette versions and is fully compatible with most CBM computers. WordPro 2 Plus will also operate on all Commodore disk drives. A wide range of popular dot matrix and letter quality printers are supported directly from the program.

WordPro 2 Plus will be available from Professional Software's dealers beginning in the first quarter of 1982, and will sell for $199.95.

Professional Software, Inc.
166 Crescent Road
Needham, MA 02194
(617)444-5224

Universal Data Systems Offers Two New Modems

Universal Data Systems, Inc., a data communications manufacturer has announced a full-featured, Bell-compatible 212A modem which will be priced below present market for comparable devices.

At $695 the UDS 212A offers a saving to data communicators utilizing full-duplex 300 and 1200 bps channels in the same system. The device is FCC certified for direct connection to the dial-up telephone network and is fully compatible with 212As offered by Bell and other modem suppliers.

The company has also announced the addition of the Model 212 LP to their family of modems. Powered from the telephone line, this manual answer unit requires no external AC power, but offers full duplex 1200 only bps asynchronous operation. The 212 LP is FCC certified for direct connection to the dial-up network, and is compatible with the high speed 1200 bps asynchronous channel of the Western Electric 212A. Packaged in a low profile housing, the unit is designed for desk top applications. It is being offered at $495.00 in single quantities.

Detailed technical specifications and quantity pricing may be obtained by contacting:

Universal Data Systems,
5000 Bradford Drive
Huntsville, AL 35805-9990
(205)837-8100

Hayden Announces Computer Literacy Package

I Speak BASIC is a machine specific computer literacy course that introduces students to BASIC programming. The course provides student instruction in the BASIC language for the Apple, TRS-80 and PET and includes a Teacher's Manual, Student Text, and Exam Set for each machine.

Written by Aubrey Jones, I Speak BASIC is designed for teachers regardless of their knowledge of microcomputers and their programming skill.

The core of the course is the Student Text that features learning objectives, definitions and examples of key terms and BASIC concepts in class programming exercises, practices, and assignments. Each version includes chapters explaining the parts and operation of the microcomputer. Chapters cover BASIC programming topics such as mathematical operations, scientific notations, conditional and unconditional branching, input statements, loops, reading data, video display graphics, arrays and subroutines.

The Teacher's Manual provides techniques for presenting the material and emphasizing particular concepts, annotations to aid in lesson planning, suggestions for implementing the course and answers to all practice exams.

The Exam Set contains 12 quizzes on spirit duplicating masters to check student understanding and reinforce learning. The quizzes can be easily reproduced for class use. A classroom set of I Speak BASIC contains one Teacher's Manual, 20 Student Texts and one Exam Set.

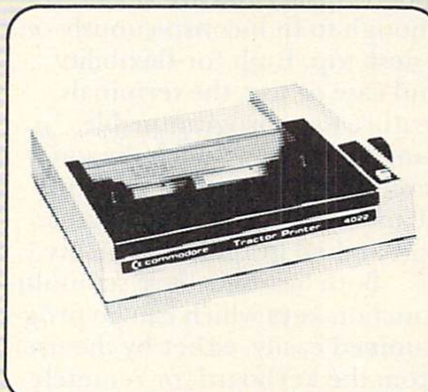
Prices for all three machines are:

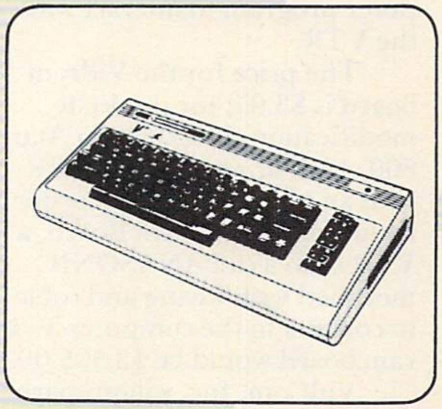
Student Text	$ 7.45
Teacher's Manual	16.20
Exam Set	12.50
Classroom Set	156.25

Hayden Book Company, Inc.
50 Essex Street
Rochelle Park, NJ 07662
Toll Free (800)631-0856
In NJ (201)843-0550

commodore
SHOW SPECTACULAR

8032-32K 80 COL CRT	REG $1495	$1065
64K ADD-ON MEMORY	REG $500	$395
9000 134K SUPER PET	REG $1995	$1795
4032 32K 40 COL CRT	REG $1295	$965
4016 16K 40 COL CRT	REG $995	$765
8050-DUAL DISK 950K	REG $1795	$1295
4040-DUAL DISK 343K	REG $1295	$995
2031-SINGLE DISK 170K	REG $695	$525
C2N-CASSETTE DRIVE	REG $75	$65
4022-80 COL PRINTER	REG $795	$595
8023P-136 COL PRINTER	REG $995	$849
8300P-40CPS LTR QLTY	REG $2250	$1995
8024-MANNESMAN TALLEY	REG $1995	$1595
8024L-LETTER TALLEY	REG $2495	$1995
25CPS-STARWRITER	REG $1895	$1445
CBM-IEEE MODEM	REG $279	$229
VOICE SYNTHESIZER	REG $395	$329





PET TO IEEE CABLE	REG $39.95	$34
IEEE TO IEEE CABLE	REG $49.95	$39
VIC 20	REG $299	$265
VIC 1540 DISK 170K	REG $599	$499
VIC 1515 30CPS PRINTER	REG $395	$349
VIC 1011 RS 232 INTER	REG $49.95	$39
VIC 1112 IEEE INTER	REG $99.95	$79

MATOR SHARK HARD DISK DRIVE

*Winchester Disk
*Commodore DOS 2.5
*24 Megabyte
$6475

Strobe 100 Plotter
*PET Interface
*Software Pack
$975

SCHOOL SPECIALS
3 for 2 Pricing
CALL US!

20% Off on Software

Special pricing on Atari, Apple, Digilog, Epson, Hayes Modem, Printwheels

PHILADELPHIA COMPUTER DISCOUNT©

P.O. Box 170 St. Davids, PA 19087 (215)687-8540

MAIL ORDER PHONE 1-800-345-1289

PREPAID ORDERS SHIPPED FREE
VISA & MASTERCARD ADD 3%
COD - UPS
PA RESIDENTS ADD 6% SALES TAX

H1181C

VidCom, Inc. Announces New Product

VidCom, Inc. has developed a device which plugs into the memory channel of a Personal Computer such as the Atari 800, Apple II, PET and others, making possible the interaction of recorded video taped programs.

The VidCom System consists of a Vidram board, a VTR with connecting cable to the computer Vidram board, a personal computer with an audio program recorder, a color TV, and a computer program to interact with the VTR.

The price for the Vidram board is $3.95; for the VCR modification, $195.00. An Atari 800, 48K, or an Apple II, 48K with an audio cassette recorder, including the Vidram Board, a VTR such as a PANASONIC modified with wiring and cable to connect to the computer Vidram board would be $3,595.00.

VidCom, Inc. will prepare the program instructions for the computer with direct or multiple-choice questions to be asked the student from the video taped materials supplied by customer at the average price of $5.00 per minute running time, or from video taped materials supplied by us at the average price of $10.00 per minute.

VidCom, Inc.
1234 Watchung Ave.
Plainfield, NJ 07060
(201) 754-2377

Personal Information Terminals Introduced By Tymshare

A new line of quiet, compact, personal information terminals, with prices starting at under $500, is being introduced by Tymshare, Inc. Called Scanset, the terminals are designed specifically for one-button information access by non-computer professionals.

The Model 410, with automatic computer log-in, sells for $495. The Model 415, with built-in modem, automatic telephone dialer, and automatic computer log-in, is priced at $649.

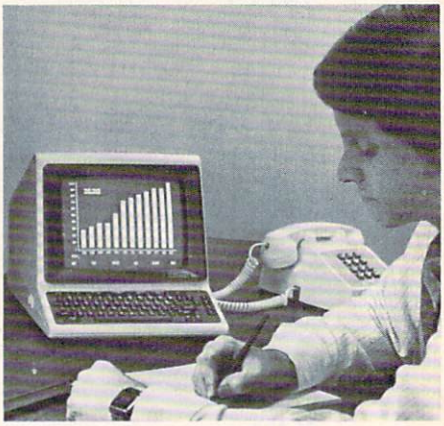
The Scanset units have the same basic abilities to communicate with computers as larger, more expensive terminals. Yet they only take about one square foot of space, and are small enough to fit inconspicuously on a desk top. Built for flexibility and ease of use, the terminals feature user programmable function keys, a 9-inch diagonal screen with 24 lines of text and 40 or 80 character line lengths, and limited graphics capability.

Both Scansets have six multi-function keys which can be programmed easily, either by the user from the keyboard, or remotely from the computer. Up to 12 user-defined tasks can be assigned to the programmable keys, giving the user easy access to host computers or frequently used data bases. They can also be programmed to handle other repetitive jobs.

The autodialer feature of the Model 415 can dial up to 36 phone numbers stored in the terminal's memory, automatically connecting the terminal to computers or data bases. The first four numbers can also be used for automatic computer log-in. In addition, this feature can also be used as an autodialer for a regular telephone. A directory of all numbers, with descriptions, is stored in the terminal and is available on-screen at the press of a key.

The user can view a full 24 lines of text at one time, with a choice of either 40 or 80 character line lengths. Automatic scrolling takes place after the 24th line of text is filled. A 25th line appears at the bottom of the screen, immediately above the function keys, and displays easy-to-read labels for the programmable keys.

The P4 phosphor screen features a flicker-free display. The integral keyboard includes a 69-key standard layout, including four cursor control keys for up, down, left, and right movement of the cursor.



Limited graphics capability, including forms and simple line or bar graphs, is also included. A printer or other device can be connected easily to the Scanset through an intelligent peripheral port. This allows a printer to print out information either directly as it is received through the terminal from a computer, or as it appears on the terminal screen. A buffer management system prevents loss of data.

TYMSHARE
20705 Valley Green Dr.
Cupertino, CA 95014
(408)446-6000

New Product releases are selected from submissions for reasons of timeliness, available space, and general interest to our readers. We regret that we are unable to select all new product submissions for publication. Readers should be aware that we present here some edited version of material submitted by vendors and are unable to vouch for its accuracy at time of publication. ©

CAPUTE!:

Corrections And Amplifications

1. "Renumbering An Appended Routine Only," **COMPUTE!**, January, 1982, #20, pg. 144: the direct BASIC command in column one should read AD=(PEEK(42)+PEEK(43)*256)-2: AH%=AD/256

2. "Starfight3," **COMPUTE!**, March, 1982, #22, pg. 112: [*These changes and hints were sent in by the author.*] Change line 590 to IF T › 0 AND KC › 0 THEN 360. Under item five of the program directions: a. 120 should be 180, b. TIS should read TI$, 450 is 410, 545 is 500, and 1530 is 1300.

This program was written before any memory expansions were available. To use Starfight3 on a VIC with expanded memory, delete lines 30 through 100, substitue an "E" for the "@" in line 920 and substitute a "K" for the "#" in line 980. There will be no little ships now, but the program will run.

If you are receiving an OUT OF MEMORY error message, it is probably due to lines 30-100. Once the program has been RUN, these lines set aside memory that cannot be touched. So, if you make a typo and try to rerun the program, you will run out of memory. The following procedure will let you make corrections:

1. Make the correction. 1. SAVE the program. 3. Turn the VIC off, then back on. 4. LOAD the corrected program. 5. RUN. ©

the SOFTWARE connection For Atari Only

EXTRA SPECIAL SPECIALS

K-razy Shootout Retail $49.95 **Now $37.50**
K-BYTE breaks into the market with this innovative game. ROM cartridge 16K.

Pathfinder Retail $39.95 **Now $25.95**
Blast your way through the maze filled with nu clear waste and radioactive monsters in this never before released game by Gebelli Software.

Raster Blaster Retail $29.95 **Now $22.50**
By Bill Budge. Detailed simulation of pinball, with full-color hi-res graphics, animation, and sound effects.

Apple Panic Retail $29.95 **Now $22.50**
One of Apple's most exciting games is now available for your Atari.

GAMES

		Retail	Now	
Mouskattack	D 32K	$34.95	**$27.95**	NEW
Star Blazer	D	31.95	**26.50**	
Empire of the Overmind	C 40K	30.00	**24.00**	
The Datestones of Ryn	C/D 32K*	19.95	**15.95**	
Rescue at Rigel	C/D 16K*	29.95	**23.95**	
Temple of Apshai Dujonquest	C/D 32K*	39.95	**31.95**	
Star Warrior	C/D 32K*	39.95	**31.95**	
Crush, Crumble & Chomp	C/D 32K*	29.95	**23.95**	
Invasion Orion	C/D 32K*	24.95	**19.95**	
Track Attack	D	29.95	**23.95**	NEW
Computer Acquire	C 32K	20.00	**16.00**	
Lords of Karma	C 40K	20.00	**16.00**	
Reversi	C 8K	19.95	**15.95**	
Deflection	C 8K	14.95	**11.95**	
Sunday Golf	C 16K	14.95	**11.95**	
Adventure Series	C 24K	19.95	**15.95**	
Mountain Shoot	C 16K	14.95	**11.95**	
Galactic Trader	C 32K	19.95	**15.95**	
Poker Solitaire	C 8K	14.95	**11.95**	
Galactic Empire	C 32K	19.95	**15.95**	
Star Trek 3.5	C 32K	19.95	**15.95**	
Gomoku	C 8K	19.95	**15.95**	
Ricochet	C 16K*	19.95	**15.95**	
David's Midnight Magic	D	34.95	**27.95**	NEW
Stone of Sisyphus	D 40K	34.95	**27.95**	
Ali Baba	D 32K	32.95	**26.50**	
The following require joystick controllers				
Angle Worm	C 8K	14.95	**11.95**	
Lunar Lander	C 24K	14.95	**11.95**	
Stocks and Bonds	D 40K*	25.00	**20.00**	
Cypher Bowl	C/D 16K	49.95	**39.95**	
Kayos	C/D 8K	34.95	**27.95**	
Ghost Hunter	C 16K	29.95	**23.95**	
Pool 1.5	D 16K	34.95	**27.95**	
Ghosthunter	D 16K	34.95	**27.95**	
Jaw Breaker	D 16K	29.95	**23.95**	NEW
Andromeda 2	D 24K	34.95	**27.95**	NEW
Match Racer	D 16K	29.95	**23.95**	NEW

COD and **Chargecard** orders may call **(916) 989-3174.** Subject to stock on hand. Prices subject to change.

MISCELLANEOUS

		Retail	Now
Visicalc**	D 32K	$250.00	**$189.00**

A number one best seller by Apple® now available for your Atari.

		Retail	Now
Letter Perfect**	D 16K	149.95	**119.95**
LJK Utility Mail/Merge**	D 16K	29.95	**22.50**
Data Perfect**	D 32K	99.95	**79.95**
Text Wizard**	D 32K	99.95	**79.95**
Datasm-65**	D 48K	149.95	**119.95**
Edit 6502***	R 24K	199.95	**160.00**
The Next Step***	D 32K	39.95	**31.95**
The following require joystick controllers			
Character Generator***	D 24K*	19.95	**15.95**
Lisp 2.0***	D 48K	149.95	**119.95**

(C) Cassette Tape (D) Diskette (R) ROM Cartridge
*Requires Atari Basic ***No printer option
**Printer optional

We carry complete lines from the following companies:

AVALON HILL GAME COMPANY • COMPUTER MAGIC, LTD. • DATASOFT • EDU-WARE • GEBELLI • INNOVATIVE DESIGN • K-BYTE • LJK • ON-LINE SYSTEMS • QUALITY SOFTWARE • SPECTRUM COMPUTERS • STRATEGIC SIMULATIONS • SYNERGISTIC • UNITED SOFTWARE OF AMERICA • VERSA COMPUTING • VOYAGER SOFTWARE • ADVENTURE INTERNATIONAL • ARCADE PLUS • ARTSCI • AUTOMATED SIMULATIONS • ATARI

Catalog free with any order or send $2 postage and handling to

THE SOFTWARE CONNECTION
5133 Vista Del Oro Way
Fair Oaks, CA 95628

MAIL ORDERS: For fast delivery, send certified check, money orders or Visa or MasterCard number and expiration date, for total purchase price plus 1% or $2 minimum for postage and handling. Add $5 for shipments outside the continental U.S. California residents add 6% sales tax.

APPLE USERS CHECK OUR CATALOG

COMPUTE!'s Listing Conventions

Many of the programs which are listed in **COMPUTE!** use special keys (cursor control keys, color keys, etc.) To make it easy to tell *exactly* what should be typed in when copying a program into the computer, we have established the following listing conventions.

For The Atari

All the editing and cursor control characters are spelled out and surrounded by brackets in the program listings: {CLEAR} for "clear screen." Other characters, such as CTRL-T (the "ball" character) will be listed as the "normal" character, but it will be within brackets: {T}. A series of identical control characters will be indicated by a number within the brackets: {3DOWN} means type ESC CURSOR-DOWN three times; {12 R} would mean type CTRL-R twelve times. Remember to press the ESC (escape) key before each cursor control key. If you should see {ESC} itself in a program listing, you would press ESC *twice*.

Two of the control characters, {=} and {-}, should be shifted. Any reverse field text will be enclosed within vertical lines. (In other words, any time you see a vertical line within a program listing in **COMPUTE!**, press the Atari logo key {⅄}.)

Atari Conventions

```
{CLEAR}= SHIFT-< (Clear Screen)
{UP}= CTRL-minus (Cursor Up)
{DOWN}= CTRL-equals (Cursor Down)
{LEFT}= CTRL-plus (Cursor left)
{RIGHT}= CTRL-asterisk (Cursor right)
{BACK S}= BACK S (Back space)
{DELETE}= CTRL-DELETE (Delete character)

{DEL LINE}= SHIFT-DELETE (Delete Line)
{INSERT}= CTRL-INSERT (Insert character)

{INS LINE}= SHIFT-INSERT (Insert line)
{ESC}= ESC (ESCape key pressed twice)
{TAB}= TAB (Tab key)
{CLR TAB}= CTRL-TAB (Clear tab setting)
{SET TAB}= SHIFT-TAB (Set tab stop)
{BELL}= CTRL-2 (Ring buzzer)
```

For PET/CBM/VIC

Generally, any PET/CBM/VIC program listings will contain bracketed words which spell out any special characters: {DOWN} would mean to press the cursor-down key; {3DOWN} would mean to press the cursor-down key three times.

To indicate that a key should be *shifted* (hold down the SHIFT key while pressing the other key), the key would be underlined in our listing. For example, S would mean to type the S key while holding the shift key. This would result in the "heart" graphics symbol appearing on your screen.

Sometimes in a program listing, especially within quoted text when a line runs over into the next line, it is difficult to tell where the first line ends. How many times should you type the SPACE bar? In our convention, when a line breaks in this way, the ~ symbol shows exactly where it broke. For example:

```
100 PRINT "TO START THE GAME  ~
    YOU MAY HIT ANY OF THE KEYS
    ON YOUR KEYBOARD."
```

shows that the program's author intended for you to type two spaces after the word *GAME*.

For The Apple

Programs listed as "Microsoft" are written for the PET/CBM, Apple, OSI, etc. Although the programs are general in nature, you may need to make a few changes for them to run correctly on your Apple. Microsoft BASIC programs written for the PET/CBM sometimes contain special cursor control characters. The following table shows equivalent Apple words. Notice that these Apple commands are *outside* quotations (and even separate from a PRINT statement). PRINT"[RVS]YOU WON" becomes INVERSE: PRINT"YOU WON":NORMAL

[CLEAR[(Clear Screen) HOME

[DOWN] (Cursor down)
Apple II +: Call -922
POKE 37,PEEK(37)+(PEEK(37)<23)

[UP] (Cursor up)
POKE 37,PEEK(37)-(PEEK(37)>0))

[LEFT] (Cursor left) PRINT CHR$(8);

[RIGHT] (Cursor right)
PRINT CHR$(21)

[RVS] (Inverse video on. Turns off automatically after a carriage return. To be safe, turn off inverse video after the print statement with NORMAL unless the PRINT statement ends with a semicolon.)
INVERSE

[OFF] (Inverse video off) NORMAL

Shifted characters can represent either graphics characters or uppercase letters. If within text, just use the non-shifted character, otherwise substitute a space. Some "generalized" programs contain a POKE such as POKE 59468,14. Omit these from the program when typing it in. One final note: you will probably want to insert a question mark or colon within an INPUT prompt. PET/CBM and many other BASICs automatically print a question mark:

INPUT "WHAT IS YOUR NAME";N$
becomes
INPUT "WHAT IS YOUR NAME?";N$

All Commodore Machines

Clear Screen	{CLEAR}	**Cursor Left**	{LEFT}
Home Cursor	{HOME}	**Insert Character**	{INST}
Cursor Up	{UP}	**Delete Character**	{DEL}
Cursor Down	{DOWN}	**Reverse Field On**	{RVS}
Cursor Right	{RIGHT}	**Reverse Field Off**	{OFF}

VIC Conventions

Set Color To Black	{BLK}	**Function Two**	{F2}
Set Color To White	{WHT}	**Function Three**	{F3}
Set Color To Red	{RED}	**Function Four**	{F4}
Set Color To Cyan	{CYN}	**Function Five**	{F5}
Set Color To Purple	{PUR}	**Function Six**	{F6}
Set Color To Green	{GRN}	**Function Seven**	{F7}
Set Color To Blue	{BLU}	**Function Eight**	{F8}
Set Color To Yellow	{YEL}	**Any Non-implemented Function**	{NIM}
Function One	{F1}		

8032/Fat 40 Conventions

Set Window Top	{SET TOP}	**Erase To Beginning**	{ERASE BEG}
Set Window Bottom	{SET BOT}	**Erase To End**	{ERASE END}
Scroll Up	{SCR UP}	**Toggle Tab**	{TGL TAB}
Scroll Down	{SCR DOWN}	**Tab**	{TAB}
Insert Line	{INST LINE}	**Escape Key**	{ESC}
Delete Line	{DEL LINE}		

©

Advertisers Index

COMPUTE! The 6502 Resource Magazine PET•ATARI•APPLE OSI•KIM•SYM•AIM

For Fastest Service,
Call Our **Toll-Free**
US Order Line
800-345-8112
In Pennsylvania call 800-662-2444

My Computer Is:

- ☐ PET ☐ Apple ☐ Atari ☐ OSI
- ☐ KIM ☐ SYM ☐ AIM ☐ Other ______
- ☐ Don't yet have one...

- ☐ $20.00 One Year US Subscription
- ☐ $36.00 Two Year US Subscription
- ☐ $54.00 Three Year US Subscription

(Readers outside of the US, please see our foreign readers subscription card or inquire for rates).

Name ______________________

Address ______________________

City ______ State ______ Zip ______

☐ Payment Enclosed ☐ VISA

☐ MasterCard ☐ American Express

Account No. ______________ Expires ____ / ____

5 6 7 8 9 10 11 12

The Editor's Feedback:

Computer: ☐ PET ☐ ATARI ☐ APPLE ☐ VIC-20 ☐ OSI ☐ OTHER ______

Are you a **COMPUTE!** Subscriber? ☐ Yes ☐ No I would like to see:

☐More	☐Fewer	Specific applications programs.	☐More	☐Fewer	Games.
			☐More	☐Fewer	Reviews of game software.
☐More	☐Fewer	BASIC programs.			
☐More	☐Fewer	Machine language programs.	☐More	☐Fewer	Reviews of business software.
☐More	☐Fewer	Tutorials.	☐More	☐Fewer	Reviews of educational software.
☐More	☐Fewer	Educational articles.			
☐More	☐Fewer	Detailed explanations of programs.	☐More	☐Fewer	Reviews of hardware.

What do you like best about **COMPUTE!**?

What do you like least?

5 6 7 8 9 10 11 12

SELECTED EDP REPORTS

Characteristics, prices, user evaluations, comparisons

☐ All About Personal Computers	$29
☐ User Ratings of Computer Systems	$25
☐ European User Ratings of Computer Systems	$25
☐ Word Processing Systems User Ratings	$19
☐ All About 369 Small Business Computers	$15
☐ All About 262 Minicomputers	$15
☐ All About 190 Microcomputers	$15
☐ All About 148 Microprocessors	$15
☐ User Ratings of Proprietary Software Packages	$25
☐ A Buyer's Guide to Database Management Systems	$15
☐ All About CAD/CAM	$25
☐ All About 260 Alphanumeric Display Terminals	$15
☐ All About Data Communications Facilities	$15
☐ All About Japanese Computers	$29
☐ All About 112 Word Processing Software Packages	$15
☐ All About 142 Teleprinter Terminals	$15
☐ All About 400 Modems	$15
☐ All About Winchester Disk Drives	$15
☐ All About 150 Word Processors	$19
☐ Directory of Over 1000 Suppliers	$29

YES Please send me the EDP Reports I have checked.

Name ______________

Title ______________

Company ______________

Phone ______________

Address ______________

City ______________

State ______ Zip ______

☐ Check enclosed ☐ Bill me, add $3.25 handling. DRO-901A

datapro DATAPRO RESEARCH CORPORATION 1805 Underwood Blvd., Delran, NJ 08075

All About Personal Computers — Only $29

Newly updated for 1982 — the industry's first unbiased, objective report on the 16 most popular personal computers.

Datapro's latest report — **All About Personal Computers** — provides valuable information that will help you compare and evaluate 16 of the most popular personal computers including: Radio Shack's TRS-80 Model III, the IBM Personal Computer, the Xerox 820, the NEC PC 8000, and the Hewlett-Packard 85. Over 70 fact-filled pages in a convenient standardized format let you easily compare data on models, capabilities, configurations and applications, popular options, warranties, prices, retail availability and discounts. In addition, you'll find detailed characteristics of each major peripheral and software product the vendor provides for his system.

All About Personal Computers also contains extensive directories on nearly 200 software vendors, over 300 peripheral equipment vendors, 34 system vendors, and 19 periodicals dedicated to users of personal computers.

The report also defines personal computers; tracks their development; details their use; outlines their current and projected markets, and even takes a look at future trends.

If you are considering buying a personal computer — for home entertainment or work — you owe it to yourself to save time and money. Compare the 16 most popular in **All About Personal Computers**. Available for only $29.

16 MOST POPULAR PERSONAL COMPUTERS

- Apple II & Apple II Plus
- Atari 400 & 800
- Commodore 64
- Commodore PET
- Commodore SUPERPET
- Commodore VIC 20
- Heath All-In-One Computer
- Hewlett-Packard 85
- IBM Personal Computer
- Mattel Intellivision
- NEC PC 8000
- Personal Micro Computer PMC 80
- Personal Micro Computer PMC 81
- Radio Shack TRS-80 Model III
- Radio Shack TRS-80 Color Computer
- Texas Instruments 99/4 Home Computer

datapro

YES Please send Datapro's $29 report **All About Personal Computers**

name ______ title ______

company ______ phone ______

address ______

city ______ state ______ zip ______

☐ Check enclosed. ☐ Bill me, add $3.25 handling. DRO-905A FD 16

FREE with this report, Datapro's latest catalog of feature reports on EDP and office systems, selected from our monthly updated information services.

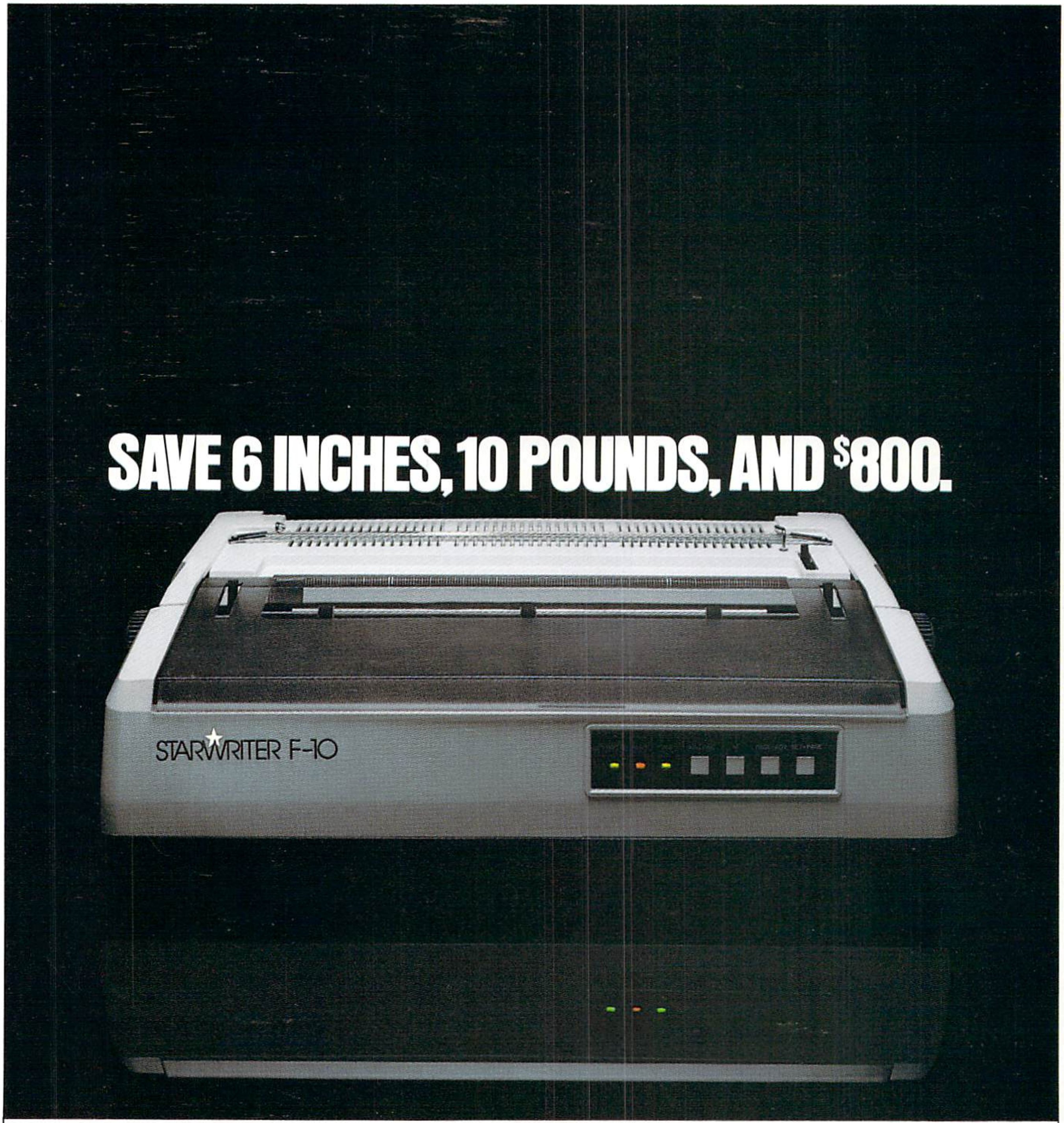
NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

CMP0582

BUSINESS REPLY CARD
FIRST CLASS / PERMIT NO. 178 / DELRAN, NJ

POSTAGE WILL BE PAID BY ADDRESSEE

datapro
DATAPRO RESEARCH CORPORATION
1805 Underwood Boulevard
Delran, NJ 08075



On the new, slicked-up, trimmed-down Starwriter F-10.

It's C. Itoh's latest generation of letter-quality printers.

It cranks out flawless copy at 40 cps; and its full 15" carriage lets it double in brass for both letter processing and business applications. You can plug it into almost any micro on the market (serial or parallel) simply by plugging it in. And then make it keep on trucking with inexpensive, easily available Diablo compatible daisy wheels and ribbons.

In its serial mode, it can print just about anything (including boldface, underlines, subscripts and superscripts), and snap the carriage back to start the next line in less than a second. In its line mode, it prints in both directions, for even faster throughput.

(While making about as much noise as a cat walking on Kleenex.)

It's a nice, portable 30 pounds—about 10 pounds lighter than the Starwriters before it. And it stands exactly as tall (or precisely as small) as a dollar bill.

Speaking of which:

Incredibly, the Starwriter F-10 sells for about the same preposterously low price as its predecessors. Which is to say, about $800 less than a lot of other printers that don't even come close to measuring up. Or even better…

Measuring down.

Distributed Exclusively by Leading Edge Products, Inc., 225 Turnpike Street, Canton, Massachusetts 02021. Call: toll-free 1-800-343-6833; or in Massachusetts call collect (617) 828-8150. Telex 951-624.

LEADING EDGE.™