Atari "Superfont"–
Design Your Own
Character Set

The Resource Magazine For CBM/PET • Apple • Atari • OSI

Learning
With Computers

COMPUTE!

$2.50
January,
1982
Issue 20
Vol. 4, No. 1
63379

The Journal For Progressive Computing™

Now Including Home And Educational COMPUTING!™

INVEST:
A Real Estate Analysis Program In BASIC

Apple Addresses:
A Program To Cross Reference Addresses Between Languages

VIC-20 Color Tips

Cryptogram:
A Code Game For Your Atari

MOUNTAIN MUSIC AIN'T WHAT IT USE TO BE!

MUSICSYSTEM™ FROM MOUNTAIN COMPUTER.

MusicSystem generates the sound of any musical instrument—real or imagined! Solo or sextet. Rock or classical. Laid-back or loud. At home, the concert hall, or the classroom. MusicSystem sets new standards for computer generated music.

- Works in your Apple®.
- Digital Synthesizer with 16 voices.
- Stereo output.
- Polyphonic—multivoice chords and note sequences.
- Additive synthesis of instruments.
- Waveforms are fully programmable for each voice to create instrument definitions and music dynamics.
- 35Khz sample rate.
- Frequency resolution is .5Hz steps.
- Graphical input of sheet music with a graphics printer.
- Complete software operating system.
- Graphical music editor using light pen (provided), game paddles, or keyboard.
- Pre-entered music provided for immediate playing and enjoyment.
- Thorough documentation and tutorial user's manual.

Drop by your Apple® Dealer and ask to hear "Mountain Music" for yourself.

NOW ONLY $395.00

suggested retail

Mountain Computer
INCORPORATED

300 El Pueblo, Scotts Valley, CA 95066
TWX: 910 598-4504 (408)438-6650

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

TM WordPro is a Registered Trademark of Professional Software, Inc. WordPro was written by Steve Punter.

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.

re Systems
nting System II

Series Computer System

Accounts Receivable

Invoices and Monthly Statements as well as Credit and Debit Memos. Accomodates full or partial payments. User sets finance charge rate and period. User also sets invoice aging reports and aging breaks. Invoices may be distributed among nine different General Ledger accounts with automatic updating to the General Ledger. During invoice data entry the invoice is displayed on the screen and typed exactly as if in a typewriter. File capacity is 2000 minus number of customers multiplied by 1.4. Five hundred customers allows room for 2100 invoices.

Accounts Payable

Accounts Payable checks with full voucher detail. Detailed check register. Automatic application of Credit Memos. User sets invoice aging reports and aging breaks. Invoices may be distributed among nine different General Ledger accounts with automatic updating to the General Ledger. File capacity is 2000 minus the number of vendors multiplied by two.

Payroll

Payroll checks with full deductions and pay detail. Pays regular, overtime, holiday and piece work hours. Accommodates Weekly, Bi-Weekly, Semi-Monthly and Monthly employees. Up to eight miscellaneous deductions or payments per employee. Prints Payroll Journal, Payroll Check Register and an Absentee Report as well as 941 information and W2 forms. Maintains Monthly, Quarterly and Yearly totals for up to 350 employees. Automatic General Ledger updating.

See your nearest Commodore dealer for a demonstration.

CMS Software Systems

2204 Camp David   Mesquite, TX. 75149   214 285-3581

TASC™. The Applesoft* Compiler. It turns your Apple into a power tool.

Step up to speed. TASC, the Applesoft Compiler, converts a standard Applesoft BASIC program into super-fast machine code. By increasing program execution speed up to 20 times, Microsoft gives you a power tool for Applesoft BASIC programming.

Highest capacity available. TASC will compile and run larger programs than any other Applesoft Compiler. As a disk-based system, it doesn't require the simultaneous presence of compiler and program in memory. The memory you save allows you to compile significantly bigger programs.

Power without bulk. Code expansion of up to 100% severely restricts other compilers. TASC's special code compression schemes typically limit code expansion to only 25%. You'll really appreciate that with complex programs or programs that utilize Apple's hi-res graphic pages.

More BASIC power. TASC's powerful new commands increase Applesoft BASIC programming capability. Chain with COMMON allows compiled programs to share variables, so a main menu supports several programs in a single runtime environment.

TASC's True Integer Arithmetic and Integer FOR...NEXT capabilities maximize the execution speed of compiled programs.

TASC's near total compatibility with Applesoft speeds compilation of existing programs with little or no modification.

What about mistakes? You perfect your programs interactively with Applesoft. If something does slip by, TASC recovers from errors discovered in compilation and traps all runtime errors. It even permits graceful interruptions during compilation.

See for yourself. Ask for a demonstration of TASC at your Microsoft dealer. Discover the software package that turns your Apple into a power tool.

MICROSOFT
CONSUMER PRODUCTS

A Division of Microsoft Inc.
10700 Northup Way • Bellevue, WA 98004

Table of Contents

January, 1982, Vol. 4, No. 1

The Editor's Notes .... Robert Lock, 10
Ask The Readers .... Robert Lock, Richard Manfield And Readers, 14
Computers And Society .... David D. Thornburg, 18
The Beginner's Page: Loops .... Richard Mansfield, 22

Applications 26
INVEST .... Gregory R. Glau, 26
Developing A Business Algorithm .... Keith Falkner, 48
Anti-Hesitation Programming: A Tutorial On Arrays .... M. R. Smith, 54
How Random Are Sequences Of Random Numbers? .... Brian J. Flynn, 59
Telecommunications: Getting Outside The Computer .... Michael E. Day, 70

Education 76
Friends Of The Turtle .... David D. Thornburg, 76
Learning With Computers .... Glenn M. Kleiman And Mary M. Humphrey, 79

The Apple Gazette 83
Apple Addresses .... Bill Grimm, 83
More Apple Hi-Res Shape Writer .... Chris Dupuy, 86
Lower Case With Unmodified Apple .... Joseph Wrubel, 89
COMPUTE! Overview: Individual Tax Plan 98

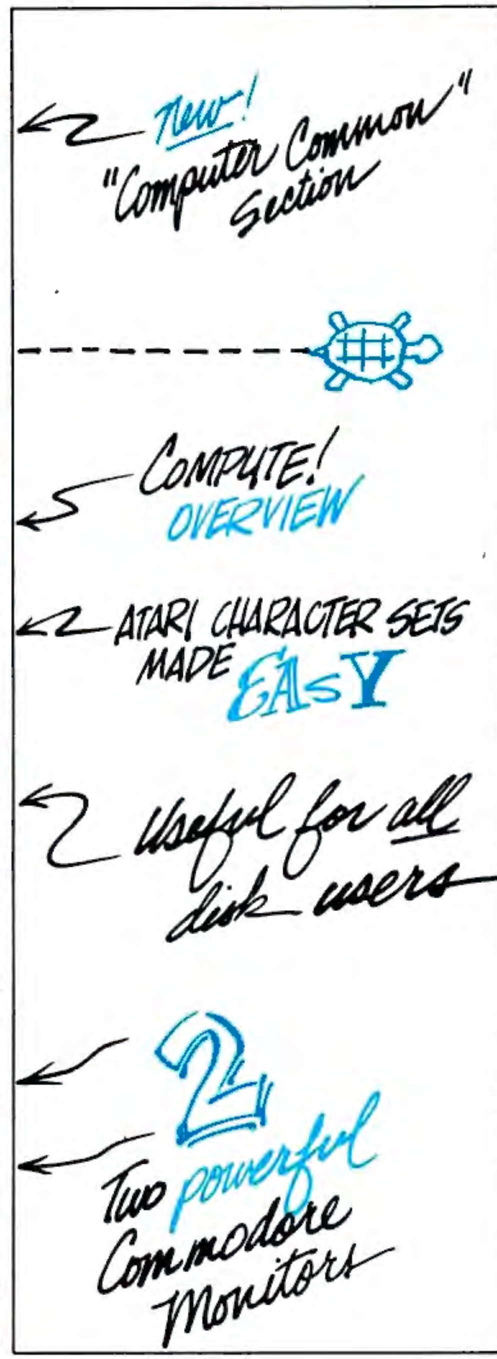
The Atari Gazette 103
Cryptogram .... Ronald And Lynn Marcuse, 103
SuperFont .... Charles Brannon, 110
Word Search .... Bob Jones, 116
Review: Screen Printer Interface (Version 2.0)
From Macrotronics .... David D. Thornburg, 118
INSIGHT: Atari .... Bill Wilkinson, 120

The OSI Gazette 136
Part I: A Small Operating System: OS65D The Disk Routines .... T. R. Berger, 136

The PET Gazette 143
A Yuletide Tale 143
Renumbering An Appended Routine Only .... Elizabeth Deal, 144
BRANCH NEVER And QUIF Assembling On SuperPET .... Richard Mansfield, 146
PET Repairs For The Amateur .... Louis F. Sander, 150
Realtime Clock On Your PET Screen .... Mark L. Robinson, 152
Tape Load Test And Head Alignment .... Louis F. Sander, 156
MICROMON: An Enhanced Machine Language
Monitor .... R. Arthur Cochrane, 160
Self-Modifying Programs In BASIC .... David Williams, 174

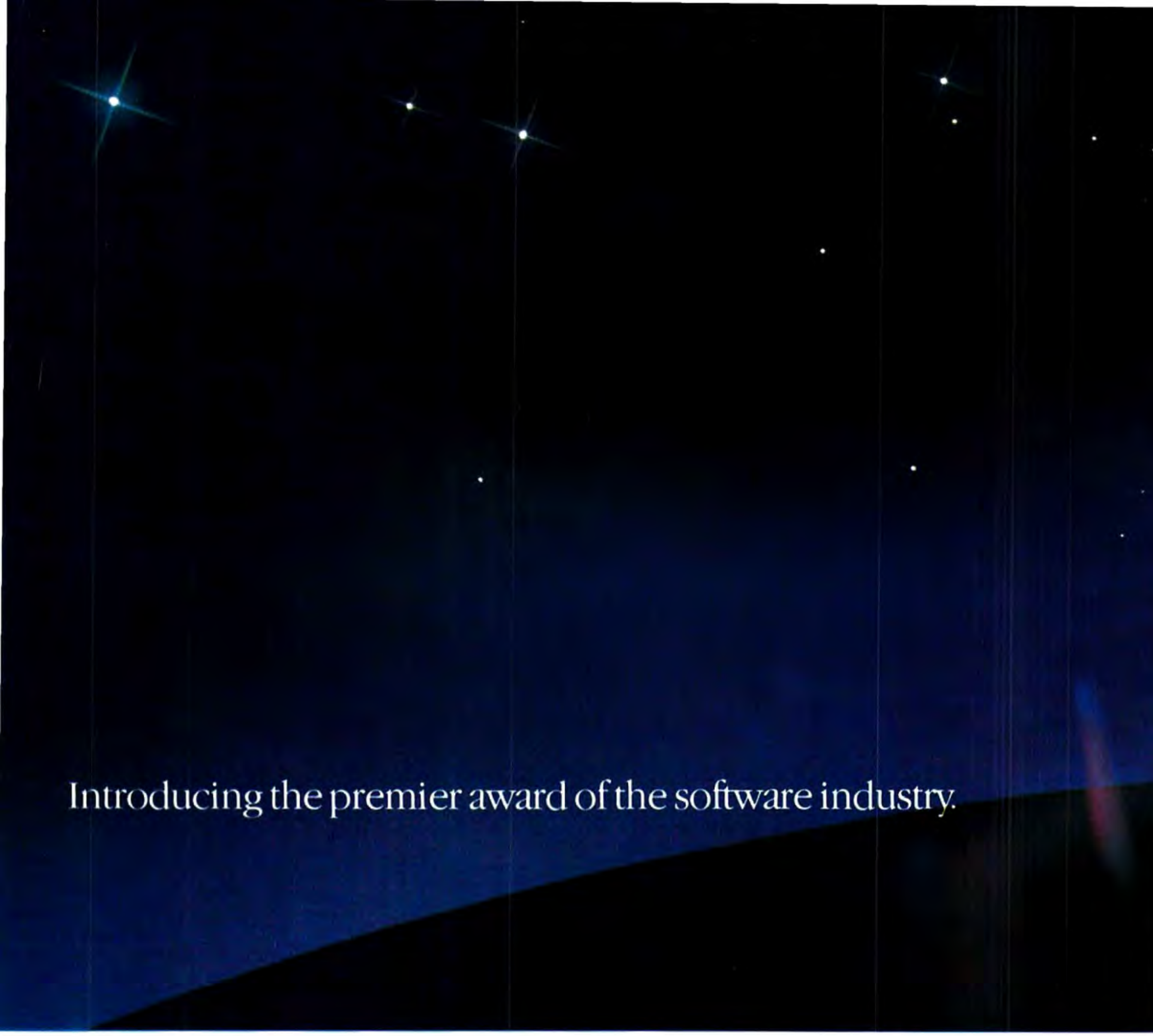
VIC-20 Update 176
TINYMON1: A Simple Monitor For The VIC .... Jim Butterfield, 176
VIC Color Tips .... Charles Brannon, 180
VIC Memory Map Above Page Zero .... Jim Butterfield, 181
ZAP!! .... Dub Scroggin, 1813

CAPUTE! Corrections And Amplifications 1831
New Products 189
Advertisers Index 208



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.



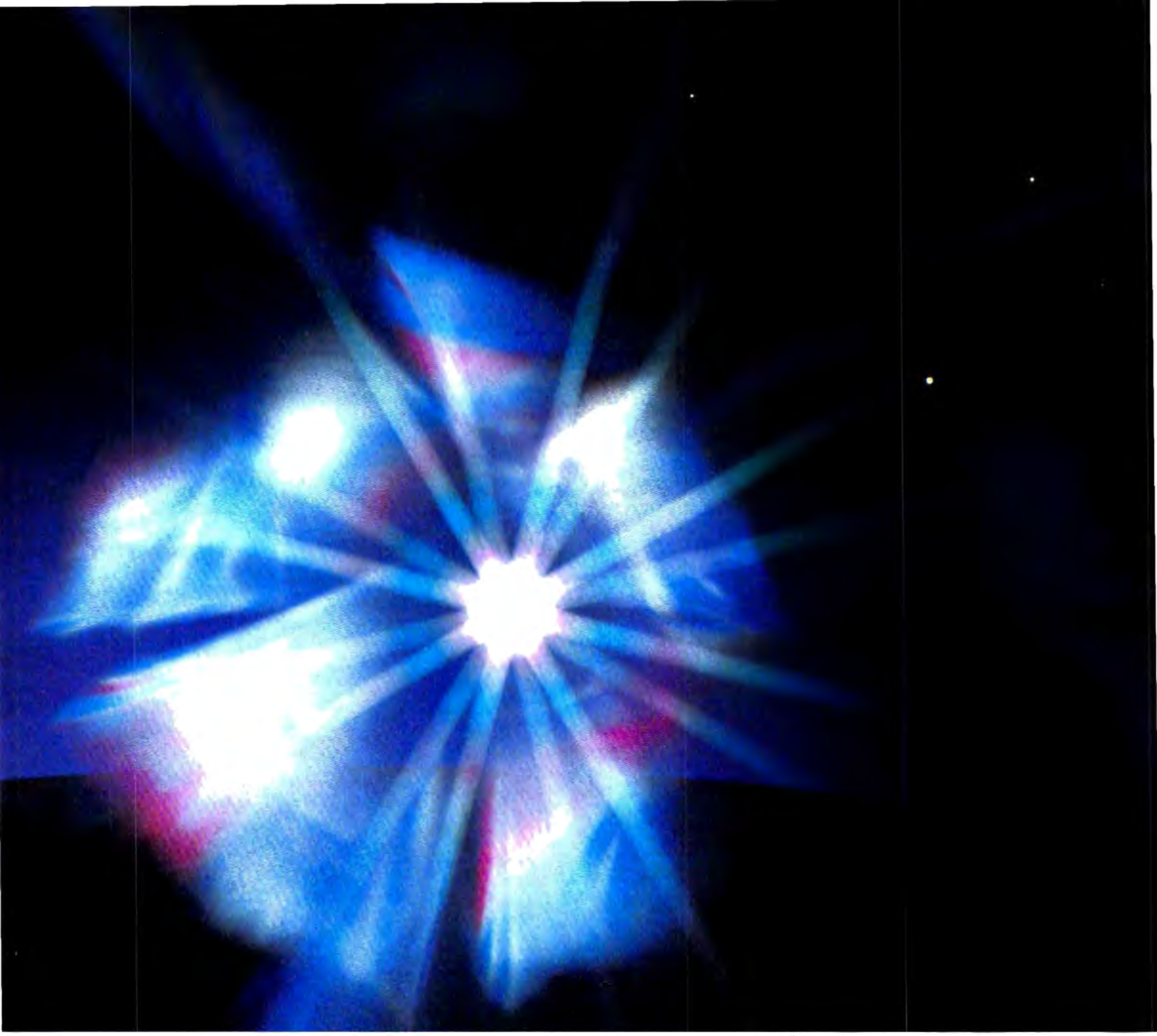
THE ATARI® STAR

Win $25,000 in cash, plus prizes and an ATARI STAR by entering your software in the ATARI® ASAP competition.

ATARI is looking for some of the greatest software from the brightest stars in the software field. And naturally, we're willing to reward you for it.

Every three months, ATARI will be presenting ATARI STARS to the writers of software programs judged first, second and third place in the following categories: consumer (including entertainment, personal interest and development); education; home business (personal finance and record keeping); and system software. The decision of the ATARI judges is final.

Quarterly prizes will be from $750 to $3,000 worth* of selected



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, Director, Advertising Sales/ Promotion
Terry Cash, Typesetting/Production Assistant
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, MD 65726
Eric Rehnke, 1067 Jadestone Lane, Corona, CA 91720
David Thronburg, 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 (12 Issue Year):
COMPUTE! Circulation Dept.
515 Abbott Drive
Broomall, PA 19008 USA

U.S. $20.00
Canada $25.00 (U.S. funds)
Europe: Surface Subscription, $25.00 (U.S. funds)

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

Canadian Retail Dealers should contact:
Micron Distributing
409 Queen Street West
Toronto, Ontario M5V 2A5
416-361-0690

Advertising Sales

If you're in **Oklahoma, Texas or the Western States, we're now represented by Phoebe Thompson and Associates.** Give them a call for space reservations, contract/insertion information or questions.You can reach them through the following office:

Phoebe Thompson and Associates
101 Church Street
Suite 13
Los Gatos, CA 95030
408-354-5553
PHOEBE THOMPSON

If you're in the East, we're now represented by **The Gittleman Company**. You can reach them through the following offices:

New England, New York State:
The Gittleman Company
Statler Office Building
Suite 582
20 Providence Street
Boston, MA 02110
617-451-0822
JOAN DONAHUE

The Gittleman Company
Summit Office Centre
7622 Summit Avenue
Fort Washington, PA 19034
215-646-5700
DOUG JOHNSON

New York City Metro Area, Mid-Atlantic and Southeastern States:
Local Numbers:
New York 212-567-6717
Atlanta 404-523-1252

If you're in the **Midwest, we're now represented by GB & Associates**. You can reach them through the office of:

GB & Associates
P.O. Box 335
Libertyville, IL 60048
312-362-1821
GORDON BENSON

COMPUTE! Home Office Contacts

Advertising Sales Manager, Harry Blair
Advertising/Production Coordinator, Alice S. Wolfe

Address all advertising materials to:

625 Fulton Street
Greensboro, NC 27403 USA

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

Telephone: 919-275-9809

Small System Services, Inc. publishes:

COMPUTE! The Journal For Progressive Computing

COMPUTE! Books

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

Corporate offices are located at 625 Fulton Street, Greensboro, NC 27403 USA

COMPUTE! Subscription Rates

US	$20 (one yr.)
	$36 (two yrs.)
	$54 (three yrs.)
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

Coming In The February Issue of COMPUTE!:

Insurance Inventory: A Database Program For Household Insurance Inventories

Transposition: Mathematically Transpose Musical Compositions

VIC-20 Secrets

And Much, Much More...

The Editor's notes...

Robert C Lock
Publisher/Editor

Injunctions, Injunctions, And More Injunctions Atari, Inc., Goes To War

And I'm not talking about the popular Eastern Front WW II simulation, either. If you picked up a computer magazine recently, you couldn't help noticing the full-page software piracy ads Atari Personal Computer Systems has been running everywhere. Atari has been moving quickly and quietly against major and minor software vendors whose products step on the toes of Atari arcade games.

One vendor on the West coast, recently losing a round of injunctions and counter-injunctions, serves as a case in point. The popular game, developed by the vendor from "scratch" for the Atari computer, mirrors in part a very successful Atari arcade game. Does Atari, Inc. have the software out for the personal computers? Well no, but that's not the point. In spite of the fact that the computer version of the game is significantly expanded, quite original in coding (there was none before this version), and rumored to be a real pleasure, the current state of software law appears to side with Atari ... at least it did at the end of the current round of claims. The visual image and theme of the game are decidedly Atari's, thus we end up with protection based to some extent on concept. Pure and simple.

Let The Vendor Beware

The way we hear it, Atari informed these vendors that they would have to stop the sale or distribution of this software. The vendors pointed out that they had developed the game and its program code originally, etc... To no avail, it turned out; Atari obtained an injunction to halt distribution. The vendors asked Atari to license the game to *them*, thus generating royalties for Atari, and permitting the vendors to pursue sale and distribution. Atari said no, but did apparently ask the vendors if *they* would consider developing a version of the game for Atari! Predictably enough, the vendors declined, and went to court, obtaining an injunction allowing sale and distribution. Whereupon, Atari went back and emerged victorious, for the moment, quashing that injunction (I believe that was Round 2?), and obtaining the one that's currently in force (Round 3?).

Then Linda Turned To John And ...

First of all, I fully support Atari's right to protect their proprietary software. That principle has to be firmly embedded in the computer industry to allow it to grow and nurture even more exciting future growth. But there does seem to be a grey area here which needs to be more fully explored. I suspect, with this recent flurry of legal activity, that the screen is becoming cloudy, as it were. I assume no one is arguing whether this game is original, unique program code. I assume no one argues that it took months to develop, perfect, refine, and yes, enhance.

So, we're back to concept, visual image, style of presentation... Would it have made any difference if the imagery had been uniquely different? Can it be? I mean there are only so many ways to program an arcade style game on a 10 or 12 inch screen. One begins with chasers and chasees, and proceeds from there. The general form is that chasers have sophisticated weaponry, and grow more sophisticated as the game progresses (we call these "skill levels"). Chasees have various means of fighting back. And that, with allowance for creative variation, is the backbone of computer-based gaming, arcade style.

At this point in the analysis, we're several stages removed from actual program code. Its uniqueness has become moot for the moment. In this case there is no computer software to check against the twice-released game. Atari hasn't developed it yet. If the existing game is a direct "copy" of the Atari arcade game, I would guess they'll end up the winners, and software vendors will be a bit wiser for it.

Let The Good Times Scroll

Here's the danger of it ... depending on the tightness of court interpretations of this portion of the fight, we're leery of ending up in a situation so broadly defined it defeats "competitive" gaming. Given that we're arguing concepts and imagery, rather than written program code and precisely comparable listings, a broad interpretation of the rights to "player-missile graphics" would cripple the software industry, leaving access to a few. Those few, at this point, would be the companies currently holding the reins on the arcade market. Two biggies, by the way, are Atari and Commodore. Com-

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

modore, you see, has the right to produce all Bally arcade games for the new VIC-20.

We're confident the courts would not allow a TV producer to claim proprietary rights to "soap operas," police shows, or westerns. Let's hope the situation that's just now shaping up in the computer gaming industry will avoid the same undue constraints, while protecting the rights of all.

An Apology, And A New Year

Our 1982 production schedule is finalized and, as you should discover with this issue, we're back on schedule. You should be receiving your magazine around the first of the cover date month (or perhaps a bit earlier). That's the way we want it. Never quite wishing to bow to the needs of newsstand distribution overseas, we don't see much point in sending you the February issue in December. We'll stay on this schedule now, our production department is rolling along, and **COMPUTE!** grows on. And, oh yes, the next time we schedule publication of a book, we'll know whereof we speak when we calculate our production time! Thanks for your patience.

Best wishes for a happy and productive New Year from all of us at **COMPUTE!** ©

With the Hayes direct-connect Micromodem II,™ your Apple II can communicate by phone with the outside world. You can access information networks like The Source for a variety of business and personal applications, exchange programs with friends anywhere in North America, and even use your Apple II when you're away from your home or office.

Dependable. The Micromodem II is so dependable it comes with a *two-year* limited warranty. That's another reason why it's the largest selling direct-connect modem for Apple II computers.

Programmable. Automatic dialing and answering? Of course! We include programs on disk that dial phone numbers for you, send messages while you're away, and much more!

Complete. You get everything you need to communicate with other Bell 103 compatible modems at 110 or 300 baud. The serial interface is built-in, and we even include our FCC-approved Microcoupler™ that plugs directly into any modular telephone jack in the U.S. — you don't even need the phone!

S-100, too. The Hayes Micromodem 100 gives S-100 microcomputers all the advantages of our Micromodem II!

Put the outside world *inside* your computer with a data communications system from Hayes. Available at computer stores nationwide — call or write for the location nearest you. And don't settle for anything less than Hayes.

Hayes™

The Hayes Micromodem II opens up your Apple II† to the outside world.

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

Micromodem II, Micromodem 100 and Microcoupler are trademarks of Hayes Microcomputer Products Inc.
†TM Apple Computer Inc. Micromodem II can also be used with the Bell & Howell computer.

Ask The Readers

Robert Lock, Richard Mansfield
And Readers

If you have any questions (or answers to the questions printed below) please write to: Ask The Readers, **COMPUTE!** Magazine, P.O. Box 5406, Greensboro, NC 27403.

Answers

"There is a small design flaw in the way that Commodore BASIC chains between programs. The flaw is small, but will result in certain strings disappearing (and being replaced by a string of BASIC tokens).

Strings defined in a program as a constant (i.e. A$ = "Hello") are not allocated space in RAM. The pointer to that string points back into the program to the line containing the literal. This is normally fine, but when the program chains into another program, the string pointers now point to some piece of your new program. Now what?

The solution is to not use any strings assigned as a constant. The assignment above should be replaced by A$ = "Hello" + " ". The null concatenation insures that BASIC will copy the string to RAM somewhere, and it will still exist after chaining to the next routine. Please note that DATA statements count as constants in this context. Use READ A$:A$ = A$ + " " to copy the string to upper RAM.

This is all wasteful if you do not chain to other routines, or if you don't use the old variables in the new routine, but be careful. Certain tokens (like RETURN) will redefine the character set or window size, a real problem if you don't realize why it happened."

Michael Schaffer

"I would like to respond to the question raised in "Ask The Readers," **COMPUTE!** *#16, regarding the future of the 6502. I keep hoping some manufacturer will do a 16 bit version of the 6502 ... as good as the 6809 seems to be, it is still a traumatic switch for those who have cut their teeth on the 6502, to say nothing of the software and hardware investment in the 6502 machines. To me, this shift to the 6809 seems more like a lateral — rather than a forward-looking move for the future.*

If I must switch, my choice would be to opt for a 16 bit machine such as the 68000. Then my present 6502 unit would become a smart terminal to access the power of the 16 bit unit. In the interim, maybe some clever and enterprising reader will create a dual (parallel operation?) 6502 machine to emulate 16 bit operation.

I believe that if there must be a change (and there will be, as progress demands) then **COMPUTE!** *magazine will demonstrate immeasurable foresight by choosing to lead the way into the 16 bit personal computer world.*

Dr. Charles DeSantis

"I have been following the discussions about 'software piracy' in various publications and I am quite impressed with the arguments about 'protection' vs. 'backupability.' I'm in favor of the backupists in general. As the King of Siam is reputed to have said, 'Is a puzzlement!'

I kind of hate to do this, but all of the discussion so far has left out one other *part of the problem. I have a PET 2001 with Upgrade ROM. There is a lot of good software out there for the Apple, Atari, and others that I can't just LOAD and RUN. Say I have a friend with an Apple. He bought a $200 program that I covet. If I convert it to PET and use it, I'm a pirate? I certainly won't buy it unconverted and, after all that work, I'm in no mood to pay the producer... After all,* he *ignored me! I don't know the answer to this searing, burning question either, but I thought I could stir the pot with it.*

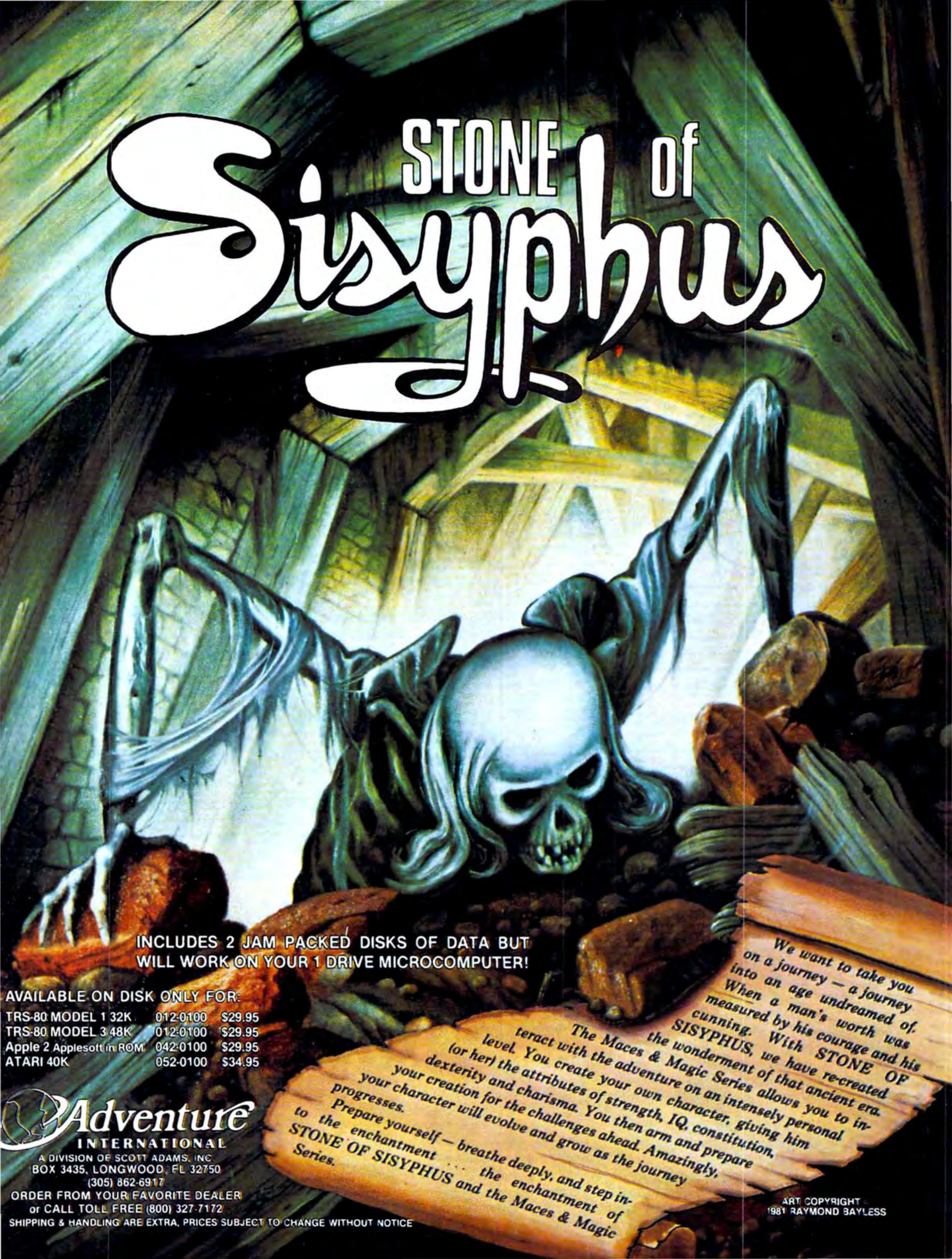
I'm looking forward to the articles about the 2.1 and 2.5 DOS. Let me throw in one thing that I've learned the hard way. COPYD0 TO D1 doesn't work in DOS 2.5 unless both disks have the same *ID! During the copy sequence, if the next source program is cataloged on a different Directory block, you get DISK ID ERROR. In partial answer to M.J. Band, the U3 through U9 commands access RAM locations where you can put disk control programs of your own. If you knew the disk environment. The possibilities are fascinating! For instance, a* sort *routine could be put in there which would presort the output of your file while the PET did other work. Or maybe one that would recognize only CHR$(13) as a delimiter so you wouldn't have to use all those GETs to recover ordinary text with commas in it. (Make that delimiter an option, I have a program that doesn't put RETURNS at the end of a line, just CHR$(29)s at the beginning. It's in ROM, I can't fix it.)"*

R. Vanderbilt Foster

Questions

"I had read that you may double your disk's holding capacity by cutting out the proper notches on the backside of your disk's envelope. So, I grabbed my Wabash single density disks, a scissor and went snipping away. Several problems arose after trying to use the backsides on my Atari 810 disk drive. First I received many 144 errors (device done) while formatting the disk under DOS 2.0S, but successful (I thought) under DOS 1. My second problem occurred when I failed to be able to copy any files to disk. I had the speed and head pressure adjusted but still no luck."

Thomas M. Krischan


STONE of Sisyphus
INCLUDES 2 JAM PACKED DISKS OF DATA BUT WILL WORK ON YOUR 1 DRIVE MICROCOMPUTER!
AVAILABLE ON DISK ONLY FOR:
TRS-80 MODEL 1 32K 012-0100 $29.95
TRS-80 MODEL 3 48K 012-0100 $29.95
Apple 2 Applesoft in ROM 042-0100 $29.95
ATARI 40K 052-0100 $34.95
Adventure INTERNATIONAL
A DIVISION OF SCOTT ADAMS, INC.
BOX 3435, LONGWOOD, FL 32750
(305) 862-6917
ORDER FROM YOUR FAVORITE DEALER
or CALL TOLL FREE (800) 327-7172
SHIPPING & HANDLING ARE EXTRA, PRICES SUBJECT TO CHANGE WITHOUT NOTICE
We want to take you on a journey — a journey into an age undreamed of. When a man's worth was measured by his courage and his cunning. With STONE OF SISYPHUS, we have re-created the wonderment of that ancient era. The Maces & Magic Series allows you to interact with the adventure on an intensely personal level. You create your own character, giving him (or her) the attributes of strength, IQ, constitution, dexterity and charisma. You then arm and prepare your creation for the challenges ahead. Amazingly, your character will evolve and grow as the journey progresses.
Prepare yourself — breathe deeply, and step into the enchantment . . . the enchantment of STONE OF SISYPHUS and the Maces & Magic Series.
ART COPYRIGHT 1981 RAYMOND BAYLESS

JINSAM

Data Manager selected by NASA, Kennedy Space Center

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

OFFICIALLY APPROVED BY COMMODORE

"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 is a trademark of JINI MICRO-SYSTEMS, Inc.
WordPro is a trademark of Professional Software, Inc.
VisiCalc is a trademark of Professional Software, Inc.
CBM is a trademark of Commodore Business Machines, Inc.

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

Although it is possible, in theory, to record on both sides of ordinary disks, it is not a good idea. Some disks are designed to be "double-sided" and provisions are made to thicken and strengthen the disk so that the recordings on each side will not interfere with one another. "Print-through," where the information being magnetized on one side also appears on the second side, can obviously play havoc with whatever was already on the second side.

In addition, disk surfaces are so delicate that a single cigarette ash, floating onto the disk, can render it useless. This would suggest that cutting holes in the outer envelope might deform the surface, or worse. One final drawback: attached to the inner surface of the envelope is a soft, textured fabric designed to trap any stray particles and keep them off the disk surface. Using the opposite disk side causes it to spin in the opposite direction, dislodging and redepositing any foreign matter back onto the surface.

"This is first and foremost a great big thank you letter. Thank you for existing so that numbskulls like me have a place to turn to in their ignorance.

Last winter I wrote another computer magazine a letter deploring the dearth of PET material in their columns. Almost as soon as the issue with my letter in it hit the post-boxes, I was inundated by letters from helpful guys (especially Jim Butterfield) who turned me on to you all. Thank goodness! Wish I had the time to thank each of them individually.

One big cloud still hangs over my head though. Why does everybody have to be so much smarter than I? Acronyms fly all over the place in everything I read. And a body would think that you all were more at home inside your PET than you are in your own living rooms. When I open up my 2001, I'm lucky if I can tell back from front! Is there any way short of becoming an electronics wizard for us above referenced numbskulls to get to know what you all are talking so glibly about? Tell me please, how do I get to address $A000 from here?

A kind word which you may wish to pass on to your advertisers is this: If you want us numbskulls to buy your products, stop writing your ads in shorthand!"

J. Paul Morris

We strive to provide articles and programs which are clearly written and easily utilized by all readers. Nevertheless, computer terms are confusing and new ones are added every year. One solution is to buy a dictionary of microcomputer terms — most bookstores carry several. From time to time, we reprint glossaries and we include a number of articles each month which are, essentially, tutorial. Also, "The Beginner's Page" explores a different subject each month (this month it's *loops*) with extensive definitions of terms and example programs. Finally, "Ask The Readers" itself has become a popular forum for the exchange of information.

"How can you get access to all 48K of memory in a 48K Atari? Is it possible to write a self-booting program (somehow) which doesn't need to use a cartridge at all? Or is there a way to remove the cartridge while in DOS, load the object code in binary form, and execute it directly?"

Rick Grosckiewicz

When you remove the cartridge from a 48K Atari, the top 8K is accessible as RAM, but without a programming language, how do you use it? You can use 6502 "machine language" in which programs can be written to "boot" (automatically load) in when the computer is turned on. They can be in the form of a cassette boot (see "SHOOT," **COMPUTE!** #16) or with DOS as an AUTORUN.SYS file. Commercial software such as Microsoft BASIC, Visicalc, or BASIC A + all can use this extra RAM. There is more information on this in the DOS Manuals and in **COMPUTE!**'s Atari Gazette.

"I would like to know where I can get a list of furniture manufacturers who make desks to house my Atari 800 and peripherals."

Robert Fersch

Computers And Society

David D. Thornburg
Innovision
Los Altos, CA

Further Ramblings On The Mind ...

When I first started reading Douglas Hofstadter's book, *Goedel, Escher and Bach*, I thought I would be lucky to finish reading it by 1990. While the book is fascinating and I pick it up from time to time, I have had to set it aside for more pressing matters. It was thus with some trepidation that I bought a copy of *The Mind's I*, a recently published book (Basic Books) by Douglas Hofstadter and Daniel Dennett.

Hofstadter's field is artificial intelligence, and Dennett's is philosophy. Dennett recently published a collection of his essays on epistemology (*Brainstorms, Philosophical Essays on Mind and Psychology*, MIT Press). It appeared that these two powerhouse thinkers decided to collaborate on a book which covered an area of immense interest to each of them — the nature of the mind.

At first glance, *Mind's I* appears to be a collection of articles from various sources, each of which deals with one perspective on the concept of the mind. Hofstadter's and Dennett's notes after each article provide a cohesive framework which helps the book hang together. For example, Alan Turing's landmark article "Computing Machinery and Intelligence," in which the famous Turing test is described, is followed by "The Turing Test: A Coffeehouse Conversation," an article Hofstadter first published in *Scientific American*.

The Turing Test

Turing's test, in its simplest form, has an experimenter sitting at two terminals — one of which is connected to a computer and the other of which is connected to a similar terminal manned by another human being. The experimenter is free to direct questions through each terminal and is supposed to deduce, from the responses, which terminal is connected to the computer. Turing suggested that if the experimenter is not able to do this reliably, then we can say that the computer is, in fact, thinking.

... he concentrates in the idea that the mind is an intentional system ...

In Hofstadter's article, the issue is raised as to whether a good simulation of thinking is the same thing as thinking itself. This theme recurrs several times in the book and is not easily answered.

The collection of articles in this book cover the concept of the mind from a multitude of approaches. Hofstadter and Dennett provide a balanced picture. The strict reductionist view of life and mind resulting from a seething molecular soup in which small units, accidentally formed, are subjected to fierce competition for resources with which to replicate, is presented by an excerpt from Richard Dawkin's book, *The Selfish Gene*. A more mysterious quality for the mind is suggested by Harold Morowitz's article "Rediscovering the Mind" which first appeared in *Psychology Today*. One cannot help but be struck by the tremendous diversity of opinion expressed in this book. There is something to please and infuriate any reader, regardless of his or her philosophical leanings.

The function of this book is less to present a particular view than to raise the level of conversation about the topic. After all, it is senseless to ask if machines can think when we have yet to agree on just what thinking or consciousness is.

Dennett's book, *Brainstorms*, has a different goal. The collection of essays in this book are designed to elucidate Dennett's own philosophical view of the mind — a view which is aided by the experimental evidence being accumulated in many fields. His theory differs from other models in important ways. The physical model of the mind, for example, implies that when two creatures have the same thought in common (e.g., the belief that snow is white), then they have something physical in common too (their brains are in the same physical state). This is extremely unlikely, as Dennett points out.

Intentional Systems

His theory does not deny the possibility of a correspondence between mental and physical states. Instead he concentrates on the idea that the mind is an *intentional* system — one whose behaviour can, at least sometimes, be explained and predicted by treating it as though it had beliefs and desires.

If one looks only at external views of the system, it is logical to ask if this model applies to machines as well as to human minds. Consider a computer programmed to play chess. One can examine this system from three perspectives. By taking the *design stance*, one can predict the game's behavior by knowing the details of the computer and its program. As long as the system behaves as programmed, predictions made from this analysis will be true. This stance is most useful when dealing with simple systems (strike a match and it will light). The *physical stance* bases predictions on the actual physical state of the system, and then uses the laws of nature to predict what will happen next. This approach is most difficult to apply to a machine as complex as a digital computer.

Chess playing computers are practically inaccessible to prediction from either the design or physical stance. Even their own designers would have a hard time describing these machine's behavior from the design stance. The best strategy for someone playing against such a machine is to treat it as if it followed the rules and goals of chess. One assumes that the computer will both function as designed and that it will "choose" the most optimal move.

This attribution of rationality to the system is the cornerstone of the *intentional stance*. One predicts behavior in such systems by assuming them to possess certain information and to be directed by certain goals. This ascription of *beliefs* and *desires* to machines appears to suggest that machines are capable of "thought."

The aspect of Dennett's argument which I find most appealing is its reluctance to tackle thought on a microscopic scale. As long as he is able to deduce the characteristics of a system from its behavior, he is unlikely to get much criticism from any of us who feel that it is nonsense to suggest that machines are capable of what we, as humans, would call consciousness or thought.

Both *The Mind's I* and *Brainstorms* are fascinating books. You should approach them cautiously — they are not light reading. You might decide that the real issue is not whether machines are capable of thought, but just what constitutes thought in the first place. ©

NEW PRINTERS. NEW PERIPHERALS. SAME OLD RELIABLE QUALITY AND VALUE.

1982 will find more OEM's, businesses, dealers and personal computer users turning to MICROTEK than ever before.

TekWriter-1

80 Column Dot Matrix Printer
(Formerly BYTEWRITER-1)

The Tekwriter-1 printer is, dollar for dollar, the finest value in the industry. And we've proved it by comparing the Tekwriter-1 to the Epson MX-80. Our print speed is 14 lines per minute faster, our life expectancy is longer, the character sets are the same, and the interface, warranty and printhead replacement cost are all identical.* But the biggest difference is the price. The Tekwriter-1 is about $300 less.

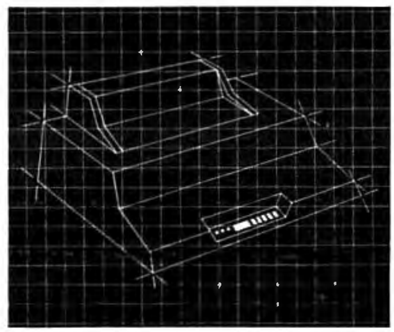
Our extensive testing has proved that the Tekwriter-1 interfaces problem-free to the TRS-80, the Apple II and the Atari 400 and 800.

The Tekwriter-1 is tough to beat for performance and quality.

*Data Source: Epson MX-800 Operation Manual

$349

TekWriter-2



NEW! 80/132 Column Dot Mattix Printer

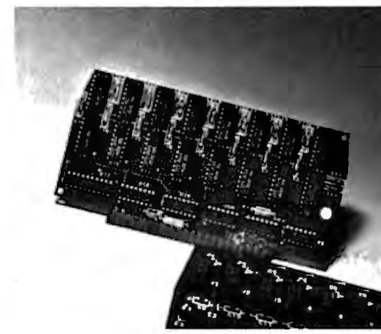
The Tekwriter-2 is perfectly suited to personal, business or OEM applications. Tekwriter-2 is designed to accept single sheet, roll or pin feed paper. It has a 9-wire dot matrix impact print head which produces crisp characters and has underlining capability. The printer is manufactured to run extremely quietly even while operating at peak output levels.

Tekwriter-2 is especially well suited to handle an abundance of text entry because of its data buffer expansion capability to 25K. This ability makes it an efficient graphics generator.

Parallel interface (Centronics type). Interfaces all models of TRS-80, Apple, and Atari 400/800, and most computers with Centronics printer interface.

$695

Peripherals



16K Memory Board, AMB-16

16K 4116 RAM (200NS)
• Assembled and tested • No modifications — hardware or software • Compatible with Atari 800

32K Memory Board, AMB-32

32K 4116 RAM (200NS) • Assembled and tested • No modifications — hardware or software • Compatible with Atari 400/800

Atari (RS-232) Serial Printer Cable

Pre-tested • 3' length • DB15 to DB25 connectors

Atari Parallel Printer Cable

Pre-tested • 3' length • Centronics compatible • DB15 to Amphenol 57-30360

16K Apple Memory Card

Expands Apple II to 64K RAM memory. Works with MICROSOFT Z-80 Softcard, Apple PASCAL and Visicalc software.

Quantity and OEM discounts available.

Continuing our quest for excellence.

TRS-80 is a trademark of Radio Shack, Inc.
Apple II is a trademark of Apple Computer, Inc.
Atari 400/800 are trademarks of Atari, Inc.
Microsoft is a trademark of Microsoft Consumer Products, Inc.
Z-80 is a trademark of Zilog, Inc.
Visicalc is a trademark of Personal Software, Inc.

MICROTEK inc.

MICROTEK
9514 Chesapeake Drive
San Diego, CA 92123
(714) 278-0633
Outside CA call
Toll Free (800) 854-1081
TWX. 910-335-1269

The Beginner's Page

Loops

Richard Mansfield
Assistant Editor

You'll hear the term *algorithm* from time to time. It merely means a *procedure*, a way of getting something done. For example, let's assume that your programming becomes so impressive that you decide to start a software business. You want to generate a list of possible names for your new venture and then pick out the best one. You could make a list yourself, but you are a programmer and you have a computer which could make your list in a jiffy. All you need to figure out is the algorithm: the steps your computer needs to follow to create the list. Most algorithms, especially for jobs involving lists, use loops.

First put all your favorite words about software into a table of DATA statements. This will give the computer something from which to make its list. Then, you use a *nested loop* to combine the data in all possible ways.

Loop Forms

The *loop* is one of the primary ways that a computer does its work: FOR I = 1 TO 10. (Do something. Print the variable *I*, for example). NEXT I. This structure means: as long as *I* is still between 1 and 10, print *I* on the screen. Raise the value of *I* by one (NEXT I) and *loop* (jump back to the FOR statement which will check to see if *I* is still within bounds). We, ourselves, loop every day (and we ask others to loop for us), but we don't think of it as looping. If you were about to make a list (by hand), you might start off by taking a sheet of paper and writing down the numbers 1. 2. 3. etc. along one side. This is precisely the loop in our example above.

Another common loop form is "please find me the map; it's in that pile." (FOR I = 1 TO 50: IF X$(I) = "MAP" THEN PRINT "HERE IT IS.": NEXT I) Of course, when you use the "IF" structure, you cannot put NEXT I on the same line. If you did, the NEXT part would *only* loop IF X$(I) = "MAP." Anything following IF is governed by that IF and will not be carried out unless the IF comes true.

"Will you please wait two seconds before telling me your name?" (FOR I = 1 TO 2000: NEXT I: PRINT "MY NAME IS COMPUTER.") This is called a *delay loop* because the computer does nothing between the FOR and the NEXT. It just waits until it counts to 2000 which takes about two seconds.

Nesting

If you put a loop within a loop, the inner one is called a *nested loop*. "Ask all six people in this room what their three favorite foods are." (FOR I = 1 TO 6: FOR J = 1 TO 3: PRINT "WHAT'S A FAVORITE FOOD OF YOURS?": NEXT J: NEXT I) It's easiest to grasp nested loops by working from the inner loop out. The *J* loop is asking the question three times before it transfers the control back to its master loop *I*. The total number of loopings (*iterations* is the technical term) will be 18 (*I's iterations multiplied by J's*).

Why do we use *I* for our counting variables in loops? It's just conventional. It must have once meant *increments* or *iterations* or *index*, but that hardly matters. It is convenient because you can then remember never to use *I* elsewhere in your programs for other variables — *I* is always your master loop variable. Then, logically, it is common practice to use *J* for a nested loop within the *I* loop. Also, for timing delay loops, it is a good habit to reserve the variable *T* as we did above. *T*, of course, stands for Time. It is not used anywhere else in programs (for the same reasons).

Picking A Company Name

Our algorithm for listing possible company names is a nested loop. We picked eight adjectives we liked and came up with seven nouns. This means we have two lists which we want to combine into one list. We put the adjectives and nouns into their own separate DATA lines and READ them into two *arrays*. An array is a table or list — a grouping of items which are somehow related to each other so we want them stored together under the same name. In this case we set up two *string arrays:* ADJECTIVE$ #1 through #8 and NOUN$ #1 through #7. The loop in line 120 hangs unique tags on each word in the DATA statement as it reads and memorizes each item. For example, when it READs "super" it tags it with the variable name ADJECTIVE$(2). If you finished RUNning the program and directly asked the computer ? ADJECTIVE$(5) it would print "QUALITY." For information on string arrays on the Atari see **COMPUTE!** #11 pg. 103 and **COMPUTE!** #16 pg. 36.

Knowing that putting a noun before an adjective usually results in nonsense (apple red) we decided to refine our list of potential names for our company by only permitting adjectives to modify nouns. This means we want to list a noun and go through all eight possible adjectives for it before listing the next noun. This is very like asking six people to name three favorite foods.

The nesting is in lines 140 to 180. Notice that

Rocket Raiders

Our High Quality Software Is More Than A Stroke Of Genius... It's A Work Of Art.

☐ **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 (PO$(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

☐ **INTRUDER ALERT!** by Dennis Zander (Atari, 16K)
This is a fast paced action game in which you must escape from the "Dreadstar" with the secret plans. The droids are after you and you must find and enter your ship in order to escape. If you fail, the rebellion is doomed. **PRICE** **$16.95** cassette **$20.95** diskette

☐ **THE RINGS OF THE EMPIRE:** by Dennis Zander (Atari 16K)
The Empire has developed a series of battle stations protected by one or more rings of energy. You must destroy these weapons by attacking them in your Y-wing fighter armed with Zydon torpedoes. Each time you blast through the rings and destroy the station, the Empire develops a new station with more protective rings.
PRICE .... **$16.95** cassette **$20.95** 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

☐ **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

☐ **ALPHA FIGHTER:** by Douglas McFarland (Atari, 16K)
Consisting of two different programs, ALPHA FIGHTER requires you to destroy the alien starships. As you become more successful, the games get harder and harder. **PRICE** **$14.95** cassette **$18.95** 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

☐ **HODGE PODGE:** by Marsha Meredith (Apple 48K, Applesoft or Integer BASIC)
This captivating program is a marvelous learning device for children from 18 months to 6 years. HODGE PODGE consists of many cartoons, animations and songs which appear when any key on the computer is depressed. A must for any family containing young children and an Apple.
PRICE . . . **$19.95** diskette

☐ **STUD POKER:** by Jerry White (Atari, 16K)
This is the classic gambler's card game. You will find the computer to be a worthy opponent who occasionally bluffs but never cheats! STUD POKER employs all of the Atari's sound, color and graphics capabilities.
PRICE .. . . **$14.95** Cassette **$18.95** diskette

TYPE-'N-TALK*

ARTWORX is offering the fantastic TYPE-'N-TALK™ from Vortrax*. This easy-to-use unit connects to your computer's serial port. Text is automatically translated into electronic speech enabling the TYPE-'N-TALK™ hobbyist to use and enjoy it immediately.
PRICE **$329.00**

The following **ARTWORX** programs are available for TYPE-'N-TALK:

STUD POKER (Atari, 24K)	**$16.95** cassette	**$20.95** diskette
TEACHER'S PET (Atari, 24K; North Star)	**$16.95**	**$20.95**
BRIDGE 2.0 (Atari, 24K; North Star)	**$19.95**	**$23.95**
NOMINOES JIGSAW (Atari, 24K)	**$17.95**	**$21.95**

Please specify "TNT" version when ordering programs.

☐ **CRANSTON MANOR ADVENTURE:** by Larry Ledden (Atari, North Star and CP/M)
You must enter mysterious Cranston Manor and attempt to collect its many treasures. This extemely challenging program will provide you with many hours (days?) of adventure. The program may be interrupted at will and your status saved onto the diskette.
PRICE . **$21.95** diskette

☐ **BLOCKADE:** by Edward Schneider (Atari, 16K)
Every games library needs Blockade program, and this is one of the best. Choose from three levels of difficulty and play against another person or by yourself against the clock.
PRICE **$14.95** cassette **$18.95** 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

☐ **FORM LETTER SYSTEM:** (Atari, North Star and Apple)
This is the ideal program for creating personalized form letters! FLS employs a simple-to-use text editor for producing fully justified letters. Addresses are stored in a separate file and are automatically inserted into your form letter along with a personalized salutation. Both letter files and address files are compatible with **ARTWORX** MAIL LIST 3.0 and TEXT EDITOR programs.
PRICE **$39.95** diskette

☐ **TEXT EDITOR:** (Atari and North Star)
This program is very "user friendly" yet employs all essential features needed for serious text editing with minimal memory requirements. Features include common sense operation, two different justification techniques, automatic line centering and straightforward text merging and manipulation. TEXT EDITOR files are compatible with **ARTWORX** FORM LETTER SYSTEM.
PRICE .. .. . **$39.95** 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 Ted 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 .. . **$23.95** cassette **$27.95** diskette

☐ **THE NOMINOES JIGSAW PUZZLE:**
by C. Minns/B. Brownlee (Atari, 24K, TRS-80, and Apple)
We quote . . ."A brainteaser supreme. . the concept of NOMINOES JIGSAW is brillant. . . this video jigsaw game is so clever and completely original that only the most hardhearted puzzle hater could fail to be charmed."—ELECTRONIC GAMES MAGAZINE
PRICE **$17.95** cassette (also available for TRS-80 color computer) **$21.95** diskette.

Artworx
ARTWORX SOFTWARE COMPANY
Highest Quality Software*, Guaranteed.

150 North Main Street Fairport, NY 14450 (716) 425-2833

Call ARTWORX toll-free number to order direct:

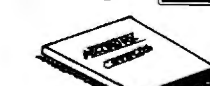
800-828-6573 In New York, Alaska, Hawaii call: (716)-425-2833

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.

*ATARI, APPLE, TRS-80, PET, NORTH STAR, CP/M, and TYPE'N TALK are registed tradenames and/or trademarks the NEXT J will always loop back to line 150 until the FOR J condition (count up to eight) is satisfied. Then the program will execute the NEXT I.

Can we nest at even deeper levels? Sure. Typing a new line: 165 FOR T = 1 TO 2000: NEXT T will provide a short delay loop between each item as it appears on the screen. Could we see the list backwards? Change two lines: 140 FOR I = 7 TO 1 STEP -1 and: 150 FOR J = 8 TO 1 STEP -1. Every other name? 150 FOR J = 1 TO 8 STEP 2. Only names beginning with the letter *A*? 155 IF LEFT$ (ADJECTIVE$(J),1) <> "A" THEN GOTO 170. (For Atari: 165 IF ADJECTIVE$(J*20–19,J*20–19 <> "A" THEN 180)

As you can see, all kinds of choices, refinements, or modifications are possible within loops by merely changing a few instructions to the machine. The combination of loops and *branches* (lines starting with IF or ON) coupled with the computer's great speed (you try to count from one to 2000 in two seconds) is the essence of the great power of computers.

Microsoft Version

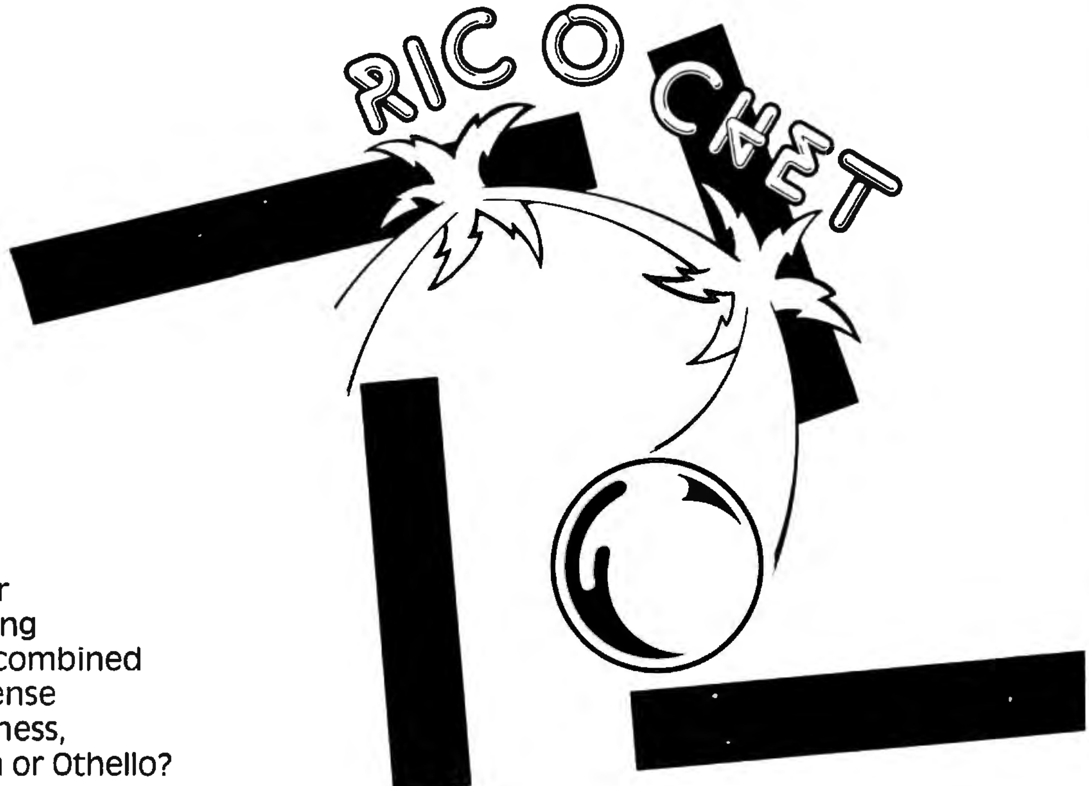
```
100 DATA SUPER,ACME,AMERICAN,RAINBO
    W,QUALITY,INTERGALACTIC,RE
    LIABLE,FOOLPROOF
110 DATA PROGRAMS,SOFTWARE,COMPUTER
    WARE,CODE,LISTINGS,INFORMA
    TION,MAGIC
120 FOR I = 1 TO 8: READ ADJECTIVE$
    (I): NEXT I
130 FOR I = 1 TO 7: READ NOUN$(I):
    NEXT I
140 FOR I = 1 TO 7
150 FOR J = 1 TO 8
160 PRINT ADJECTIVES(J)" "NOUN$(I)
170 NEXT J
180 NEXT I
```

Atari Version

```
100 DATA SUPER,ACME,AMERICAN,RAINB
LITY,INTERGALACTIC,RELIABLE,FOOLPR
110 DATA PROGRAMS,SOFTWARE,COMPUTE
CODE,LISTINGS,INFORMATION,MAGIC
120 DIM ADJECTIVE$(8*20),NOUN$(7*2
P$(20),L1(8),L2(7)
130 FOR I=1 TO 8:READ TEMP$:ADJECT
*20-19,I*20)=TEMP$:L1(I)=LEN(TEMP$
 I
140 FOR I=1 TO 7:READ TEMP$:NOUN$(
9,I*20)=TEMP$:L2(I)=LEN(TEMP$):NEX
150 FOR I=1 TO 7
160 FOR J=1 TO 8
170 PRINT ADJECTIVE$((J-1)*20+1,(J
+L1(J));" ";NOUN$((I-1)*20+1,(I-1)
(I))
180 NEXT J
190 NEXT I
```

©

It's Here! The Computer Strategy Game with Bounce!



For:
Apple
Atari
TRS-80

Have you ever seen an exciting action game combined with the intense strategy of chess, backgammon or Othello? Ricochet...the first abstract strategy game designed **exclusively** for the computer owner... is both. And loads of fun.

You maneuver your blocks, both to protect your own goal from attack **and** to hit your opponent's goal. Two launchers to fire. Your shots ricochet off the blocks, earning you points on the way to their targets. It's twice as challenging because the position changes with both your own and your opponent's moves and shots.

You don't have to play alone, either. Play against any one of four different opponents (each a different personality) inside your computer, or against another human.

And Ricochet is truly competitive...if you want it to be. A "smart clock" lets you put more pressure on your opponent by forcing him to play faster than you. But you've got to win two out of three (or three out of five) games to claim victory. Your computer rates you after each match, so you can compare your mastery of the game with that of other players—perfect for tournament play. So perfect that MIND TOYS and Automated Simulations are sponsoring the first national Ricochet tournament. See your local dealer to find out how you can become a regional or national champion.

Price $19.95

Another Mind Toy From Automated Simulations

P.O. Box 4247, Mountain View, CA 94040.

*APPLE, ATARI and TRS-80 are trademarks of Apple Computer, Inc., Atari, Inc. and Tandy Corp., respectively.

INVEST

Gregory R. Glau
P.O. Box 1627
Prescott, AZ 86302

Editor's Note: Program 1 is the Microsoft version. Program 2 contains the lines which should be changed to permit "INVEST" to run on the Atari. Lines 12000-14999 are the printer routine and might need slight modifications for different printers. For the Atari, change all PRINTs to LPRINT and remove the TAB statements. — RTM

Other than having a place to live, there's only one reason to buy real estate: *to make money*.

INVEST will give you a head start if you're considering this unique investment medium. It'll show you how real estate leverage, inflation, and rent income will all add up to put real cash into your pocket: if you make the right investment decisions.

And once you find some properties to consider, INVEST will detail the benefits to you, to aid you in making that *right* decision.

You may have heard lots of stories about real estate — many are true! — about how you can buy property for little or no cash down, and then let your wonderful tenants pay for the property for you! About how the government allows a *real* tax break for real estate investors (called depreciation) which will put cold cash into your pocket come tax time. And, these days, depreciation works a double advantage for us because, while we're allowed to depreciate the property on our tax returns, the building is actually *increasing* in value every day — often as much as 15 or 20% in a year's time.

Stocks and bonds and gold and jewelry simply can't match this.

There are four areas which give you a return on your realty investment: cash flow, equity build-up, asset appreciation, and tax savings.

The whole purpose of INVEST is to let you figure, by changing the data, exactly what an investment will do for you and detail that information for all four factors (cash flow, appreciation, etc.). Then it's your job to find some properties and use INVEST to help you determine which is the best for you!

Learning The Vocabulary

Cash flow is simply what's left after you collect your rents and then make the payments, pay any expenses, etc. For instance, if your rents (for, say, a duplex you want to buy) are $500 per month, you'll have a total income of $6000 per year ($500 per month x 12 months).

If your monthly payments and expenses total $400 per month, you'll have a yearly cost of $4800 ($400 per month x 12 months). This will give you a positive cash flow of $1200 per year ($6000 collected less the $4800 spent).

Sometimes, particularly with an investment which has a low down payment, you could have a *negative* cash flow. For instance, if your payments plus expenses ran $7000 per year, you'd be $1000 in the hole at the end of the year ($6000 collected less $7000 spent equals a minus $1000). This isn't always bad, as we'll see in a moment.

Equity buildup is the second area where you get a return on your investment. As you make the payments on the property, part of the payment goes for interest, and part for principal. At the start, this interest section eats up most of the payment and, as time passes, the part devoted to principal gets larger and larger.

Note that this is not cash which you'll get as you do when you collect the rents. It's like a savings account — you'll get this part of your investment when you sell the property, because each part of your payment that goes against the principal reduces what you owe on the property.

For instance, if you bought a $100,000 fourplex with $10,000 down, you'd have to borrow $90,000. If you sold it to me tomorrow for, say, $120,000, you'd come out of the deal with a $20,000 profit, right?

However, say you held it for a year and then sold it for the same price. At the end of the year, your payments would have reduced the amount you owed on the property — the actual reduction would depend on the interest rate and length of the loan. But let's say that it, the principal, had been reduced $5,000 over the course of that year. Now, you'd end up with $25,000 (instead of the $20,000 above) — while the extra $5,000 is not profit, it does come back to you, just as if you'd put it into a savings account.

1st example:

Selling price	**$120,000**	
still owe	**– 90,000**	
down payment	**– 10,000**	
cash	**20,000**	**(all profit)**

2nd example (hold the property for a year):

selling price	**$120,000**	
still owe	**–85,000**	**remember — our payments have reduced the principal from $90,000 down to $85,000**
down payment	**–10,000**	
cash	**$ 25,000**	

Of this, $20,000 is your profit and $5,000 represents a return to you of your *equity* just as if you'd saved this money in a bank. And each payment you make (actually it's those wonderful tenants who make the payments for you, right?) increases the equity, your ownership, in the property.

The third way you get a return on your investment is through *asset appreciation*. This is the amount a building increases in value because of rising prices, inflation. In many cases in recent years, income property has gone up in value *faster* than the rate of inflation.

And, as you may already know, there are two types of inflation: *normal* inflation and *forced* inflation. But even if normal inflation slows down — and don't bet that it will — you can use *forced inflation* ... fixing up a property to make it rent for more, thus making it worth more.

Perhaps you could buy a property for, say, $65,000 and with some paint and carpeting and cleaning *increase* its value to $75,000. And, of course, you can always *combine* the two types of inflation, and really increase the return on your investment.

Finally, *tax savings* is the fourth method of return on a real estate investment. Tax savings stems from depreciation, the concept that everything wears out and thus, at some time in the future, it will have to be replaced. The Congress of the US has recognized this fact, particularly in regard to investment real estate, and allows the owner of such property to depreciate a part of the building and of its contents each year (just as if he took X amount of cash and put it into a bank) to help pay for the replacement cost of the building or contents.

Depreciation is based on what accountants call "useful" life, and varies on a building with its age, condition, etc. An old building might have a "useful life" of only 10 or 12 years, while a new structure might be expected to last 30 years.

The actual length for depreciation for any particular property must be determined *by your accountant.*

Obviously, the shorter the "useful life," the more depreciation you can take per year, and the more the tax savings will be.

For instance, let's picture that you bought (or want to buy) a triplex which will cost you $100,000. First, we have to deduct the value of the land — land cannot be depreciated, it doesn't wear out. Let's say that you figure, from tax records and property comparisons, that the land value is about 15% of the total purchase price. This means the land cost was $15,000 (15% of $100,000). Deducting this from the purchase price of $100,000, you now have $85,000 left.

Now, the carpeting drapes, appliances, and so on will wear out faster than the building, so you're allowed a faster rate of depreciation on these items. Again, *ask your accountant.* In INVEST, we figure that about half the value of the furnishings are in items that have a three year "useful life" for depreciation, and then about half the value is in items that would have a seven year "useful life," so we've taken them and lumped them together, and figured an average of a five year useful life.

In this example, if you have furnishings worth 5% of the value of the property, you'd have furnishings worth $5,000 (5% of $100,000).

So, you deduct the value of the furnishings ($5,000) from the net property value (after the land has been removed) of $85,000, which gives you a net building value of $80,000.

Let's further assume that your accountant tells you that this building has a "useful life" of 20 years.

Now, to figure the depreciation: you have an $80,000 building, with a life of 20 years...you simply divide the value by the years, to get a per-year amount for depreciation. $80,000 divided by 20 years equals $4,000 per year. This is the amount of depreciation per year allowed on this building.

INVEST takes things a step farther, by asking you how many months this year you'll own this property. It will then give you two displays and printouts — one for this year, the number of months you'll own the property, and then for next year, which is figured at a full twelve months. Obviously, if you're buying the building in June, you wouldn't own it for a full year, so INVEST automatically will calculate the exact depreciation (and tax savings) for the part of the year you'll actually own the property.

Added to the building depreciation is the depreciation you're allowed on its contents. Remember that we had $5,000 worth of carpeting, drapes, appliances, and so on. We're using an "average useful life" of five years, so we divide the amount of $5,000 by five years, for an allowable depreciation of $1,000 per year on the building's contents.

Total depreciation, then, is the building depreciation of $4,000 per year plus the contents depreciation of $1,000 ... for a total of $5,000 per year.

The Tax Savings

This is the amount you can deduct from your income tax. To figure your tax savings (how much *less* you'll have to pay in taxes, or how much *cash* they'll send back to you), multiply your tax bracket by the amount of depreciation.

For example, if you're in the 30% tax bracket, you'd save 30% of $5,000 depreciation, or $1,500 on your taxes.

A Wooden Computer?

Not from Commodore!

So why should the desk look like wood? A pleasant cream and charcoal trimmed desk looks so much better with Commodore systems. One look and you'll see. Interlink desks are right. By design.

The specifications only confirm the obvious:

- •Cream and charcoal color beautifully matches the Commodore hardware and blends with your decor.
- •An ideal 710 mm (28") keyboard height yet no bumping knees because a clever cutout recesses the computer into the desk-top.
- •High pressure laminate on both sides of a solid core for lasting beauty and strength.
- •Electrostatically applied baked enamel finish on welded steel legs—no cheap lacquer job here.
- •T-molding and rounded corners make a handsome finish on a durable edge that won't chip.
- •Knocked down for safe, inexpensive shipment.
- •Patented slip joints for quick easy assembly.
- •Leveling glides for uneven floors.
- •Room enough for a Commodore printer on the desk, yet fits into nearly any den or office niche—H: 660 mm (26") W: 1170 mm (46") D: 660 mm (26").
- •Matching printer stand available with slot for bottom feeding.

Price: $299

In short, as Commodore dealers, we won't settle for anything that looks good only in the catalog! Our customers won't let us. They don't buy pictures. And neither should you. This is why **we will let you use one of our desks for a week** and then decide. If for any reason you don't like it, just return it in good condition for a cheerful refund.

If your Commodore dealer doesn't carry our desks yet, send a check for $299 and we will ship your desk freight paid!

Name____________________

Address____________________

City__________ **St**_____ **Zip**_____

Interlink, Inc., Box 134, Berrien Springs, MI 49103
Master Charge and Visa welcome. Call our order line:
616-473-3103

CONNECT

Your Commodore PET/CBM, HP-85, Osborne-1 to any RS-232 Serial Printer, Plotter, CRT Terminal, Modem, or other device.

TNW-2000

TNW-1000	ONE CHANNEL OUTPUT ONLY	**$129**
TNW-2000	ONE CHANNEL INPUT/OUTPUT	**$229**
TNW 232D	TWO CHANNELS INPUT/OUTPUT + RS-232 Control Signals	**$369**
TNW-103	AUTO ANSWER/AUTO DIAL (use with DAA)	**$389**

All units are addressable IEEE-488 devices. Comes complete with cabinet, documentation, one year warranty. Brochure available other products.

TNW CORPORATION™

3444 Hancock St., Dept. C, San Diego, CA 92110
(714) 296-2115 TWX 910-335-1194

VISA/Mastercharge Welcome • Dealer inquiries invited

HAVE YOUR CAKE AND EAT IT TOO

As Low As $690†

IT'S A PIECE OF CAKE TO CONNECT AN Actek TRIX I INTERFACE TO YOUR OLIVETTI PRAXIS 30 OR 35 CORRECTING ELECTRONIC TYPEWRITER. DAISY WHEEL QUALITY AT DOT MATRIX PRICES!

- 10+ CPS
- 15 MINUTE INSTALLATION
- HALF SPACE JUSTIFICATION
- CABLE REMOVES IN SECONDS
- TYPEWRITER FUNCT. UNIMPAIRED
- AVAILABLE NOW: ATARI & APPLE
- OTHER DIRECT CONNECTIONS AND RS232 AVAILABLE SOON
- PRINT AND PRINT *N OPERATE
- NO INTERFACE NEEDED: USES FRONT CONNECTOR-ATARI USES CONTROLLER PLUG-APPLE
- PRICE: $215 - APPLE ADD $10 †BASED ON WARDS PR30 PRICE
- TYPEWRITER AND SERVICE WIDELY AVAILABLE

Actek

12225 SW 2nd/SUITE 200 -E
P.O.B. CCC
BEAVERTON, OR 97075

It's important to note that these savings — tax savings — return to you in the form of cash, either in a tax refund or because you will pay less in taxes than you would have had to without them, as compared to *equity buildup* and *asset appreciation*, which return to your wallet only when you sell the property.

In fact, the tax savings are often enough to offset any negative cash flow you might have from a property. So, while you might have $100 a month in negative cash flow, perhaps at the end of the year you'll get back that cash, just as if you'd saved it in a bank every month!

INVEST will show you exactly what your results will be.

So, these are the four vital areas we need to consider for any real estate investment: cash flow, equity buildup, asset appreciation, and tax savings.

"INVEST" will not only show you what each item will do, based on your own data, but will also summarize and total them, compare them to your down payment, and provide you with a return on your investment percentage.

And this, your *return on investment*, is really the important figure for any investment program. All the various parts of any investment, such as the real estate we've been examining here, are combined in this figure.

It's fascinating to see this in action, because many of us think in terms of savings accounts (perhaps 5 or 6% on our money) ... or certificates of deposit (perhaps 14%) ... or stocks (what will the market do tomorrow?) ... or limited edition prints (nice to hang on the wall, but who can we sell it to?) ... or money market accounts (10 or 12%) ... and when you see what real estate can do for you — even a small duplex or triplex — you will be astonished.

Using INVEST

Once you get a printout of a specific set of data, the program will automatically end. Up to that point — before you ask for a printout — you can alter any data any number of times, to display different results. Then, when you have the display you want, you can request a hard copy.

There's a delay at the end of page two of the instructions: while you're reading them, your computer is reading array information into its RAM. We're using three double-dimension arrays:

Q is the information used to get your monthly payment. The program will multiply the amount of your loan(s) by the proper monthly figure, to arrive at a monthly payment. You can input up to 3 loans, for 15, 20 or 25 years, and at interest rates from 10% to 18%, in .5% steps. Then "INVEST" will total the payment, display it, and let you change the amount, if you wish to. This situation might occur if you happen to be assuming an old loan, at less than 10% — you can answer 10%, and then change the payment total to match your correct figure.

E1 ... is the array with the figures for the first-year equity buildup.
E2 ... is the array for the second-year equity buildup.

Two arrays are used here because the equity buildup is different for each year — you will pay *more* on the principal of your loan during the second year than you did the first. The actual multipliers are based on the length and terms of your loan — a loan at 10% for 15 years will have a much faster and higher equity buildup than one at 16% for 25 years.

Total rents are just that — if the property you're considering is a fourplex, input total rents from *all* units.

If you don't know the actual amount of taxes and insurance, or expenses, use your best estimate.

Your accountant will know your approximate tax bracket, or you can check the tables on Form 1040, or look back at your latest tax return.

When you're asked to input payment information, you must input something — if you skip around and just put the payment amount in (without the interest rate or length of the loan), you won't get credit for any equity buildup — the computer just can't tell what equity buildup will actually be *better* than what's shown.

Following is a list of the major variables used in INVEST. There are others used mathematically, so if you change the program, please read through it to make sure you don't use something already used.

Table 1.
INVEST
Major variables:

E$	... date
A$	... property address
M1$	.. misc. information (1)
M2$	.. misc. information (2)
PR	... asking/purchase price of the property
L	.... estimated life for depreciation
A	.... % estimated annual asset appreciation
R	.... current rents
AR	... anticipated rents
M	.... months of ownership this year
T	.... estimated taxes and insurance per month
E	.... estimated expenses per month
V	.... % land value (as a percent of the price)
B	.... the tax bracket you're in
DP	... down payment amount
F	.... % furnishing's value (% of the price)
F1	... first year cash flow
F2	... second year cash flow (full year)
EB	... equity buildup, first year
ET	... equity buildup, second year
A5	... asset appreciation, first year
A6	... asset appreciation, second year

HUNTINGTON COMPUTING

ONE OF THE WORLD'S LARGEST INVENTORIES

GAMES

Golden Mountain	$19 95 now	$16.89
Space Eggs	$29 95 now	$19.99
Apple® Panic	$29 95 now	$19.99
Thief	$29 95 now	$25.39
Snack Attack	$29 95 now	$25.39
Med Fly Mania	$29 95 now	$24.59
The Book	$19 95 now	$17.99
Hi-Res Soccer	$29 95 now	$25.39
Apple®-Oids	$29 95 now	$25.39
Wurst of Huntington Computing		**$19.99**
Ultima	$39 95 now	$33.89
Autobahn	$29 95 now	$25.39
Battle Cruiser Action	$44 95 now	$38.89
Gorgon	$39 95 now	$33.89
Super Stellar Trek	$39 95 now	$33.89
Hellfire Warrior	$39 95 now	$33.99
Gamma Goblins	$29 95 now	$25.39
Mission Asteroid	$19 95 now	$17.99
Wizardry	$49 95 now	$42.49
Warp Factor	$39 95 now	$33.99
Microsoft Adventure	$29 95 now	$25.39
Wizard and the Princess	$32 95 now	$28.99
Flight Simulator	$34 95 now	$29.49
Odyssey	$29 95 now	$25.39
Sargon II	$34 95 now	$29.69
Space Eggs	$29 95 now	$25.39
Hi-Res Cribbage	$24 95 now	$21.19
Lords of Karma (cass)	$20 00 now	$16.99
Oh Shoot	now	$19.99
ABM	$24 95 now	$21.19
Computer Conflict	$39 95 now	$35.99
Computer Air Combat	$59 95 now	$52.99
Temple of Apshai	$39.95 now	$33.89
Zork	$39 95 now	$33.99
All Nibble Software		**15% off list**
Robot Wars	$39 95 now	$33.99
Cranston Manor	$34 95 now	$29.69
Dragon s Eye	$24 95 now	$21.19
Twala s Last Redoubt	$29 95 now	$25.39
Snoggle	$24 95 now	$21.19
Alien Rain	$24 95 now	$21.19
Alien Typhoon	$24 95 now	$21.19
Raster Blaster	$29 95 now	$25.39
Creature Venture	$24 95 now	$21.19
Hodge Podge	$23 95 now	$21.29
Meteoroids in Space	$19 95 now	$16.99
Dragon Fire	$49 95 now	$42.49

MISCELLANEOUS

Escape from Acturus	$29 95 now	$25.39
Basic Mailer	$69 95 now	$59.49
Memory Management II	$49.95 now	$42.89
Castle Wollenstein	$29.95 now	$25.39
Upper Reaches of Apshai	$19.95 now	$16.89
Bridge Tutor	$39 95 now	$35.19
PASCAL Animation Tools		**$65.99**
Hand Holding BASIC		**$84.99**
Crossfire	$29.95 now	$26.29
Epoch	$29 95 now	$25.39
Outpost	$29.95 now	$26.29
Ulysses	$34 95 now	$29.69
Space Quark	$29 95 now	$25.39
Beneath Apple DOS (book)	$19 95 now	$16.89
Birth of the Phoenix	$14 95 now	$12.69
Goblins	$27 50 now	$23.29
Painter Power	$39 95 now	$33.89
U S Constitution	$29 95 now	$25.39
Merger	$49 95 now	$42.49
Super Stellar Trek	$39 95 now	$33.89
LISA	$79 95 now	$67.89
V-Plot (Yuccipa)	$29 95 now	$25.39
V-Stat	$29 95 now	$25.39
V-Print	$29 95 now	$25.39
Brain Surgeon	$49 95 now	$42.49
Info Master	$150 00 now	$127.49
Waterloo II	$49 95 now	$42.39
Speedstar	$139 95 now	$114.69
Kaves of Karkhan	$49 95 now	$42.39
Dos Boss	$24 00 now	$20.39

WORD PROCESSORS

Apple® Writer		$65.99
Magic Window	$100 00 now	$84.99
Easy Writer Professional	$250 00 now	$219.00
Letter Perfect	$150 00 now	$127.49
Super Text	$150 00 now	$127.49
Superscribe	$129 95 now	$110.39
Executive Secretary	$250 00 now	$212.49
Apple® Wordstar	$375 00 now	$286.00
Hebrew II	$60 00 now	$50.99
Apple® Writer Extended	$29 95 now	$25.39

Softlights

By Fred Huntington

More and more fantastic things are becoming available for the Apple®. We just received one of my favorites – the **Passport Designs SOUNDCHASER Computer Music System.** It's easy to set up and an absolute blast to play with.

Price is $650 for the keyboard and $350 for each synthesizer board.

We recommend two boards. When you buy the complete system, we will give you $100 worth of free software (your choice from our current catalog).

Having worked my way through college playing in a small band, I keep thinking how great my group would have sounded if we had had the Passport Designs setup.

You can actually record a song and then play it back and even play along with it – no tapes or accessing the disk. You have to see it (and hear it) to believe it. Five stars!

HARD DISK

We're now official dealers for Nestar. We think the Nestar system is so great, we've adopted it for in-house use. We have four Apples hooked up to a 16 Megabyte disk. This allows us to have several people processing orders at the same time, all working off the same inventory disk.

Our system has the capability of expanding to 64 Apples hooked up together.

And, by the time you read this, CP/M® (a registered trade mark of Digital Research) should be available for the system.

Call for more information.

COMING SOON!
California Toll-Free Number
800-692-4143

STOCK PROGRAMS

Portfolio Master	$75 00 now	$63.69
Market Charter	$129 95 now	$110.39
Dowlog for Market Charter	$99 95 now	$89.99
Investment Decisions	$99 95 now	$84.99
Stock Tracker	$190 00 now	$161.49
Stock Tracker (Auto Ver)	$300 00 now	$254.99

BUSINESS APPLICATIONS

Invoice Factory (Special)	$200 00 now	$149.00
Regression Trend Analysis	$26 95 now	$22.89
Multiple Regression	$29 95 now	$25.39
Microsoft Fortran	$200 00 now	$165.69
Microsoft Cobol-80	$750 00 now	$637.49
Business Pac 100	$99 95 now	$84.99
Desktop Plan II	$200 00 now	$169.99
Visicalc 3 3 Special	$200 00 now	$149.00
Visiplot	$179 95 now	$152.89
Visitrend/Visiplot	$259 95 now	$220.89
Visiterm	$149 95 now	$127.39
Complete Mailing (Avant-Garde)	$59 95 now	$50.89
DB Master	$229 95 now	$194.59
PFS	$95 00 now	$80.69
PFS Report	$95 00 now	$80.69
Data Factory	$150 00 now	$124.00
Thinker	$495 00 now	$420.69
Request	$225 00 now	$191.19
Super Kram	$175 00 now	$148.69
ASCII Express	$99 95 now	$84.99
BPI Accounts Receivable	$395 00 now	$335.69
BPI General Ledger	$395 00 now	$335.69
The Mail Room	$29 95 now	$25.39
Continental General Ledger	$250 00 now	$212.49
Cont Accounts Receivable	$250 00 now	$212.49
Cont Accounts Payable	$250 00 now	$212.49
Continental Payroll	$250 00 now	$212.49
Broderbund Payroll	$395 00 now	$335.69
Informer II	$49 95 now	$42.49
Creative Financing	$150 00 now	$127.49
Real Estate Analyzer	$150 00 now	$127.49
Accounting Assistant (cass)	$7 95 now	$6.89
Spell Star	$250 00 now	$212.49
Muse Form Letter	$100 00 now	$84.99
Property Management System	$225 00 now	$191.19

PERSONAL/HOME

Interlude (disk)	$19 95 now	$16.99
Handwriting Analysis	$19 95 now	$16.89
Decision Master	$29 95 now	$25.39
The Correspondent	$44 95 now	$38.19
Diet Planning	$24 95 now	$21.19
Win at the Races	$39 95 now	$33.89
Pro Football (SDL)	$26 95 now	$22.89
College Football (SDL)	$26 95 now	$22.89
Grocery List	$19 95 now	$16.89
Financial Management System II	$39 95 now	$33.89
Home Money Minder	$34 95 now	$29.69

MISCELLANEOUS

D C Hayes Micromodem	$375 00 now	$299.00
D C Hayes Smart Modem	$279 00 now	$249.00
460G Paper Tiger	$1094 00 now	$961.99
560G Paper Tiger	$1394 00 now	1225.00
Tiger Trax	$16.95 now	$15.25
Z-80 Softcard	$395 00 now	$299.00
NEC 12 Green on Black	$260 00 now	$209.00
Videx 80-col	$350 00 now	$299.00
TG Joystick	$59 95 now	$50.89
TG Game Paddles	$39 95 now	$33.89
Paymar LCA Rev. 7		**$29.69**
Don t Book	$8 95 now	$8.05
Dragon Fire	$49 95 now	$42.49
Hungry Boy	$24 95 now	$21.19
Ring of Saturn	$39 00 now	$33.89
Streets of the City & Trucker	$24 95 now	$21.19
Race for Midnight	$29 95 now	$25.39
Galactic Empire	$24 95 now	$22.99
Space Warrior	$24 95 now	$21.19
Pulsar II	$29 95 now	$25.39
Star Cruiser	$24 95 now	$20.99
Both Barrels	$24 95 now	$21.89
Cyber Strike	$39 95 now	$36.99
Phantom 5	$29 95 now	$25.39
Memorex Disks		10 for $24.99
Verbatim Datalife-plain w/hubs		10 for $27.99
Dysans		10 for $49.99
Flipsort Box		$29.80
Scotch Disk Cleaner	$29 95 now	$26.99
E-Z Port	$24 94 now	$21.19
Atari®/TRS-80®/Pet®		**Write for information**

Call Toll-Free 800-344-4111 (Outside California)

HUNTINGTON COMPUTING
Post Office Box 1235
Corcoran, California 93212

Order by Phone 800-344-4111
In California (209) 992-5411

Apple® is a registered trademark of Apple Computer, Inc
Pet® is a registered trademark of Commodore
TRS-80® is a registered trademark of Tandy Corp
Atari® is a registered trademark of Atari, Inc

VISA

We take MasterCard or VISA (Include card # and expiration date) California residents add 6% tax Include $2 00 for postage Foreign and hardware extra Send for free catalog Prices subject to change

Figure 1: Sample Run

```
10/22/81
PROPERTY ANALYSIS REPORT FOR A SAMPLE FOURPLEX INVESTMENT

*** PREPARED FOR COMPUTE! MAGAZINE ***

ASKING/OFFERING PRICE 100,000.00


*******************************************************************************
CASH FLOW ESTIMATE, BASED ON OWNING THIS PROPERTY FOR 6 MONTHS
THE FIRST YEAR, 12 MONTHS THE SECOND YEAR. FIRST YEAR CASH FLOW
BASED ON CURRENT RENTS OF 1200 MONTHLY, AND THE 2ND YEAR IS BASED ON
ANTICIPATED RENTS OF 1300 PER MONTH. ESTIMATED
APPRECIATION IS 8 %.
----------------------------------------
ALL FIGURES ARE APPROXIMATE
----------------------------------------

                        1ST YEAR          2ND YEAR
MONTHLY RENTS           7200.00           15,600.00

MORTGAGE PAYMENTS       7338.01           14,676.01
TAXES + INSURANCE        240.00              480.00
MISC. EXPENSES           300.00              600.00

ESTIMATED CASH FLOW     -678.01             -156.01

==============================================================================

RETURN ON INVESTMENT ANALYSIS
                                     1ST YEAR          2ND YEAR

CASH FLOW (FROM ABOVE )              -678.01           -156.01
ASSET APPRECIATION                   4000.00           8000.00
EQUITY BUILDUP (APPROXIMATE)          242.21            567.86

  YOUR ESTIMATED TAX SAVINGS ARE
  BASED ON A TAX BRACKET OF 30 %
  AND A LIFE FOR DEPRECIATION
  OF 20 YEARS. DEPRECIATION
  THE FIRST YEAR IS 2500 AND
  THE 2ND YEAR IS 5000
  THE FURNISHINGS ARE WORTH 5
   % OF THE PROPERTY COST.

TAX SAVINGS                           750.00           1500.00
------------------------------------------------------------------------------
YOUR RETURN ON INVESTMENT IS         4314.20           9911.85

YOUR DOWN PAYMENT WAS $ 10000
YOUR % RETURN ON INVESTMENT IS         43.14 %           99.12 %
```

Program 1.

```
4 GOSUB21000
5 GOSUB17000
7 REM PAYMENT PERCENTAGE FIGURES
   ARE HERE
8 REM TO GET THE MONTHLY PAYMENT,
   MULTIPLY THE
9 REM AMOUNT OF THE LOAN BY THE F
   IGURE.
```

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

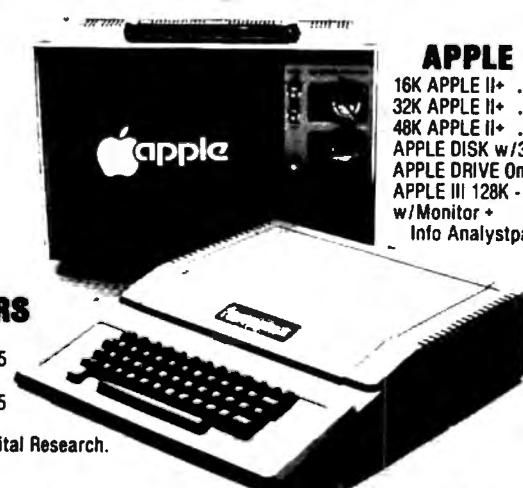
CALL NEECO FOR ANY OF YOUR COMMODORE COMPUTER NEEDS

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

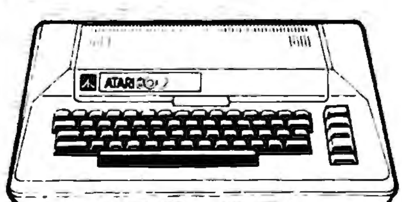
AMDEK MONITORS

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

```
10 DATA 10.746,9.650,9.087,11.054,
   9.984,9.44
12 DATA 11.366,10.3219,9.8013
13 DATA 11.6919,10.6643,10.1647
14 DATA 12.0017,11.0109,10.5323,12
   .3253,11.3615,10.9036
16 DATA 12.6525,11.7158,11.2784,12
   .9832,12.0738,11.6565
18 DATA 13.3175,12.4383,12.0377,13
   .6551,12.8,12.4217
20 DATA 13.9959,13.1679,12.8084,14
   .34,13.5389,13.1975
22 DATA 14.6871,13.9126,13.5889,15
   .0371,14.2891,13.9825
23 DATA 15.3901,14.6681,14.378
24 DATA 15.7458,15.0495,14.7753,16
   .1043,15.4332,15.1743
30 DIM Q(18,4)
32 DIM E1(18,4)
34 DIM E2(18,4)
40 FORY=1TO17
50 FORI=1TO3
60 READQ(Y,I)
70 NEXTI
80 NEXTY
100 REM EQUITY FIRST YEAR BUILD-UP
110 DATA 30.3165,16.5472,9.4702,29.
    0169,15.5381,8.713
120 DATA 27.7628,14.5823,8.009,26.5
    522,13.6764,7.3549
130 DATA 25.3853,12.8195,6.7496,24.
    2612,12.0094,6.1886
140 DATA 23.1776,11.2426,5.6695,22.
    132,10.5195,5.1902
150 DATA 21.1306,9.8378,4.7482,20.1
    653,9.1939,4.3401
160 DATA 19.2365,8.5881,3.9648,18.3
    459,8.0195,3.6187
170 DATA 17.4898,7.4830,3.3007,16.6
    672,6.9813,3.0097
180 DATA 15.8796,6.5076,2.7419,15.1
    234,6.0639,2.4964
190 DATA 14.4001,5.6481,2.2718
200 FORY=1TO17
210 FORI=1TO3
220 READ E1(Y,I)
240 NEXTI
250 NEXTY
300 REM EQUITY BUILDUP FOR 2ND YEAR

310 DATA 33.4911,18.2799,10.4619,32
    .2146,17.2504
315 DATA 9.6733
320 DATA 30.9755,16.2697,8.9358,29.
    7719,15.3348,8.2467
330 DATA 28.6048,14.4453,7.6056,27.
    4737,13.5996,7.0081
340 DATA 26.3768,12.7945,6.4521,25.
    312,12.0309,5.9359
360 DATA 24.2863,11.3070,5.4573,23.
    2916,10.6193,5.0130
380 DATA 22.3289,9.9687,4.6022,21.4
    005,9.3547,4.2213
390 DATA 20.5028,8.7721,3.8693,19.6
    351,8.2232,3.5456
400 DATA 18.7997,7.7043,3.2462,17.9
    929,7.2145,2.9700
410 DATA 17.2170,6.7530,2.7162
420 FORY=1 TO 17
430 FORI = 1 TO 3
440 READE2(Y,I)
460 NEXTI
470 NEXTY
2000 PRINT"HIT ANY KEY TO CONTINUE..
     .";:GETL$
2004 HOME:PRINT
2005 PRINT:INVERSE:PRINT TAB(17)"INV
     EST   ":NORMAL:PR:PRINT"PL
     EASE ANSWER THE FOLLOWING
2007 PRINT
2008 INVERSE:PRINT"ANSWER 'END' TO S
     TOP NOW":NORMAL:PRINT:PRIN
     T
2010 INPUT"TODAY'S DATE";E$
2015 IFE$="END"THENPRINT"   END OF P
     ROGRAM   ":END
2020 PRINT
2030 INPUT "PROPERTY ADDRESS";A$
2032 HOME
2033 PRINT"MISC. INFORMATION IS ANY
     DATA THAT"
2034 PRINT"YOU'D LIKE LISTED ON THE
     PRINTOUT,"
2036 PRINT"PROPERTY (DUPLEX, TRIPLEX
     ), AND SO"
2037 PRINT"ON.  IF YOU DON'T WANT AN
     YTHING PRINTED"
2038 PRINT"FOR MISC. INFO, JUST HIT
     RETURN.":PRINT
2040 PRINT:INPUT"MISC INFO(1)";M1$
2050 INPUT"MISC INFO(2)";M2$
2060 PRINT:PRINT:INPUT"ASKING/OFFERI
     NG PRICE ";PR
2065 IFPR<1THEN2060
2070 HOME:PRINT
2072 PRINT"DEPRECIATION, THE 'WEARIN
     G-OUT' OF"
2073 PRINT"A PROPERTY, IS WHERE THE
     MAJOR"
2074 PRINT"TAX SAVINGS FROM A REAL E
     STATE"
2075 PRINT" INVESTMENT COME FROM.  "
     :PRINT
2076 PRINT"CONSULT WITH YOUR ACCOUNT
     ANT -- ";:INVERSE:PRINT"PL
     EASE";:NORMAL
2077 PRINT" AS TO THE USEFUL LIFE OF
     THIS"
2078 PRINT" PROPERTY.  NATURALLY, TH
     E SHORTER"
2079 PRINT" THE BETTER. AS THE SHORT
     ER PERIOD WILL SAVE MORE I
```

```
    N TAXES.":PRINT
2080 PRINT" YOU MIGHT ALSO WANT TO C
    HANGE THE BASIS FOR THE DE
    PRECIATION"
2081 PRINT" IN THIS PROGRAM.":PRINT:
    PRINT"TO SEE WHAT THE DIFF
    ERENCE IS IN"
2082 PRINT" TAX SAVINGS FOR, FOR INS
    TANCE, 15 YEARS OR 20 YEAR
    S OR 25 YEARS, ETC."
2083 INPUT "ESTIMATED LIFE FOR DEPRE
    CIATION IN YEARS ";L:PRINT

2084 IFL < 1 THEN 2070
2085 HOME:PRINT
2086 PRINT"APPRECIATION IS WHAT INFL
    ATION WILL":PRINT"DO TO A ~
    PROPERTY.  IF YOU THINK
2087 PRINT" THIS MIGHT GO UP IN VALU
    E 10% PER YEAR, ANSWER 10.
      YOU'LL HAVE THE
2088 PRINT" CHANCE TO CHANGE THIS AN
    SWER LATER ON, SO YOU'LL B
    E ABLE TO SEE WHAT
2089 PRINT" DIFFERENT INFLATION RATE
    S WILL DO TO YOUR RETURN."
    :PRINT
2090 INPUT "ESTIMATED APPRECIATION P
    ER YEAR ";A
2091 HOME:PRINT
2092 INPUT"CURRENT TOTAL RENTS PER M
    ONTH ";R
2093 HOME:PRINT
2100 INPUT"ANTICIPATED TOTAL RENTS P
    ER MONTH ";AR
2102 HOME:PRINT
2108 PRINT
2110 PRINT"HOW MANY MONTHS WILL YOU ~
    OWN THIS"
2115 PRINT"PROPERTY THIS YEAR ? ";M
2116 IF M > 12 THEN 2108
2117 IF M < 0 THEN 2108
2118 PRINT:PRINT
2120 INPUT"TAXES + INSURANCE PER MON
    TH ";T
2130 PRINT
2140 INPUT "ESTIMATED EXPENSES PER M
    ONTH ";E
2141 PRINT:PRINT:PRINT"YOU CAN'T DEP
    RECIATE THE LAND, SO":PRIN
    T"THE VALUE OF THE LAND HA
    S TO BE
2142 PRINT"DEDUCTED FROM THE TOTAL P
    RICE, BEFORE
2143 PRINT"THE DEPRECIATION CAN BE C
    ALCULATED.":PRINT
2144 PRINT"AS A PERCENT OF THE TOTAL
     PRICE":PRINT"(10%=10...15
    %=15, ETC.)";:INPUT V
2145 V5=V:V5=INT(V5*10^2+.5)/100
2147 HOME:PRINT
2150 INPUT "%TAX BRACKET YOU'RE IN (
    30%=30) ";B
2155 IF B<0 THEN 2147
2156 IF B>100 THEN 2147
2161 PRINT:PRINT"(YOUR DOWN PAYMENT ~
    MUST BE AT LEAST":PRINT"$1
    --FOR MATH PURPOSES":PRINT

2162 PRINT:INPUT"DOWN PAYMENT";DP
2163 IF DP<1 THEN 2161
2166 PRINT:PRINT"(20%=20  30%=30) ET
    C.)
2167 PRINT"FURNISHING AS A PERCENT O
    F THE PRICE":INPUT F
2168 IF F>100 THEN 2167
2169 IF F<0 THEN 2167
2170 PRINT
2185 F=INT(F*10^2+.5)
2190 GOSUB6000
2195 GOSUB5000:REM  INPUT PAYMENT DA
    TA
2200 HOME:PRINT
2210 PRINT "TOTAL MONTHLY PAYMENT ";

2215 FOR C=1 TO 3
2220 P(9)=P(9)+P(C)
2230 NEXT C
2240 Z9=P(9):GOSUB 15000
2250 PRINT Z9$
2280 PRINT
2290 INPUT "DO YOU WANT TO CHANGE TH
    IS <1=YES>";Q
2300 IF Q=1 THEN 2400
2310 GOTO 2420
2400 REM CORRECT PAYMENT AMOUNT
2410 PRINT:INPUT "CORRECT PAYMENT TO
    TAL ";P(9)
2420 GOSUB7000
2430 GOTO9000:REM PRINT
3032 HOME:PRINT
5000 HOME:PRINT:PRINT"NOW WE HAVE TO
     FIGURE YOUR"
5001 PRINT"MONTHLY PAYMENT FOR THIS ~
    PROPERTY.":PRINT:PRINT"YOU
     CAN INPUT UP TO 3 PAYMENT
    S":PRINT
5002 P(8)=0:P(3)=0
5003 PRINT"IF YOUR PAYMENT DATA IS D
    IFFERENT":PRINT"THAT WHAT ~
    IS ASKED FOR, ANSWER
5004 PRINT"AS CLOSELY AS YOU CAN.":P
    RINT
5005 PRINT:INVERSE:PRINT"YOU MUST IN
    PUT SOMETHING"
5006 PRINT"--EVEN IF YOU CHANGE IT L
    ATER ON"
5007 NORMAL:PRINT
5010 PRINT:PRINT:PRINT"ANSWER 1 TO C
    ONTINUE...
5015 PRINT"ANSWER 2 WHEN DONE....."
5020 PRINT:INPUT Q
```

```
5030 IF Q=1 THEN 5200
5040 IF Q=2 THEN 5500:REM RETURN
5050 GOTO 5007
5200 REM TO ZERO OUT ALL PRIOR PAYME
    NT DATA
5210 C=C+1:REM COUNTER
5250 INPUT "YEARS (15-20-25)";Y(C)
5260 IF Y(C)=15 THEN I=1:GOTO 5300
5270 IF Y(C)=20 THEN I=2:GOTO 5300
5280 IF Y(C)=25 THEN I=3:GOTO 5300
5285 GOTO 5250
5300 PRINT:PRINT"YOUR RATE CAN BE FR
    OM 10 TO 18"
5305 PRINT"IN .5 STEPS.":PRINT:PRINT

5310 INPUT "PERCENT RATE ";Q
5315 IF Q=10 THEN Y=1:GOTO 5400
5320 IF Q=10.5 THEN Y=2:GOTO 5400
5330 IF Q=11 THEN Y=3:GOTO 5400
5332 IF Q=11.5 THEN Y=4:GOTO 5400
5334 IF Q=12 THEN Y=5:GOTO 5400
5336 IF Q=12.5 THEN Y=6:GOTO 5400
5338 IF Q=13 THEN Y=7:GOTO 5400
5340 IF Q=13.5 THEN Y=8:GOTO 5400
5350 IF Q=14 THEN Y=9:GOTO 5400
5352 IF Q=14.5 THEN Y=10:GOTO 5400
5354 IF Q=15 THEN Y=11:GOTO 5400
5356 IF Q=15.5 THEN Y=12:GOTO 5400
5358 IF Q=16 THEN Y=13:GOTO 5400
5360 IF Q=16.5 THEN Y=14:GOTO 5400
5370 IF Q=17 THEN Y=15:GOTO 5400
5372 IF Q=17.5 THEN Y=16:GOTO 5400
5382 IF Q=18 THEN Y=17:GOTO 5400
5390 GOTO 5300
5400 INPUT "AMOUNT OF LOAN ";A(C)
5410 REM   TO FIGURE PAYMENT AMOUNT
5420 P(C)=A(C)*Q(Y,I)
5425 P(C)=P(C)/100:REM TO PUT DECIMA
    LS IN THE RIGHT PLACES
5428 P(C)=P(C)/10
5450 IF C=3 THEN 5500
5490 GOTO 5010
5500 RETURN
6000 HOME:PRINT
6001 V5=V:B5=B
6010 PRINT "THIS SECTION WILL ALLOW
    YOU TO
6020 PRINT"CORRECT ANY DATA
6030 GOSUB 11000
6050 PRINT"1.  DATE ";E$
6060 PRINT"2.  ";A$
6070 PRINT"3.  ";M1$
6080 PRINT"4.  ";M2$
6100 PRINT"5.  PRICE ";PR
6110 PRINT"6.  EST LIFE FOR DEPRECIA
    TION ";L
6120 PRINT"7.  EST APPRECIATION/YEAR
    ";
6126 PRINT A
6130 PRINT"8.  CURRENT RENTS ";R
6140 PRINT"9.  ANTICIPATED RENTS ";A
    R
6160 PRINT"10. MONTHS OF OWNERSHIP T
    HIS YEAR ";M
6170 PRINT"11. EST TAXES + INSURANCE
    /MONTH ";T
6180 PRINT"12. EST EXPENSES/MONTH ";
    E
6190 PRINT"13. % LAND VALUE ";V5
6200 PRINT"14. % TAX BRACKET ";B5
6210 PRINT"15. DOWN PAYMENT  ";D9
6220 PRINT"16. % FURNISHINGS OF VALU
    E ";F
6225 GOSUB 11000
6300 PRINT"TO CHANGE, ANSWER THE NUM
    BER"
6310 INPUT "WHEN DONE, ANSWER -1";Q
6315 HOME:PRINT:PRINT
6320 IF Q=-1 THEN 6500
6330 ONQGOTO6350,6360,6365,6370,6390
    ,6400,6410,6420,6430,6440,
    6450,6460,6470,6480
6331 GOTO 6490
6350 INPUT "CORRECT DATE ";E$
6355 GOTO 6000
6360 INPUT "ADDRESS ";A$:GOTO 6000
6365 INPUT "MISC INFO ";M1$:GOTO6000

6370 INPUT "MISC INFO ";M2$:GOTO6000

6380 INPUT "ASKING/OFFERING PRICE ";
    PR
6382 IF PR<1 THEN 6380
6385 GOTO 6000
6390 INPUT "LIFE FOR DEPRECIATION ";
    L
6391 IF L<1 THEN 6390
6395 GOTO 6000
6400 INPUT "% APPRECIATION EXPECTED
    ";A
6405 GOTO 6000
6410 INPUT "CURRENT RENTS ";R:GOTO 6
    000
6420 INPUT "ANTICIPATED RENTS ";AR
6425 GOTO 6000
6430 INPUT "MONTHS OF OWNERSHIP THIS
     YEAR ";M
6432 IF M>12 THEN 6430
6434 IF M<0 THEN 6430
6436 GOTO 6000
6440 INPUT "EST TAXES + INSURANCE/MO
    NTH";T:GOTO6000
6450 INPUT "EST EXPENSES PER MONTH "
    ;E:GOTO 6000
6460 INPUT "PERCENT LAND VALUE ";V
6470 INPUT "TAX BRACKET ";B:GOTO 600
    0
6480 INVERSE:PRINT "REMEMBER - IF YO
    U CHANGE
6482 PRINT"YOUR DOWN PAYMENT, THE MO
    NTHLY PAYMENT SHOULD ALSO
    BE CHANGED":NORMAL:PRINT
```

```
6484 INPUT "DOWN PAYMENT";DP
6485 IF DP<1 THEN 6484
6486 IF DP>=PR THEN 6484
6487 GOTO 6000
6490 INPUT "FURNISHINGS % OF VALUE "
    ;F:GOTO 6000
6500 REM
6900 RETURN
7000 HOME:VTAB10:HTAB10:PRINT"----DO
    ING MATH----"
7002 V=100-V
7003 V=V/100
7004 B=B/100
7005 F=F/100
7010 R9=R*M :REM CURRENT RENTS THIS ~
    YEAR
7015 R9=INT(R9*10^2+.5)/100
7020 P(8)=P(9)*M: REM PAYMENTS THIS ~
    YEAR
7025 P(8)=INT(P(8)*10^2+.5)/100
7030 T9=M*T:REM TAXES Y-T-D THIS YEA
    R
7035 T9=INT(T9*10^2+.5)/100
7040 E9=E*M:REM EXPENSES Y-T-D THIS ~
    YEAR
7045 E9=INT(E9*10^2+.5)/100
7050 F1=R9-P(8)-T9-E9:REM F1=CASH FL
    OW THIS YEAR
7055 F1=INT(F1*10^2+.5)/100
7210 F2=(AR*12)-(P(9)*12)-(T*12)-(E*
    12)
7215 REM **F2=CASH FLOW 2ND YEAR
7220 F2=INT(F2*10^2+.5)/100
7300 REM FIGURE ASSET APPRECIATION
7310 A5=(PR*A)/100
7320 A5=A5/12
7330 REM A5=MONTHLY ASSET APPRECIATI
    ON
7340 A6=A5*12:REM FOR A FULL YEAR
7345 A5=A5*M:REM APPRECIATION FOR TH
    E 1ST YEAR
7350 A5=INT(A5*10^2+.5)/100
7360 A6=INT(A6*10^2+.5)/100
7400 REM FIGURE EQUITY BUILDUP
7410 REM P(8)=TOTAL PAYMENTS THIS YE
    AR
7420 REM T9=TAXES + INS THIS YEAR
7430 REM E9=EXPENSES TOTAL THIS YEAR

7440 REM F1=CASH FLOW 1ST YEAR
7450 REM F2=CASH FLOW 2ND YEAR
7500 EB=P(8)*E1(Y,I):REM EQUITY BUIL
    DUP 1ST YEAR
7505 EB=EB/100
7510 ET=(P(9)*12)*E2(Y,I):REM EQUITY
     BUILDUP 2ND YEAR
7515 ET=ET/100
8000 REM L IS PROPERTY VALUE
8010 REM V IS LAND VALUE %
8020 REM F=IS VALUE OF FURNISHINGS
8030 REM PR IS PRICE OF PROPERTY
8040 REM M IS MONTHS YOU OWN IT THIS
     YEAR
8050 REM B IS TAX BRACKET
8100 REM FIGURE 1ST YEAR TAX SAVINGS

8210 TS=PR*V:REM THIS IS THE NET PRO
    PERTY VALUE AFTER LAND IS ~
    DEDUCTED
8220 F5=PR*F:REM F4=VALUE OF THE FUR
    NISHINGS
8230 TS=TS-F5:REM TS IS NOW THE VALU
    E OF THE PROPERTY AFTER LA
    ND AND
8232 REM FURN ARE DEDUCTED
8240 TS=TS/L:REM THIS IS WHAT YOU CA
    N DEPRECIATE PER YEAR
8250 F5=F5/5:REM THIS IS THE AVERAGE
     DEP ON FURNISHINGS
8260 REM PART OVER 3 YEARS AND PART ~
    OVER 7=5 AVERAGE
8270 D5=F5+TS:REM THIS IS DEP FOR 1S
    T YEAR
8280 D6=D5
8290 D5=(D5/12)*M:REM THIS IS 1ST YE
    AR'S DEP, AND D6=2ND YEAR ~
    DEP
8300 TS=D5*B:REM THIS IS TAX SAVINGS
     1ST YEAR
8310 TT=D6*B:REM THIS IS TAX SAVINGS
     2ND YEAR
8400 REM RETURN ON INVESTMENT/EQUITY

8410 RO=F1+A5+EB+TS:REM THIS IS 1ST ~
    YEAR EQUITY TOTAL
8420 RE=F2+A6+ET+TT:REM THIS IS 2ND ~
    YEAR EQUITY BUILDUP
8430 RE=INT(RE*10^2+.5)/100
8440 RO=INT(RO*10^2+.5)/100
8500 TS=INT(TS*10^2+.5)/100
8510 F5=INT(F5*10^2+.5)/100
8520 D5=INT(D5*10^2+.5)/100
8530 TT=INT(TT*10^2+.5)/100
8540 D6=INT(D6*10^2+.5)/100
8550 EB=INT(EB*10^2+.5)/100
8560 ET=INT(ET*10^2+.5)/100
8900 RETURN
8999 V=20
9000 HOME:PRINT:INVERSE:PRINT TAB (1
    7)"INVEST                     ~
        ":NORMAL:GOSUB11000
9001 PRINT:PRINT"YOU WILL OWN THIS P
    ROPERTY ";M:PRINT"MONTHS T
    HIS YEAR. THE CASH FLOW":P
    RINT
9002 PRINT"IS BASED ON CURRENT RENTS
     THE 1ST"
9003 PRINT"YEAR OF $";R;"PER MONTH, ~
    AND OF ANTICIPATED RENTS F
    OR THE 2ND YEAR OF"
9004 PRINT"$";AR;"PER MONTH.":PRINT:
    PRINT"YOUR DOWN PAYMENT IS
```

SPECIALS on INTEGRATED CIRCUITS

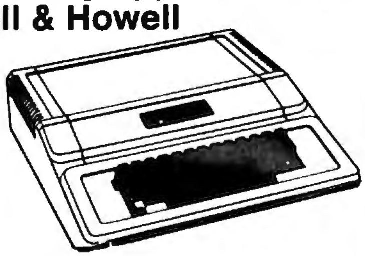
6502	7.45	10/6.95	50/6.55	100/6.15
6502A/6512A	8.40	10/7.95	50/7.35	100/6.90
6520 PIA	5.15	10/4.90	50/4.45	100/4.15
6522 VIA	6.45	10/6.10	50/5.75	100/5.45
6532	7.90	10/7.40	50/7.00	100/6.60
2114-L200		3.75	25/3.50	100/3.25
2114-L300		3.15	25/2.90	100/2.65
2716 EPROM		7.00	5/6.45	10/5.90
2532 EPROM				14.50
6116 Hitachi 2K x 8 CMOS RAM				14.50
4116				8 for 17
Zero Insertion Force 24 pin Socket				2.00
6550 RAM (PET 8K)				12.70
S-100 Wire Wrap Socket				2.40

A P Products 15% OFF
A P Hobby-Blox 15% OFF

THE STAR MODEM
From Prentice/Livermore Data Systems

RS232 MODEM	SALE	$128
IEEE 488 MODEM	SALE	$199
RS232 CCITT		$170
IEEE 488 CCITT		$270

We carry Apple II+ from Bell & Howell



fullFORTH+ for Apple II $75
A full-featured FORTH with enhancements. Conforms to FORTH Interest Group standards.

VIC 3K RAM (with 2 ROM Sockets) 65
Write for VIC Software List

REVERSAL (Spracklen) 32K Apple	28.00
Data Manager (Lutus) 24K Apple	40.00
Energy Miser - PET, Apple, Zenith	24.50
Histo-Graph (Boyd) 48K Apple	24.50
Data-Graph (Boyd) 48K Apple	40.00
Apple II User's Guide (Osborne)	12.00
Introduction to Pascal (Sybex)	10.30
Pascal Handbook (Sybex)	12.00
Musical Applications of Micros (Chamberlin)	20.00
Basic FORTRAN (Coan)	7.25
CP/M Handbook (with MP/M) Zaks	11.85
Programming the 6502 - Zaks	10.00
Microprocessor Interfacing Techniques	13.00
PET/CBM Personal Computer Guide (Osborne)	12.00
PET and the IEEE-488 Bus (GPIB)	12.25
6502 Assembly Language Prog. - Levanthal	13.50
Z-80 Assembly Language Prog. - Levanthal	12.75
8080A/8085 Assembly Language Programming	12.75
Z8000 Assembly Language Programming	16.00
6809 Assembly Language Programming	13.50
6800 Assembly Language Programming	12.75
The 68000 Microprocessor Handbook	5.75
BASIC and the Personal Computer (Dwyer)	11.00

commodore

CBM-PET SPECIALS

	list	SALE
8023 Printer - 136 col, 150 cps bi-directional	(995)	775
8300 (Diablo 630) Daisy Wheel - 40 cps bi-directional	(2250)	1725
8032 80 x 25 CRT, business keyboard	(1495)	1100
Micro Mainframe	(1995)	1600
8096 Board (extra 64K RAM for 8032)	(500)	400
8050 Dual Disk Drive - 1 megabyte	(1795)	1345
8250 Dual Disk Drive - 2 megabyte	(2195)	1760
CBM IEEE Modem	(395)	199
4016 full size graphics keyboard	(995)	795
4032 full size graphics keyboard	(1295)	999
4040 Dual Disk Drive - 330,000 bytes	(1295)	999
2031 Single Disk Drive - 165,000 bytes	(695)	560
4022 Tractor Feed Printer	(795)	630
C2N External Cassette Deck	(75)	65
VIC 20 Color Computer	(299)	259
VIC 1515 Graphic Printer	(395)	315
Used CBM/PET Computers		CALL
8024-7 High Speed Printer	(1995)	1345

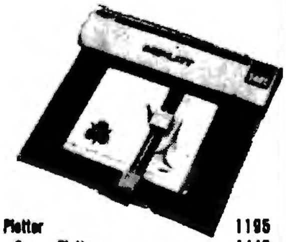
WE WILL MATCH ANY ADVERTISED PRICE

*** EDUCATIONAL DISCOUNTS ***
Buy 2 PET/CBM Computers, receive 1 FREE

WordPro 3 Plus - 32K CBM, disk, printer	200
WordPro 4 Plus - 8032, disk, printer	325
OZZ Data Base System for CBM 8032	335
VISICALC for PET, ATARI, or APPLE	155
SM-KIT - Super PET ROM Utilities	40
Programmers Toolkit - PET ROM Utilities	35
PET Spacemaker II ROM Switch	36
2 Meter PET to IEEE or IEEE to IEEE Cable	40
Dust Cover for PET	7
IEEE-Parallel Printer Interface for PET	110
IEEE-RS232 Printer Interface for PET	120
The PET Revealed	17
Library of PET Subroutines	17

4 PART HARMONY MUSIC SYSTEM for PET
The Visible Music Monitor, by Frank Levinson, allows you to easily enter, display, edit, and play 4 part harmony music. Includes whole notes thru 64ths (with dotted and triplets), tempo change, key signature, transpose, etc. The KL-4M unit includes D to A converter and amplifier ready to hook to your speaker.
KL-4M Music Board with VMM Program $60

Watanabe Intelligent Plotter



WATANABE WX4671 Plotter 1195
WATANABE WX4675 6-pen Plotter 1445

DISK SPECIALS

SCOTCH (3M) 5"	10/2.75	50/2.65	100/2.60
SCOTCH (3M) 8"	10/2.80	50/2.70	100/2.65
Verbatim 8" Double Dens.	10/3.45	50/3.35	100/3.20
Verbatim 5" Datalife	10/2.45	50/2.40	100/2.35
(add 1.00 for Verbatim 5" plastic storage box)			
BASF 5" soft	10/2.40	20/2.35	100/2.30
Wabash 5" in Plastic Box	10/2.70	50/2.60	100/2.50
Wabash 8" in Plastic Box	10/2.75	50/2.65	100/2.55

WE STOCK MAXELL DISKS
Diskette Storage Pages 10 for 3.95
Disk Library Cases 8" - 2.85 5" - 2.15
Disk Hub Rings 8" - 50 @ 7.50 5" - 50 @ 6.00

CASSETTES - AGFA PE-611 PREMIUM
High output, low noise, 5 screw housings.

C-10	10/.56	50/.50	100/.48
C-30	10/.73	50/.68	100/.66

All other lengths available. Write for price list.

SPECIALS

EPSON MX-80 Printer	
EPSON MX-80 F/T Printer	
EPSON MX-70 Printer	
EPSON MX-100 Printer	
Centronics 739 Printer with dot graphics	675
STARWRITER Daisy Wheel Printer	1445
Zenith ZVM-121 Green Phosphor Monitor	115
Amdek Color Monitor	355

ALL BOOK and SOFTWARE PRICES DISCOUNTED
OSBORNE/McGraw-Hill, HAYDEN, SYBEX etc.

Synertek Systems

SYM-1 Microcomputer SALE	199
SYM BAS-1 BASIC or RAE 1/2 Assembler	85
KTM-2/80 Synertek Video and Keyboard	349
KTM-3/80 Synertek Tubeless Terminal	385

ZENITH data systems

Z90-80 64 K	2170
Z90-82 64K, 1 double dens. drive	2395
Z89-0 48K	1950
Z89-1 48K, 1 drive	2150
Z67 10 Megabyte + Floppy Drive	4495
Z37 1.3 Megabyte Dual Floppy	1495
Z25 High Speed Printer	1195
Z19 Video Terminal (VT-52 compatible)	725
ZVM-121 Green Phosphor Monitor	115
All Zenith Software discounted	

ATARI SPECIALS

800 Computer	749	DOS 2	21
400 - 16K	329	Pilot	68
810 Disk Drive	449	Microsoft BASIC	68
825 Printer	629	Educ. Series	20% off
850 Interface	139	**MISSILE COMMAND**	32
822 Printer	359	**ASTEROIDS**	32
Paddle Pair	17	**STAR RAIDERS**	32
Joystick Pair	17	Space Invaders	17
16K RAM	85	Music Composer	45
Assembler/Editor	46	Chess	30
TeleLink	20	Super Breakout	30

Write for prices on other Atari items.

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.

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, Relocator, 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

SUPERSORT by James Strasma $35

Supersort 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

by John Fluharty **$30**

SuperGraphics provides machine language extensions 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 that 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.

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.

SuperGraphics commands include GRAPHIC, TEXT, RVS, SET, DRAW, FILL, PLOT, MOVE, PRINT, CSET, CMOVE, DISPLAY, PUT, SWAP, PAUSE, and SOUND.

Please specify machine type and ROM version, disk or tape.

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.

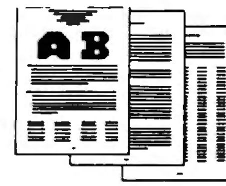
Runs on any 16K or 32K PET/CBM (including 8032) with ROM 3 or 4, and CBM disk drive. Please specify configuration when ordering.

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

Jukebox Series for PET by L C Cargile
Excellent 4 part harmony music--write for list
Automated Simulations (EPYX) Fantasy Games

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.

```
     $";DP;" YOU'RE IN THE ";
9005 PRINT100*B" % TAX BRACKET.":PRI
    NT"THE ESTIMATED USEFUL LI
    FE FOR":PRINT"DEPRECIATION

9006 PRINT" IS ";L;" YEARS.":PRINT "
    THE FIRST YEAR DEPRECIATIO
    N IS $";D5;" AND THE
9007 PRINT" SECOND YEAR DEPRECIATION
     IS $";D6;":?:GOSUB 11000:
    ?
9009 PRINT"HIT ANY KEY TO CONTINE...
    ";:GET L$:HOME:Q5=5
9013 PRINT" RETURN ON INVESTMENT:"
9014 GOSUB11000:INVERSE:PRINT TAB (Q
    5)"YEAR 1","YEAR 2":NORMAL

9015 GOSUB11000:PRINT"CASH FLOW:"
9016 Z9=F1:GOSUB15000
9017 PRINT TAB (Q5)Z9$,
9018 Z9=F2:GOSUB15000
9019 PRINTZ9$:PRINT
9020 PRINT"ASSET APPRECIATION:"
9021 Z9=A5:GOSUB15000
9022 PRINT TAB (Q5)Z9$,
9023 Z9=A6:GOSUB15000
9024 PRINTZ9$:PRINT
9030 PRINT"EQUITY BUILDUP:"
9032 Z9=EB:GOSUB15000
9034 PRINT TAB (Q5)Z9$,
9036 Z9=ET:GOSUB15000
9038 PRINTZ9$:PRINT
9040 PRINT"TAX SAVINGS:"
9042 Z9=TS:GOSUB15000
9044 PRINT TAB (Q5)Z9$,
9046 Z9=TT:GOSUB15000
9048 PRINTZ9$
9055 GOSUB11000
9060 PRINT"GROSS RETURN:"
9062 Z9=RO:GOSUB15000
9064 PRINT TAB (Q5)Z9$,
9066 Z9=RE:GOSUB15000
9068 PRINTZ9$
9069 GOSUB20000
9071 G6=RO/DP:G6=INT(G6*10^2+.5)/100

9072 G7=RE/DP:G7=INT(G7*10^2+.5)/100

9073 G6=G6*100:G7=G7*100
9074 PRINT"RETURN ON EQUITY %":PRINT
    TAB(Q5))G6;"%",G7;" %"
9075 GOSUB 20000:PRINT"HIT ANY KEY T
    O CONTINUE...";:GET L$
9079 GOTO 10000
9080 V=V*100
9082 V=100-V
9083 B=B*100
9084 F=F*100
9100 V=INT(V*10^2+.5)/100
9110 F=INT(F*10^2+.5)/100
9120 B=INT(B*10^2+.5)/100
9130 GOTO10106
10000 REM MENU
10010 HOME:PRINT
10015 PRINT:PRINT
10017 INVERSE:PRINT TAB (17)"INVEST  ~
                     ":NORMAL
    :PRINT
10020 PRINT"<1> TO SEE THE SAME DATA ~
    AGAIN"
10025 PRINT
10030 PRINT"<2> TO CHANGE OR PRINT TH
    E DATA"
10035 PRINT
10040 PRINT"<3> TO STOP NOW"
10050 PRINT
10060 INPUTQ
10070 IFQ=1THEN GOTO 9000
10080 IFQ=3THENPRINT"END OF";:INVERSE
    :PRINT"INVEST";:NORMAL:PRI
    NT"PROGRAM":END
10085 IFQ>3THEN10000
10086 IFQ<1THEN10000
10090 REM MENU
10100 HOME:PRINT
10105 GOTO9080
10106 PRINT
10108 PRINT:PRINT:INVERSE:PRINTTAB(17
    )"INVEST               ":N
    ORMAL:PRINT:PRINT
10110 PRINT"<1> CHANGE FINANCIAL DATA
    "

10115 PRINT
10120 PRINT"<2> CHANGE THE PAYMENT DA
    TA"
10122 PRINT
10125 PRINT"<3> PRINT THE DATA"
10126 PRINT
10127 PRINT"<4> STOP NOW"
10130 PRINT
10135 INPUTQ
10140 IFQ<1THEN10100
10142 ONQTOTO10150,10152,12000,10154
10150 GOSUB6000
10151 GOTO10155
10152 C=0:P(1)=0:P(2)=0:P(3)=0:GOSUB5
    000:REM C IS ZEROED TO RES
    TART COUNTER
10153 P(9)=0:GOTO2200:REM ZERO PAYMEN
    T AND THEN DO MATH TO ADD ~
    UP NEW PAYMENTS
10154 PRINT"END OF PROGRAM ":END
10155 GOSUB7000:REM MATH
10158 GOTO9000:REM PRINT
10160 GOTO10000
10165 GOSUB7000:REM MATH
10166 GOTO9000:REM PRINT
11000 PRINT"-------------------------
    -------"
11010 RETURN
12000 HOME:PRINT
12005 VTAB6
```

```
12010 FLASH:PRINT"TURN ON THE PRINTER
    ":NORMAL:PRINT
12015 PRINT:PRINT:GOSUB11000:PRINT:PR
    INT
12020 PRINT"ANSWER 1 TO CONTINUE..."
12022 PRINT"ANSWER 2 TO STOP......."
12024 PRINT:INPUTQ
12025 PRINT:PRINT
12026 IFQ=2THENPRINT"END OF ";:INVERS
    E:PRINT"INVEST;":NORMAL:PR
    INT"PROGRAM":END
12030 D$=CHR$(4)
12040 PRINTD$;"PR#1"
12050 PRINT""
12100 PRINT"        "
12110 FORCO=1TO200:NEXTCO
12130 PRINTTAB(5)E$
12135 PRINTTAB(5)"PROPERTY ANALYSIS R
    EPORT FOR ";A$
12140 PRINT"        "
12150 PRINTTAB(5)M1$:PRINTTAB(5)M2$
12170 PRINTTAB(5)"ASKING/OFFERING PRI
    CE ";
12171 Z9=PR:GOSUB15000:PRINTZ9$
12172 PRINT"        "
12179 PRINT"        "
12180 FORCO=1TO70:PRINT"*";:NEXTCO
12181 PRINT"        "
12190 PRINT TAB(5)"CASH FLOW ESTIMATE
    , BASED ON OWNING THIS PRO
    PERTY FOR";M;"MONTHS"
12200 PRINTTAB(5)"THE FIRST YEAR, 12
    MONTHS THE SECOND YEAR. FI
    RST YEAR CASH FLOW"
12210 PRINTTAB(5)"BASED ON CURRENT RE
    NTS OF ";R;"MONTHLY, AND T
    HE 2ND YEAR IS BASED"
12220 PRINTTAB(5)"ON ANTICIPATED RENT
    S OF ";AR;" PER MONT.  EST
    IMATED"
12222 PRINTTAB(5)"APPRECIATION IS ";A
    ;" %"
12230 GOSUB11000:PRINTTAB(5)"ALL FIGU
    RES ARE APPROXIMATE":GOSUB
    11000:PRINT"          "
12232 FORCO=1TO400:NEXTCO
12235 PRINTTAB(29)"1ST YEAR
       2ND YEAR"
12237 FORCO=1TO400:NEXTCO
12300 PRINTTAB(5)"MONTHLY RENTS
       ";
12310 Z9=R9:GOSUB15000
12320 Q9=LEN(Z9$)
12330 PRINTTAB(11-Q9)Z9$;
12340 Z9=AR*12:GOSUB15000
12350 Q9=LEN(Z9$)
12360 PRINTTAB(11-Q9)Z9$;
12370 PRINT"        "
12372 FORCO=1TO400:NEXTCO
12400 PRINTTAB(5)"MORTGAGE PAYMENTS
       ";
12410 Z9=P8:GOSUB15000
12420 Q9=LEN(Z9$)
12430 PRINTTAB(11-Q9)Z9$;
12440 Z9=P(9)*12:GOSUB15000
12450 Q8=LEN(Z9$)
12460 PRINTTAB(20-Q8)Z9$;
12470 FORCO=1TO400:NEXTCO
12500 PRINTTAB(5)"TAXES + INSURANCE
       ";
12510 Z9=T9:GOSUB15000
12520 Q9=LEN(Z9$)
12530 PRINTTAB(11-Q9)Z9$;
12540 Z9=T*12:GOSUB15000
12560 Q8=LEN(Z9$)
12570 PRINTTAB(20-Q8)Z9$;
12580 FORCO=1TO400:NEXTCO
12600 PRINTTAB(5)"MISC. EXPENSES
       ";
12610 Z9=E9:GOSUB15000
12620 Q9=LEN(Z9$)
12630 PRINTTAB(11-Q9)Z9$;
12640 Z9=E*12:GOSUB15000
12650 Q8=LEN(Z9$)
12660 PRINTTAB(20-Q8)Z9$;
12690 PRINT"        "
12695 FORCO=1TO400:NEXTCO
12700 PRINTTAB(5)"ESTIMATED CASH FLOW
       ";
12710 Z9=F1:GOSUB15000
12720 Q9=LEN(Z9$)
12730 PRINTTAB(11-Q9)Z9$;
12740 Z9=F2:GOSUB15000
12750 Q8=LEN(Z9$)
12760 PRINTTAB(20-Q8)Z9$;
12770 PRINT"        "
12780 FORCO=1TO400:NEXTCO
12785 PRINT"        "
12790 PRINT"        "
12795 FORCO=1TO400:NEXTCO
12800 PRINTTAB(5)"RETURN ON INVESTMEN
       T ANALYSIS"
12810 PRINT"        "
12815 FORCO=1TO400:NEXTCO
12820 PRINTTAB(40)"1ST YEAR";
12822 PRINTTAB(11)"2ND YEAR"
12830 PRINT"        "
12835 FORCO=1TO400:NEXTCO
12900 PRINTTAB(5)"CASH FLOW (FROM ABO
    VE)             ";
12910 Z9=F1:GOSUB15000
12920 Q9=LEN(Z9$)
12930 PRINTTAB(11-Q9)Z9$;
12940 Z9=F2:GOSUB15000
12950 Q8=LEN(Z9$)
12960 PRINTTAB(20-Q8)Z9$;
12970 FORCO=1TO400:NEXTCO
13000 PRINTTAB(5)"ASSET APPRECIATION
                   ";
13010 Z9=A5:GOSUB15000
13020 Q9=LEN(Z9$)
13030 PRINTTAB(11-Q9)Z9$;
```

```
13040 Z9=A6:GOSUB15000
13050 Q8=LEN(Z9$)
13060 PRINTTAB(20-Q8)Z9$;
13065 FORCO=1TO400:NEXTCO
13100 PRINTTAB(5)"EQUITY BUILDUP (APP
    ROXIMATE)        ";
13110 Z9=EB:GOSUB15000
13120 Q9=LEN(Z9$)
13130 PRINTTAB(11-Q9)Z9$;
13140 Z9=ET:GOSUB15000
13150 Q8=LEN(Z9$)
13160 PRINTTAB(20-Q8)Z9$;
13165 FORCO=1TO400:NEXTCO
13200 PRINT"                ":PRINTTAB(7)
    "YOUR ESTIMATED TAX SAVING
    S ARE"
13202 PRINTTAB(7)"BASED ON A TAX BRAC
    KET OF ";B5;" %"
13204 PRINTTAB(7)"AND A LIFE FOR DEPR
    ECIATION "
13213 PRINTTAB(7)"OF ";L" YEARS. DEPR
    ECIATION"
13215 PRINTTAB(7)"THE FIRST YEAR IS "
    ; INT (D5);" AND"
13217 PRINTTAB(7)"THE 2ND YEAR IS ";D
    6;" ."
13218 PRINTTAB(7)"THE FURNISHINGS ARE
     WORTH ";F
13219 PRINTTAB(7)"% OF THE PROPERTY C
    OST."
13224 PRINT"      ":FORCO=1TO400:NEXTC
    O
13225 PRINTTAB(5)"TAX SAVINGS
                          ";
13230 Z9=TS:GOSUB15000
13240 Q9=LEN(Z9$)
13250 PRINTTAB(20-Q9)Z9$;
13255 Z9=TT:GOSUB15000
13260 Q8=LEN(Z9$)
13270 PRINTTAB(20-Q8)Z9$;
13300 FORCO=1TO70:PRINT"-";:NEXTCO
13305 PRINT"      "
13310 PRINTTAB(5)"YOUR RETURN ON INVE
    STMENT IS      ";
13320 Z9=RO:GOSUB15000
13330 Q9=LEN(Z9$)
13340 PRINTTAB(11-Q9)Z9$;
13350 Z9=RE:GOSUB15000
13360 Q8=LEN(Z9$)
13370 PRINTTAB(20-Q8)Z9$;
13380 PRINT"      "
13395 FORCO=1TO400:NEXTCO
13400 PRINTTAB(5)"YOUR DOWN PAYMENT W
    AS $ ";DP
13405 FORCO=1TO400:NEXTCO
13410 PRINTTAB(5)"YOUR % RETURN ON IN
    VESTMENT IS    ";
13430 Z9=100* (RO/DP):GOSUB15000
13440 Q9=LEN(Z9$)
13450 PRINTTAB(12-Q9)Z9$;" %";
13460 Z9=100* (RE/DP):GOSUB15000
13470 Q8=LEN(Z9$)
13480 PRINTTAB(18-Q8)Z9$;" %";
14800 D$=CHR$(4)
14810 PRINTD$;"PR#0"
14999 PRINT"END OF ";:INVERSE:PRINT"I
    NVEST;":NORMAL:PRINT"PROGR
    AM":END
15000 REM PRINTUSING ROUTINE
15005 IFZ9<0THEN16000
15010 REM Z9= VARIABLE TO BE CHANGED
15020 IF(100*Z9-INT(100*Z9))<.5THENZ9
    =INT(100*Z9)/100:GOTO15030

15022 Z9=(INT(100*Z9)+1)/100
15024 REM MOVE ALPHANUMERIC TO STRING
     VARIABLE
15030 Z9$=STR$(Z9)
15035 REM ADD DOLLAR SIGN
15045 REM ADJUST DECIMAL IF REQUIRED
15050 Z9=LEN(Z9$):IFZ9<=2THEN15200
15055 Y9$=RIGHT$(Z9$,3)
15060 IFY9$<="$99"THEN15080
15070 IFY9$=<".99"THEN15220
15080 Y9$=RIGHT$(Z9$,2)
15090 IFY9$<=".9"THENZ9$=Z9$+"0":GOTO
    15210
15200 Z9$=Z9$+".00"
15205 REM NOW TO ADD A COMMA, IF REQU
    IRED
15210 Z9=LEN(Z9$)
15220 IFZ9<8THEN15400
15230 Y9$=RIGHT$(Z9$,6)
15240 Y9$=","+Y9$
15250 Y9$=LEFT$(Z9$,(Z9-6))+Y9$
15255 REM Z9$ IS THE EDITED FIELD
15260 79$=Y9$
15265 REM Z9 WILL CONTAIN THE LENGTH
    OF THE EDITED FIELD
15267 Z9=Z9+1
15400 RETURN
15752 GOSUB11000:PRINT
16000 Z9$=STR$(Z9)
16010 REM
16020 RETURN
17000 HOME:PRINT:INVERSE:PRINTTAB(17)
    "INVEST                ":NORM
    AL:PRINT
17010 PRINT"THIS IS A REAL ESTATE ANA
    LYSIS"
17020 PRINT"PROGRAM, WHICH WILL FIGUR
    E CASH FLOW,"
17030 PRINT"EQUITY BUILDUP, ASSET APP
    RECIATION"
17040 PRINT"AND TAX SAVINGS FOR AN IN
    COME PROPERTY."
17050 PRINT
17060 PRINT"IF THE LOANS YOU HAVE, OR
     ARE GETTING"
17070 PRINT"FOR A PARTICULAR PROPERTY
     ARE FOR"
17080 PRINT"A DIFFERENT TERM, OR AT A
```

```
     DIFFERENT"
17090 PRINT"RATE THAN WHAT THE PROGRA
17070 PRINT"FOR A PARTICULAR PROPERTY
     ARE FOR"
17080 PRINT"A DIFFERENT TERM, OR AT A
     DIFFERENT"
17090 PRINT"RATE THAN WHAT THE PROGRA
     M ASKS FOR,"
17100 PRINT"INPUT THE ANSWER AS CLOSE
     AS POSSIBLE."
17105 PRINT
17110 PRINT"FOR INSTANCE, YOU CAN USE
     AN INTEREST"
17120 PRINT"RATE FROM 10 TO 18% IN .5
     STEPS"
17130 PRINT"IF YOUR LOAN HAPPENS TO B
     E AT 11.75%,"
17140 PRINT"USE THE CLOSEST ANSWER--1
     1.5%"
17150 PRINT
17152 PRINT:GOSUB11000:PRINT
17160 PRINT"HIT ANY KEY TO CONTINUE..
     .";:GETL$
17190 HOME:PRINT
17200 PRINT"IF YOU MAKE AN ERROR IN A
     NSWERING,"
17210 PRINT"JUST CONTINUE, AS YOU'LL ~
     HAVE THE "
17220 PRINT"CHANCE TO CORRECT YOUR DA
     TA IN A"
17230 PRINT"MOMENT."
17240 PRINT
17250 PRINT"ALSO, ONCE YOU HAVE THE D
     ATA INTO"
17260 PRINT"THE COMPUTER, YOU'LL BE A
     LLOWED TO"
17270 PRINT"CHANGE IT, AS YOU WISH.  ~
     "
17280 PRINT
17290 PRINT"SO, YOU MIGHT WANT TO SEE
     THE RESULTS"
17300 PRINT"OF AN INVESTMENT WITH $10
     ,000 DOWN,"
17400 PRINT"AND SEE WHAT HAPPENS IF Y
     OU PUT"
17410 PRINT"$15,000 DOWN.  OR IF YOU ~
     TAX BRACKET"
17420 PRINT"WAS A BIT HIGHER, OR IF T
     HE PAYMENTS"
17430 PRINT"STRETCHED OUT A BIT LONGE
     R, AND SO ON."
17440 PRINT
17450 PRINT"WHEN YOU HAVE THE RESULTS
     YOU WANT,"
17460 PRINT"YOU CAN ASK FOR A PRINTOU
     T.  ONCE   "
17470 PRINT"YOU GET A PRINTOUT OF THE
     INFORMATION,"
17480 PRINT"THE PROGRAM WILL END. "
17482 PRINT
17490 INVERSE:PRINT TAB (17)"INVEST" ~
     ":NORMAL
17800 RETURN
20010 RETURN
21000 REM
21142 HOME:VTAB6
21143 HTAB 16
21144 SPEED=255
21145 INVERSE:PRINT "INVEST           ~
     ":NORMAL:PRINT:PRINT:PR
     INT
21150 NORMAL
21160 PRINT
21165 GOSUB11000:PRINT
21170 PRINT"......A REAL ESTATE ANALY
     SIS PROGRAM"
21175 PRINT
21180 GOSUB11000
21190 PRINT
21200 PRINT"......GREGORY R. GLAU"
21210 PRINT"      P.O. BOX 1627"
21220 PRINT"      PRESCOTT AZ 86302"
21250 PRINT:GOSUB11000:PRINT
21280 PRINT
21300 PRINT"       HIT ANY KEY TO CO
     NTINUE...";:GETL$
21900 SPEED=255
22000 NORMAL
22010 RETURN
```

Program 2.

```
1 OPEN #1,4,0,"K:"
2 DIM E$(20),A$(80),M1$(80),M2$(80),P(10
),Z9$(90),Y9$(80)
60 READ TEMP:Q(Y,I)=TEMP
220 READ TEMP:E1(Y,I)=TEMP
440 READ TEMP:E2(Y,I)=TEMP
2000 ? "HIT ANY KEY TO CONTINUE...";:GET
 #1,TEMP
2004 PRINT "{CLEAR}{DOWN}"
2005 ? "|                 INVEST
     |":? "PLEASE ANSWER THE FOLLOWING
..."
2008 PRINT "|ANSWER 'END' TO STOP NOW|{2
 DOWN}"
2010 PRINT "TODAY'S DATE ";:INPUT E$
2030 PRINT "PROPERTY ADDRESS ";:INPUT A$
2032 ? "{CLEAR}"
2040 ? :? "MISC INFO (1)";:INPUT M1$
2050 ? "MISC INFO (2)";:INPUT M2$
2060 ? :? :? "ASKING/OFFERING PRICE";PR
2070 ? "{CLEAR}"
2076 ? "CONSULT WITH YOUR ACCOUNTANT - |
PLEASE|"
2081 ? "ESTIMATED LIFE FOR DEPRECIATION
IN YEARS";:INPUT L:?
2085 ? "{CLEAR}"
2089 ? "ESTIMATED APPRECIATION PER YEAR"
;:INPUT A
```

```
2090 ? "{CLEAR}"
2091 ? "CURRENT TOTAL RENTS PER MONTH";:
INPUT R
2100 ? "ANTICIPATED TOTAL RENTS PER MONT
H";:INPUT AR
2115 ? "PROPERTY THIS YEAR";:INPUT M
2120 ? "TAXES + INSURANCE PER MONTH";:IN
PUT T
2140 ? "ESTIMATED EXPENSES PER MONTH";:I
NPUT E
2147 ? "{CLEAR}"
2150 ? "% TAX BRACKED YOU'RE IN (30%=30)
";:INPUT B
2162 ? :? "DOWN PAYMENT";:INPUT DP
2200 ? "{CLEAR}"
2290 ? "DO YOU WANT TO CHANGE THIS <1=YE
S>";:INPUT Q
2410 ? :? "CORRECT PAYMENT TOTAL";:INPUT
 TEMP:P(9)=TEMP
3032 ? "{CLEAR}"
5000 ? "{CLEAR}":? :? "NOW WE HAVE TO FI
GURE YOUR":? "MONTHLY PAYMENTS FOR THIS
PROPERTY."
5001 ? "YOU CAN INPUT UP TO 3 PAYMENTS."
:?
5005 ? :? "|YOU MUST INPUT SOMETHING|"
5006 ? "|--EVEN IF YOU CHANGE IT LATER O
N|"
5007 ?
5250 ? "YEARS (15-20-25)";:INPUT TEMP:Y(
C)=TEMP
5310 PRINT "PERCENT RATE";:INPUT Q
5400 ? "AMOUNT OF LOAN";:INPUT TEMP:A(C)
=TEMP
6000 ? "{CLEAR}"
6310 ? "WHEN DONE, ANSWER-1";:INPUT Q
6315 ? "{CLEAR}":?
6350 ? "CORRECT DATE";:INPUT E$
6360 ? "ADDRESS";:INPUT A$:GOTO 6000
6365 ? "MISC INFO ";:INPUT M1$:GOTO 6000

6370 ? "MISC INFO ";:INPUT M2$:GOTO 6000

6380 ? "ASKING/OFFERING PRICE";:INPUT PR
6390 ? "LIFE FOR DEPRECIATION";:INPUT L
6400 ? "% APPRECIATION EXPECTED";:INPUT
A
6410 ? "CURRENT RENTS";:INPUT R:GOTO 600
0
6420 ? "ANTICIPATED RENTS";:INPUT AR
6430 ? "MONTHS OF OWNERSHIP THIS YEAR";:
INPUT M
6440 ? "EST TAXES + INSURANCE/MONTH";:IN
PUT T:GOTO 6000
6450 ? "EST EXPENSES PER MONTH";:INPUT E
:GOTO 6000
6460 ? "% LAND VALUE";:INPUT V
6470 ? "TAX BRACKET";:INPUT B:GOTO 6000
6480 ? "|REMEMBER-IF YOU CHANGE|"
6482 ? "|YOUR MONTHLY PAYMENTS, THE MONT
HLY|"
6483 ? "|PAYMENTS SHOULD ALSO BE CHANGED
.|":?
6484 ? "DOWN PAYMENT";:INPUT DP
6490 ? "FURNISHINGS % OF VALUE";:INPUT F
:GOTO 6000
7000 ? "{CLEAR}":POSITION 10,10:? "----D
OING MATH----"
9000 ? "{CLEAR}":POSITION 17,1:? "|INVES
T|":GOSUB 11000:? :? "YOU WILL OWN THIS
PROPERTY ";M:? "MONTHS THIS YEAR. THE CA
SH FLOW"
9001 ? "IS BASED ON CURRENT RENTS THE 1S
T"
9009 ? "HIT ANY KEY TO CONTINUE...";:GET
 #1,TEMP:? "{CLEAR}";:Q5=5
9014 GOSUB 11000:POKE 85,Q5:? "|YEAR 1|"
,"|YEAR 2|"
9017 POKE 85,Q5:? Z9$,
9034 POKE 85,Q5:? Z9$,
9044 POKE 85,Q5:? Z9$,
9064 POKE 85,Q5:? Z9$,
9074 ? "RETURN ON EQUITY %":POKE 85,10:?
 G6;" %",G7;" %"
9075 GOSUB 20000:? "HIT ANY KEY TO CONTI
NUE...":GET #1,TEMP
10010 ? "{CLEAR}"
10017 ? "|                 INVEST
       |"
10080 IF Q=3 THEN ? "END OF |INVEST| PRO
GRAM":END
10100 ? "{CLEAR}"
10108 ? "{2 DOWN}|                 INVEST
                |{2 DOWN}"
12000 END
12026 IF Q=2 THEN ? "END OF |INVEST| PRO
GRAM":END
15055 Y9$=Z9$(LEN(Z9$)-3)
15080 Y9$=Z9$(LEN(Z9$)-2)
15090 IF Y9$<=".9" THEN Z9$(LEN(Z9$)+1)=
"0":GOTO 15210
15200 Z9$(LEN(Z9$)+1)=".00"
15210 Z9=LEN(Z9$)
15230 Y9$=Z9$(Z9-6)
15240 TEMP$=Y9$:Y9$=",":Y9$(2)=TEMP$
15250 TEMP$=Z9$(1,Z9-6):TEMP2$=Y9$:Y9$=T
EMP$:Y9$(LEN(Y9$)+1)=TEMP2$
17490 ? "|                 INVEST
       |"
21142 ? "{CLEAR}"
21143 POSITION 16,6
21144 REM
21145 ? "|INVEST|":? :? :? :?
21300 ? "   HIT ANY KEY TO CONTINUE...";
```

```
:GET #1,A
21900 REM
22000 REM
```

Mr. Glau has offered to make disk copies of the program for Apple owners: send him $3 and a disk in an SASE mailer. ©

Developing A Business Algorithm

Keith Falkner
Venice, FL

The heart of a computer program is its algorithm procedure. This is the case in this program. The purpose of the program is to solve a simple and fairly common problem in business: if a customer wishes to lease a durable article, with a view toward buying it at the end of the lease, what should the rental payment be? As written, this program limits the term to 6 or 12 or 24 or 36 months, and includes consideration of an annual charge for insurance. These considerations were part of a specific user's business environment.

The program uses an algorithm to calculate the lease payment and then verifies its result by simulating the passage of time and showing that the expected result actually happens. This will be illustrated in detail later. What is more important is how the algorithm was developed.

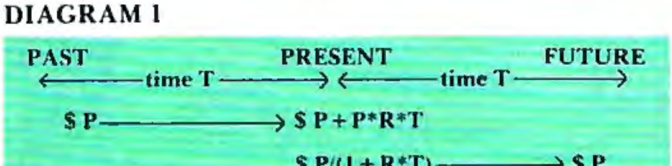
Creating An Interest Algorithm

Almost always, the idea behind an algorithm is very simple. This is certainly true here. The main idea is that interest is the product of principal, rate, and time. This is the simple formula which most of us have forgotten since high school.

Applying a simple formula can be a complex task, but is usually understandable in small pieces. For an example see Diagram 1, which merely illustrates that P dollars will grow to P + P * R * T dollars in T at rate R. This process can be treated in reverse: if money is to accumulate at interest in order to be worth P dollars at future time T at rate R, the present value of that money is P/(1 + R * T) dollars. These simple formulae are the heart of all interest calculations, however complicated they become.

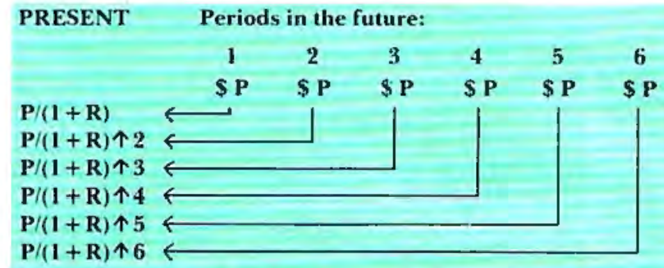
Diagram 2 shows the values of each of six payments of P dollars each, at intervals of unit time (that time which is the basis of the interest rate, e.g. 2% per month, unit time would be one month).

DIAGRAM 1



How money increases when the interest rate is R during time T.

DIAGRAM 2



Present value of six payments of $P with interest rate R.

The above is simple high school math. To add up the values of the six payments, we need another idea from high school. The sum of a geometrical progression of N terms, first A, ratio X:

$$S = A + A*X + A*X\uparrow 2 + A*X\uparrow 3 + \ldots + A*X\uparrow(N-1)$$
$$= A * \frac{X\uparrow N - 1}{X - 1}$$

By substituting P/(1 + R) ↑ 6 for A, and (1 + R) for X, we get:

$$S = \frac{P}{(1+R)\uparrow 6} * \frac{(1+R)\uparrow 6 - 1}{1 + R - 1}$$
$$= \frac{P}{R} * \left(1 - \frac{1}{(1+R)\uparrow 6}\right)$$

The value S above is the present value of what the customer will eventually pay in lease payments, six of them in this example. That money must equal the present value of the contract, which is the value of the article being leased, reduced by the value it will fetch after the lease is done, and increased by some fee for insurance.

Let's delve into the specific workings of the program. Table 1 identifies the variables used.

TABLE 1

Variable	Meaning
D	Fraction to buy it after lease
F	= 1 + R (for convenience)
I	Annual insurance premium factor
P	Payment each month of lease
Q	Optional price to buy after lease
R	Rate of return as % monthly
S	State sales tax rate
T	Number of months and payments
V	Value of the article being leased
W	Worth of contract (computed)
Z	Insurance factor (computed)

The program collects input values for I, R, S, T, and D; since I, R, and S will usually not change, the program knows standard values for these, which should be set to your standards, not those

actually shown in the listing.

Lines 140 to 170 calculate the insurance factor Z. For a six-month lease, Z is half the annual insurance factor I. For leases longer than a year, Z is I plus the present value of I for each future year of the lease.

Line 190 computes the total worth of the contract (the value of the article) plus the fee for insurance, minus the present value of the article's eventual selling price. That present value is expressed as D*V / F / F T. In plain English, that is the purchase-fraction D (for example .10 to buy at 10% of original price), times V the item's value, divided by F T to bring the future selling price into the present, further divided by F, so that the customer can buy the article, not on the day of the final payment, but a month later.

At last the payment P can be computed, since W (as calculated in line 190) is equal to the sum of the series of payments calculated above as S. The payment amount P is finally calculated in line 210, and is truncated to the last cent, *not* rounded to the nearest cent.

The loop in line 250 simulates the behaviour of the lease as time passes. Each month the indebtedness X is multiplied by F the interest factor, then a payment of P reduces that debt. Any debt remaining after all T payments have been made, represents the result of having ignored all the fractions of pennies which were dropped in line 230.

Well, you didn't think we were going to let the customer get away with fractions of pennies, did you? So the calculation in line 270 will show an amount slightly greater than the purchase-fraction D times the value V. Taxation laws may insist upon some minimum purchase fraction, and the above methods ensure that the final price will be at least D times V, and usually a few cents more.

The results of all this are promptly displayed on the screen. The value of the monthly payment is shown, and the eventual optional purchase price is shown, both before and after state sales tax.

Add this program to your bag of tricks, and you will have a new and potent way to attract investors. To verify that, just take a modest (nowadays) interest rate such as 2% per month, and calculate the investor's annual rate of return, which is (1 + R) 12 – 1. I leave to you the task of exploiting that algorithm.

Program 1.

```
100 REM LEASE CALCULATION
110 REM WITH OPTION TO BUY
120 REM
130 GOSUB550
140 REM CALC INSURANCE
150 IFT=6THENZ=I/2:GOTO180
160 Z=I:IFT>12THENZ=Z+I/F^12
170 IFT>24THENZ=Z+I/F^24
180 REM CALC WORTH OF LEASE
190 W=V+V*Z-D*V/F/F^T
200 REM CALC PAYMENT
210 P=R*W/(1-F^-T)
220 REM ROUND TO LAST CENT
230 P=.01*INT(P*100)
240 REM CALC FINAL PRICE
250 X=W:FORN=1TOT:X=X*F-P:NEXTN
260 REM BUY IT 1 MONTH LATER
270 Q=X*F+D*V
280 REM PRINT RESULTS
290 X=P:GOSUB430
300 PRINT:PRINT"MONTHLY PAYMENT IS ~
    ...";TAB(25);Z$
310 PRINT:PRINT"AFTER ";T;" PAYMENT
    S, THE"
320 X=Q:GOSUB430
330 PRINT"PRICE WILL BE ...";TAB(25
    );Z$
340 X=Q+Q*S:GOSUB430
350 PRINT"TAX INCLUDED, THAT'S ..."
    ;TAB(25);Z$
360 END
370 REM NUMERIC INPUT:
380 REM PRESET X$, XH, & XL
390 PRINTX$;:INPUT"";X
400 IFX>XHTHENPRINT"TOO HIGH!":GOTO
    390
410 IFX<XLTHENPRINT"TOO LOW!":GOTO3
    90
420 PRINT:RETURN
430 REM ROUND & FORMAT MONEY:
440 Z=.01*INT(X*100+.5)+.001
450 Z$=STR$(Z):Z$=LEFT$(Z$,LEN(Z$)-
    1)
460 Z$=RIGHT$("          $"+Z$,14)
470 RETURN
480 REM ANSWER YES-OR-NO
490 REM PRESET X$
500 PRINTX$;:INPUT"";Z$:PRINT
510 Z$=LEFT$(Z$,1)
520 IFZ$="Y"THENOK=1:RETURN
530 IFZ$="N"THENOK=0:RETURN
540 PRINT"PLEASE ANSWER 'Y' OR 'N'.
    ":GOTO500
550 REM INITIALIZATION
560 FORK=1TO24:PRINT:NEXT
570 PRINTTAB(12)"LEASE WITH OPTION ~
    TO BUY."
580 PRINTTAB(12)"BY: KEITH FALKNER ~
    - 1981."
590 PRINT:PRINT:PRINT
600 X$="SKIP INSTRUCTIONS? ":GOSUB4
    80
610 IFOKGOTO750
620 PRINT:PRINT"YOU ARE LEASING AN ~
```

NEW FROM CYBERIA ★ NEW FROM CYBERIA ★ NEW FROM CYBERIA

Mari

A comprehensive accounts receivable and insurance billing system for modern health care offices and clinics

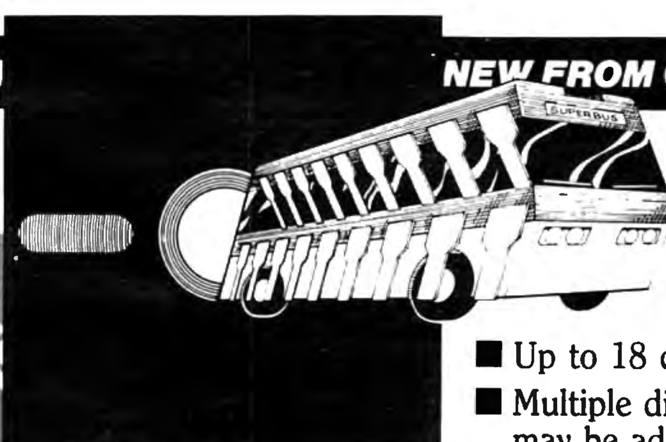
DESIGNED FOR COMMODORE 8000 SERIES COMPUTERS AND DISK DRIVES

- Help functions are always on-line
- Supports CPT, ICD and RVS medical coding
- Supports open item or balance forward accounting systems
- Interfaces to popular word processing programs
- Multiple terminals may be added with Superbus 4.0
- Multiple disk drives may be used—no limit on number of patients or accounts
- Includes a data base and forms generator to fill out any insurance form
- Includes a computer aided instruction program to train new users

Includes these standard reports:

- A/R aging
- A/R transactions
- General ledger
- Instant cash receipt
- Income analysis by physician
- Doctor referral report
- Patient/account cross reference
- Standard SuperBill insurance form

NEW FROM CYBERIA ★ NEW FROM CYBERIA ★ NEW FROM CYBERIA



SUPERBUS 4.0

Commodore Computer Networking System

- Up to 18 computers can be interconnected
- Multiple disk drives, printers and other devices may be added to the network
- Sophisticated security system prevents unauthorized use
- WordPro and Wordcraft programs are supported to allow multiple terminal word processing

*WordPro is a trademark of Professional Software, Inc.

CYBERIA INC. 2330 LINCOLN WAY, AMES, IA 50010 • 515-292-7634

```
     ITEM FOR A TERM OF"
630 PRINT"6, 12, 24, OR 36 MONTHS. ~
     AFTER THAT,"
640 PRINT"THE CUSTOMER CAN BUY THE ~
     ITEM FOR SOME"
650 PRINT"FRACTION OF ITS PRICE.":P
     RINT
660 PRINT"I CALCULATE THE MONTHLY P
     AYMENT."
670 PRINT:PRINT"I NEED TO KNOW SOME
     THINGS:":PRINT
680 PRINT"THE VALUE OF THE ITEM."
690 PRINT"THE COST OF INSURANCE."
700 PRINT"THE MONTHLY INTEREST RATE
     ."
710 PRINT"THE LENGTH OF THE TERM."
720 PRINT"THE LOCAL SALES TAX RATE.
     "
730 PRINT"THE PURCHASE FRACTION."
740 PRINT
750 X$="STANDARD SET-UP? ":GOSUB480
760 REM HERE IS THE STANDARD SETUP:
770 I=.02:REM 2% INSURANCE
780 R=.025:REM 2.5% / MONTH
790 S=.04:REM 4% FLORIDA TAX
800 IFOKGOTO900
810 PRINT"WHAT FRACTION OF THE ITEM
     'S VALUE IS"
820 PRINT"CHARGED EACH YEAR FOR INS
     URANCE?"
830 XL=0:XH=.2:X$="INSURANCE: ":GOS
     UB370:I=X
840 PRINT"WHAT IS THE MONTHLY INTER
     EST RATE?"
850 PRINT"(EXAMPLE: ENTER 3% AS .03
     )"
860 XL=.001:XH=.05:X$="INTEREST:  "
     :GOSUB370:R=X
870 PRINT"WHAT IS THE SALES TAX PER
     CENT?"
880 PRINT"(EXAMPLE: ENTER 8% AS .08
     )"
890 XL=0:XH=.3:X$="SALES TAX: ":GOS
     UB370:S=X
900 PRINT:PRINT"WHAT IS THE ITEM'S ~
     VALUE?"
910 XL=50:XH=50000:X$="VALUE      $"
     :GOSUB370:V=X
920 PRINT:PRINT"HOW MANY MONTHS? (6
      OR 12 OR 24 OR 36)"
930 XL=6:XH=36:X$="MONTHS:    ":GOSU
     B370:T=X
940 IFT=6ORT=12ORT=24ORT=36GOTO960
950 PRINT"I CAN'T HANDLE THAT!":GOT
     O920
960 PRINT"WHAT FRACTION OF THE ORIG
     INAL PRICE"
970 PRINT"WILL BUY THE ITEM AFTER T
     HE LEASE?"
980 XL=0:XH=.75:X$="FRACTION: ":GOS
     UB370:D=X
990 F=1+R:PRINT:PRINT"OK, HERE WE G
     O!":PRINT:RETURN
```

Program 2: Atari Version

```
105 DIM Z$(20),T$(20),X$(30)
300 PRINT :PRINT "MONTHLY PAYMENT IS ...
";:POKE 85,25:PRINT Z$
330 PRINT "PRICE WILL BE ...";:POKE 85,2
5:PRINT Z$
350 PRINT "TAX INCLUDED, THAT'S ...";:PO
KE 85,25:PRINT Z$
390 PRINT X$;:INPUT X
450 Z$=STR$(Z):IF LEN(Z$)>1 THEN Z$=Z$(1
,LEN(Z$)-1)
460 T$=Z$:Z$="          $":Z$(LEN(Z$)+1)=
T$:Z$=Z$(1,14)
500 PRINT X$;:INPUT X
510 Z$=Z$(1,1)
570 POKE 85,12:PRINT "LEASE WITH OPTION
TO BUY."
580 POKE 85,12:PRINT "BY: KEITH FALKNER
- 1981"
```

©

Anti-Hesitation Programming:
A Tutorial On Arrays

M. R. Smith
Calgary, Alberta, Canada

Editor's Note: The delays discussed and corrected in this article are a problem common to Microsoft BASICs (Apple, PET/CBM, OSI, etc.). Because the Atari has a different variable storage format, no hesitation is observed using the structure of Program 1. Atari BASIC, though, is similar to Microsoft with respect to GOTO — it searches the program for the target from top to bottom. And the time-saving effect of relocating REMs can be seen on the Atari. — RTM

Have you ever had a series of hesitations or pauses occuring at the start of your BASIC program? It is particularly obvious when using loops or subroutines. First time into a FOR...NEXT loop, the program seems to hiccup and pause. Thoughts of the dreaded *infinite loop* occur, but then the program seems to recover. The second time into the loop, the response is so fast that the screen smokes. What causes this alteration in behaviour?

To demonstrate the effect, enter and run Program 1:

```
 1 REM PROGRAM #1
20 PRINT "LINE 20" : DIM A(500), B(500), C(500)
30 PRINT "LINE 30"
40 FOR H =1 TO 5: I =1
50 J = 1 : K = 1: PRINT "LINE 50"
60 NEXT I
70 FOR I = 1 TO 5: PRINT "LINE 70"
80 L = 1 : M = 1: P = 1 : PRINT "LINE 80"
90 NEXT I : STOP
```

You'll notice a pause between line 20 and line 30. More pauses occur before lines 50 and 80. However, the next four times that the program gets to these lines, there is no pause.

On adding just one statement, line 10, to this program, you'll notice a real difference.

```
 1 REM PROGRAM #2
10 H = 0 : I = 0 : J = 0 : K = 0 : REM INITIALIZE
   VARIABLES
20 PRINT "LINE 20" : DIM A(500), B(500), C(500)
30 PRINT "LINE 30"
40 FOR H = 1 TO 5 : I = 1
50 J = 1 : K = 1 : PRINT "LINE 50"
60 NEXT I
70 FOR I = 1 TO 5 : PRINT "LINE 70"
80 L = 1 : M = 1 : P = 1 : PRINT "LINE 80"
90 NEXT I : STOP
```

In this version, the pause before line 50 has disappeared. This change occurs because the simple variables, H, I, J and K, are names in line 10 of the program. This means that these variables are used before any of the arrays, A(500), B(500), C(500) are made.

To explain why all this is occurs, you have to understand how a BASIC interpreter stores things in the computer memory. In the middle of a program (say line 90 of Program 1), memory is split up like this:

```
---------- BOTTOM
PROGRAM
----------
SIMPLE VARIABLES
----------
ARRAYS
----------
UNUSED
----------
CHARACTER ARRAYS
----------TOP
```

For each variable, array or string used in the program, there is a definite place reserved in memory.

Before we ran the program, things looked a lot simpler.

```
---------- BOTTOM
PROGRAM
----------
UNUSED
----------TOP
```

After line 20 in Program 1, things were different yet again.

```
---------- BOTTOM
PROGRAM
----------
A(500)
B(500)            ARRAYS
C(500)
----------
UNUSED
---------- TOP
```

The first pause in the program, before line 30, occurred while the arrays were being set up. The second pause occurred when the variables H and I were used for the first time. After line 40, the memory allocation was like this.

```
---------- BOTTOM
PROGRAM
----------
H
I            SIMPLE VARIABLES
----------
A(500)
B(500)       ARRAYS
```

INTRODUCING MTU-BASIC

MICROSOFT BASIC + USER ORIENTED ENHANCEMENTS = MTU-BASIC

CAN YOU

- Save and load BASIC programs in either memory image or ASCII format?
- Input COMMANDS and data to BASIC from a disk file as well as from the keyboard, i.e. drive BASIC from an ASCII "job" file on disk?
- Execute ANY Disk Operating System command from a BASIC program?
- Redefine the effect of keyboard function keys and display legends on the CRT to indicate their present function?
- Use a lightpen to input actual X, Y coordinates on a 480 x 256 pixel array in 1/60 second?
- Obtain very precise coordinate input using a moveable crosshair positioned by the cursor keys?
- Plot high resolution images using screen coordinates or floating point coordinates with the necessary transformations and image clipping accomplished automatically?
- Easily extend BASIC's command set with your own application oriented machine language routine library (up to 8 at once)?

MTU-BASIC CAN DO all of the above yet is based on the industry standard, Microsoft BASIC. If you are missing even one of the above functions, you should find out how an MTU-130 computer can make your association with BASIC a lot more pleasant and better suited to your special needs.

The MTU-130 also comes with other standard features that most computers offer only as options at extra cost — such things as 19.6K Bytes/sec sustained disk data transfer rate, digitized speech playback, 4 voice music synthesis, 480 x 256 bit mapped CRT screen display, fiber optic lightpen, RS-232 port, two parallel ports, hardware for cassette input and output, interface for local network, 80K RAM, 18 bit address bus, 8 bit audio DAC with 1 watt amplifier and a 3" x 5" speaker.

Shouldn't you be using MTU-BASIC on an MTU-130 Computer?

EXAMPLES FROM MTU-BASIC

ENTER "TRANSFER3"
Reads in an ASCII text file as program statements.

SYSTEM "ASSIGN 1 BASICIN"
Redirects input from keyboard to disk file named BASICIN.

LEGEND 1, "First," "Second"
Relegends function keys 1 and 2 to read "First" and "Second".

LTPEN F, X, Y
Sets F = 1 and X, Y to coordinates when lightpen picks a point.

GRIN NW$, X, Y
Displays crosshair and inputs X, Y location of its final position; NW$ contains the exit key.

DRAW .0645, 3*Y
Draw a vector from current location of graphic cursor to specified coordinates.

LIB "VGL," "IGL"
Select library extensions to be linked to BASIC.

The base MTU-130-1S system comes with one single-sided double-density 8" floppy disk, a 12" green phosphor CRT, and MTU-BASIC for $3995. Three other models priced up to $4995 contain 1 or 2 single or double sided drives for up to 2 Megabytes of storage. 4 Megabyte systems available on request.

We obviously cannot describe fully all of the details of the MTU-130 here. If you wish to know more about this complete desktop computer, call or write for our comprehensive 15 page descriptive literature. International requests include $5.00 U.S.

COME TO MTU — for excellence in microcomputing systems.

MTU
Micro Technology Unlimited
P.O. Box 12106
2806 Hillsborough St.
Raleigh, NC USA 27605
(919) 833-1458

```
C(500)
----------
UNUSED
---------- TOP
```

To make room for the variable H, the BASIC interpreter had to first move the arrays A( ), B( ) and C( ) higher up in memory. Then it had to move these arrays again to find room for variable I. During line 50, the arrays needed to be moved twice more; first for variable J and then to place variable K. All this movement caused the second pause. The more there is to move and the more variables there to place, the longer the pause will be.

The second time around in the FOR...NEXT loop, places for the variables H to J were already available in memory, so no more pauses occurred. The pauses, however, started again when the arrays had to be moved three times to provide room for the variables L, M and P in line 80.

In BASIC, each time a simple variable is used for the first time, all the arrays then in existence have to be moved up in memory. This causes a pause in the execution of the program. If a large number of variables is introduced, these pauses can accumulate into a sizeable delay. To avoid the pauses, we have to initialize (that means establish) all simple variables before we introduce any arrays.

To understand how this improves things, consider the memory after line 20 in Program 2. It looked like this:

```
---------- BOTTOM
PROGRAM
----------
H
I          SIMPLE VARIABLES
J
K
----------
A(500)
B(500)     ARRAYS
C(500)
----------
UNUSED
---------- TOP
```

This is very different to the appearance of the memory after line 20 of Program 1. When the program reaches line 40, the variables H to J will have already been fitted into memory, so that the arrays will not need to be moved. Therefore the pauses will vanish. At line 80, new variables will again have to be placed in memory, which means a pause while all the arrays move over. You can see the advantage of predefining all the simple variables before the arrays.

Systematic Initialization

Taking a systematic approach to the initialization of variables in a program can prevent a lot of problems. Program 2, rewritten for systematic initialization, might look something like this:

```
1 REM PROGRAM #2 NEW
10 GOSUB 60000 : REM DO INITIALIZATION
20 PRINT "LINE 20
30 PRINT "LINE 30
40 FOR H = 1 TO 5 : I = 1
50 = 1 : K = 1 : PRINT "LINE 50"
60 NEXT I
70 FOR I = 1 : PRINT "LINE 50"
60 NEXT I
70 FOR I = 1 TO 5 : PRINT "LINE 70"
80 L = 1 : M = 1 : P = 1 : PRINT "LINE 80"
90 NEXT I : STOP
59990 REM
60000 REM INITIALIZE SIMPLE VARIABLES
60010                        REM VARIABLES A - E
60020 H = 0 : I = 0 : J = 0 : REM VARIABLES F - J
60030 K = 0 : L = 0 : M = 0 : REM VARIABLES K - O
60040 P = 0                 : REM VARIABLES P - T
60050                         REM VARIABLES U - Z
60100 REM INITIALIZE ARRAYS
60110 DIM A(500), B(500), C(500)
60200 RETURN
```

This does seem to overdo things for such a short program, but this approach does have advantages for long programs.

1) Use a subroutine for initialization.

There is an obscure advantage of doing initialization using a subroutine. You could put equivalent statements to 60000 - 60200 at the beginning of a program. The advantage lies in the way that the BASIC interpreter handles GOSUB and GOTO commands. When a GOSUB command occurs, most BASIC interpreters skip to the beginning of the program. They then look at every line number (including those of REM statements) trying to find the line number wanted. Suppose that statements which are used only once in a program are placed at its start. There would be a tremendous waste of time while the interpreter unsuccessfully looks at these lines each time it searches for the line number it wants. Placing these lines at the end of the program makes for a great and simple way of speeding up your programs. This is particularly true when a GOTO command is issued from the middle of a FOR...NEXT loop near the end of the program.

The effect can be demonstrated by using the following program.

```
1 REM PROGRAM #3
10 PRINT "LINE 10"
20 REM
30 REM
.......keep inserting statements until you have about
  40 REM's
430 REM
440 FOR J = 1 TO 2500
450 GOTO 470
460 GOTO 480
```

```
470 GOTO 460
480 NEXT J
490 PRINT "490"
```

This is a short timing loop involving three interlinked GOTO statements. Measure the time it takes for the program to go between the two PRINT statements using the second hand of your watch. Now remove the REM statements and place them at the end of your program. Time again and notice the difference.

On my APPLE, the timing was 28 seconds with the REM's at the beginning compared to eight seconds with the REM's at the end. Quite a time saving. Shifting the initialization statements of your program can have the same effect. This also works the other way. If you have a subroutine that you use often, then place that at the beginning of the program. That way the BASIC interpreter can find it quickly.

2) List the variables in groups.

The main advantage of grouping the variables (A to E, F to I, etc.) on separate lines is that it becomes easy to determine if a variable has already been used.

It is not as obvious as you might think to determine whether or not a variable has already been used in a program. Consider a long program which uses variable YES at its beginning, and variable YEAR near its end. Many BASIC interpreters consider (since these two variables have the same two starting letters) that they must both be equal to the variable YE. This means that, although you intended the two variables to be different, they are actually being treated as the same game by the interpreter. Spotting a conflict like this can absorb a lot of time. However, if you put all variables in one location, then you are more likely to spot possible conflicts in names.

Declaring (initializing) all the variables at the beginning of a program can decrease the number of strange pauses in the middle of a program. It also decreases the chance of accidentally getting two independent variables with the same name. ©

How Random Are Sequences Of Random Numbers?

Brian J. Flynn
Vienna, VA

Chance is a word void of sense; nothing can exist without a cause. — Voltaire

---

Editor's Note: RND is one of the more intriguing aspects of computers: how do you generate accidents in a world dedicated to logic? Though Mr. Flynn analyzes the TRS-80 RND here, his approach and methods are applicable to any RND analysis. — RM

You turn on a Model I TRS-80 and key in "PRINT RND (0)." The response is ".768709." You key in the same command. The response this time is ".781397." You do this again, and again, and again. Using a FOR NEXT loop, suppose you generate a "random" sequence of 1,000 numbers. Or perhaps you generate 10,000. Or maybe even 100,000. Have you ever wondered how such "random" sequences of numbers are?

Before performing a statistical experiment a short while ago, I wanted to make sure the TRS-80's random number generator was a good one. So I examined its degree of randomness using a few popular statistical tests and a few common sense indicators.

But before discussing these, first note that the phrase "a random number" is used popularly to denote a member of a "random sequence" of figures. Strictly speaking, however, the adjective "random" should modify only "sequence," unless we happen to be concerned with the digits which comprise a number. This is because 0.768709 is not any more or any less random than 0.5 or 0.372 or any other positive fraction. Each occurs with zero probability in the selection of one number from the infinitely dense continuum of fractions from 0 to 1.

Executions of RND (0) on the TRS-80 generate rational numbers between 0 and 1, inclusive. "Rational," in this case, does not mean *sensibility*, but rather means that the fraction is expressible as a ratio of two integers. For instance, 0.625 is equivalent to 5/8. And the "ratio-nal" number 0.768709, from above, equals 768709 divided by one million. Fractions generated by RND (0) are supposed to be distributed in roughly uniform fashion as in Figure 1. Almost as many fractions should fall between 0 and 0.1 as between 0.1 and 0.2, and so on.

How close to uniformity are distributions of TRS-80 fractions? From machine-off to machine-on position, 100,000 executions of RND (0) generate the spread shown in Table 1. Non-TRS-80 owners may want to use the BASIC program listed here to see how well the random number generators on their machines compare.

The distribution in Table 1 is highly, but not perfectly, uniform. Less than perfect uniformity, however, is desirable. For if exactly 10,000 figures fell into each category, then the mechanism that generated this spread would seem awfully mechanical, too good to be true. While a good random number generator may father a perfectly uniform distribution, the probability of this is very low.

Just how close to uniformity should the distribution of fractions be? The chi (pronounced "ki") square goodness-of-fit statistic provides an answer:

$$X^2=\sum_{i=1}^{k}\frac{(0_i-E_i)^2}{E_i}$$

is the Greek capital letter sigma, for sum; k is the number of categories, also called cells or intervals; 0_i is the number of fractions observed in the i^{th} interval; and E_i is the number expected. In our case, $K=10$ and $E_i=100{,}000/10 = 10{,}000$.

Let's reveal the mystery of the formula. First, 0_i-E_i is the deviation of the expected from the actual number of observations for category "i." Next, this deviation is squared because $(0_i-E_i)=0$. Finally, the squared deviation is divided by E_i to give equal importance to each category in cases where the E_i's are different from one another. To clarify this last point, let $E_1=500$ and $E_2=1{,}000$ for the two-interval case. If 0_1 and 0_2 are 10% higher than E_1 and E_2, respectively, then $(0_1-E_1)^2 = (550-500)^2 = 2{,}500$ and $(0_2-E_2)^2 = (1{,}100-1{,}000)^2 = 10{,}000$. The second squared deviation is four times the first. Now, weighting each squared deviation relative to expected number, $(0_1-E_1)^2/E_1 = 2{,}500/500=5$ and $(0_2-E_2)^2/E_2 = 10{,}000/1{,}000 = 10$. The second term is now only twice as large as the first, just as E_2 is twice as large as E_1.

$X^2=12.07$ for 100,000 executions of RND (0), grouped into 10 cells. As Figure 2 shows, 10% of all values in a chi-square distribution with nine degrees of freedom (number of cells minus one) are less than 4.2 and 10% are greater than 14.7. Our value does not fall within either of these extreme percentiles. The sequence of fractions cannot, therefore, be accused of non-randomness on the basis of this test alone.

One test, however, is not conclusive evidence of randomness. The X^2 test performed on 100,000 numbers may suggest global randomness while hiding locally non-random behavior. For example, the distribution of the first 500,000 numbers generated by RND (0) may be skewed towards 0 while the distribution of the second 50,000 is skewed towards 1. The two distributions added together may appear uniform. To uncover such deception, the X^2 test is performed on each successive block of 10,000 fractions, and on each cumulative block. Table 2 shows that the TRS-80 random number generator produces an acceptable X^2 value in each case examined.

Batteries of statistical tests such as the chi-square will never prove that a random number generator is a good one, however. But they may diminish doubt, for each passed test boosts confidence in the quality of the generator. To strengthen or shatter this faith, sequences of TRS-80 fractions are now "RUNS" tested.

Let's explain this procedure using a list of Presidents of the United States and their political parties. We start with Franklin Pierce to avoid the Whigs and Federalists before him.

PRESIDENT		PRESIDENT	
Franklin Pierce	D	Woodrow Wilson	D
James Buchanan	D	Warren G. Harding	R
Abraham Lincoln	R	Calvin Coolidge	R
Andrew Johnson	R	Herbert Hoover	R
Ulysses S. Grant	R	Franklin D. Roosevelt	D
Rutherford B. Hayes	R	Harry S. Truman	D
James A. Garfield	R	Dwight D. Eisenhower	R
Chester Alan Arthur	R	John F. Kennedy	D
Grover Cleveland	D	Lyndon B. Johnson	D
Benjamin Harrison	R	Richard M. Nixon	R
Grover Cleveland	D	Gerald R. Ford	R
William McKinley	R	Jimmy Carter	D
Theodore Roosevelt	R	Ronald Reagan	R
William Howard Taft	R		

Are Democrats (D) and Republicans (R) randomly distributed here? Notice the string of six Republicans from Lindoln to Arthur. And notice that Grover Cleveland appears twice! Let's compare your guess to the probabilistic answer of the Runs Test. We first count the number of runs of Democrats or Republicans:

DD	RRRRRR	D	R	D	RRR	D	RRR
1	2	3	4	5	6	7	8

DD	R	DD	RR	D	R
9	10	11	12	13	14

A run is a succession of identical symbols followed and preceded by the opposite symbol, or by no symbol at all. There are 14 runs in our sequence. The essence of the Runs Test is to determine if this number is "too many," or "too few," or "about right." "Too many" runs is best exemplified by a sequence where Democrats and Republicans perpetually alternate: D R D R D R ... and so on. It is highly unlikely that a random sequence will follow a pattern so mechanical. "Too few" runs, on the other hand, is exemplified in its most grievous form by a sequence of all Democrats or all Republicans: R R R R R R ... and so on. Again, it is highly unlikely that a random sequence will display this. The Runs Test formula (reference 2) is:

$$z = \frac{\left| \frac{2n_1n_2}{N} - R \right| + c}{\sqrt{\frac{2n_1n_2}{N} \times \frac{2n_1n_2 - N}{N^2 - N}}}, \text{ where}$$

n_1 = the number of Democrats
n_2 = the number of Republicans
$N = n_1 + n_2$
R = the number of runs

This leaves "c," which is Yates' factor to make z's distribution better approximate a normal curve. Specifically,

$$c = +0.5 \text{ if } R < 2n_1n_2/N \text{ and } c = -1.5 \text{ if } R > 2n_1n_2/N$$

Actually, the z-formula is supposed to be used only if n_1 and/or n_2 is more than 20; a special table is used otherwise. For our example, however, the table and the formula give the same result. We march with z to demonstrate its use.

In calculating z, first note that $2n_1n_2/N = 2*10*17/27 = 12.5926$. With $R = 14$, $c = -1.5$. Therefore:

$$z = \frac{\left| \frac{2*10*17}{27} - 14 \right| - 1.5}{\sqrt{\frac{2*10*17}{27} * \frac{2*10*17-27}{27*27-27}}} = -0.04.$$

We reject with 95% confidence the assumption that a sequence is random whenever $z = 1.96$ or more. Since our calculated value is less than this, the Runs Test won't allow us to call the sequence of political parties non-random.

To "Runs" test a sequence of fractions, replace the "D's" and "R's" with "+'s" and "–'s." A "–" denotes a fraction below the expected median, 0.5, and a "+" denotes a fraction above it. For example, [.3 .7 .1 .2 .6] becomes [– + – – +]. Executing the Runs Test on 100,000 TRS-80 fractions, and on blocks therein, gives the results shown in Table 2. Each sequence appear random.

The computer program also generates four

DTL BASIC COMPILER

- Compatible with all existing Basic interpreter commands
- Accepts extensions to Basic implemented in RAM or ROM
- Provides demanding two pass syntax and logic analysis
- Improved Performance based on faster execution times
- Large Programs (16K+) will benefit from memory savings
- Security Key attaches to either cassette port

A Basic COMPILER for your Commodore Microcomputer by Drive Technology

DTL-BASIC is a Basic compiler for Commodore machines designed to convert existing programs to machine code and run them without modification. Compiled programs will run much faster and operate in exactly the same way as the un-compiled versions. Compiled code is typically 20 to 50% smaller than source code. For large programs this saving will more than offset the 4K run-time library appended to each compiled program, providing additional internal memory space.

The compiler implements true integer arithmetic as well as real arithmetic. Use of integers can lead to significant speed improvements. Special compile time options make identification and conversion of real variables to integers a simple task.

A 'Compiler' security key, which plugs into either cassette port, is supplied together with the DTL-BASIC compiler. This key must be used in order to compile a program or to run the compiled version. In order to allow for the distribution of compiled versions of user developed programs, a second type of key known as a 'Run-Time' key is available in any required quantities. Software developers can obtain private security key sets with unique serial numbers providing comprehensive protection of their products while allowing customers to make backup copies of compiled programs.

DTL-BASIC is a disk based system requiring a 32K PET/CBM and comes complete with an in-depth user manual and a Compiler Security Key. Three versions of the compiler exist for CBM 3032, CBM 4032, and CBM 8032 machines. Please specify machine type and disk type (4040 or 8050) on which compiler is to be supplied.

DTL BASIC WITH MANUAL AND SECURITY KEY $350.00

RUN-TIME KEYS .... $50.00 EACH

FROM

CMD

CANADIAN MICRO DISTRIBUTORS LTD.

365 Main St., Milton, Ontario L9T 1P7
416 878-7277

THE COMPANY THAT BROUGHT YOU
MUPET • DOUBLE-MUPET • SPOOLER
THE MANAGER • I/O PRODUCTS

descriptive statistics useful in evaluating degree of randomness: mean and variance of the fractions, and covariance and correlation coefficient of successive fractions. The expected mean is 0.5, or the midpoint of the uniform distribution of Figure 1. The expected variance is 1/12; this can be derived using a bit of integral calculus. Finally, the remaining two statistics are expected to be zero since the elements of our sequence of numbers are supposed to be independent. Table 3 shows results for the first three statistics. All conform very closely to expectations.

The X^2 test, the Runs Test, and a small battery of descriptive statistics suggest that RND (0) is a decent random number generator. But our evidence can never be conclusive, and the next test that we subject the generator to may be the one that it fails. So:

> Be not too presumptuously sure in any business; for things of this world depend on such a train of unseen chances that if it were in man's hands to set the tables, still he would not be certain to win the game.
>
> Herbert

References:

1. **Knuth, Donald E.** *The Art of Computer Programming*. Vol. 2. Reading: Addison-Wesley Publishing Company, 1971.

2. **Langley, Russell.** *Practical Statistics Simply Explained*. New York: Dover Publications, Inc., 1970.

TABLE 1

Distribution Of The First 100,000 Fractions Generated By RND (0)

INTERVAL	TALLY	% OF TOTAL
0.1	9969	9.97
0.1 to <0.2	10084	10.08
0.2 to <0.3	9980	9.98
0.3 to <0.4	9904	9.90
0.4 to <0.5	9985	9.99
0.5 to <0.6	10099	10.10
0.6 to <0.7	10098	10.10
0.7 to <0.8	9938	9.94
0.8 to <0.9	9774	9.77
0.9	10169	10.17

Uniform Distribution Between 0 And 1

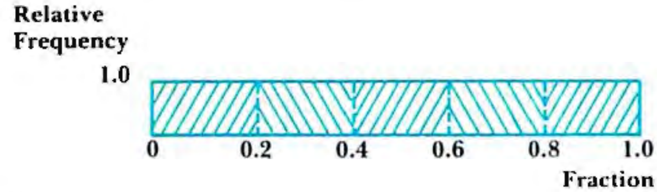


Figure 1.

Table 2. Test Results

Cumulative Number Of Fractions Generated	X^2 Values Block	X^2 Values Cumulative	"RUNS" Values Block	"RUNS" Values Cumulative
10,000	13.87	13.87	1.34	1.34
20,000	10.16	14.14	0.19	0.80
30,000	7.20	6.71	0.03	0.69
40,000	4.23	8.55	0.03	0.58
50,000	4.99	6.21	1.50	0.16
60,000	11.59	10.06	1.47	0.75
70,000	5.51	9.77	0.63	0.46
80,000	5.35	12.33	0.85	0.73
90,000	12.19	10.87	1.40	1.18
100,000	5.26	12.07	0.21	1.05

Table 3. Descriptive Statistics

Cumulative Number of Fractions Generated	Mean		Variance		Covariance	
	Block	Cumulative	Block	Cumulative	Block	Cumulative
Expected Values	0.500		0.083		0.000	
10,000	0.498	0.498	0.085	0.085	-0.001	-0.001
20,000	0.499	0.499	0.083	0.084	0.001	-0.000
30,000	0.499	0.499	0.083	0.084	-0.001	-0.000
40,000	0.499	0.499	0.083	0.083	0.001	-0.000
50,000	0.502	0.500	0.083	0.083	-0.000	-0.000
60,000	0.503	0.500	0.083	0.083	0.001	0.000
70,000	0.504	0.501	0.084	0.083	-0.001	-0.000
80,000	0.501	0.501	0.084	0.084	0.001	-0.000
90,000	0.496	0.500	0.084	0.084	0.001	0.000
100,000	0.501	0.500	0.083	0.083	-0.000	0.000

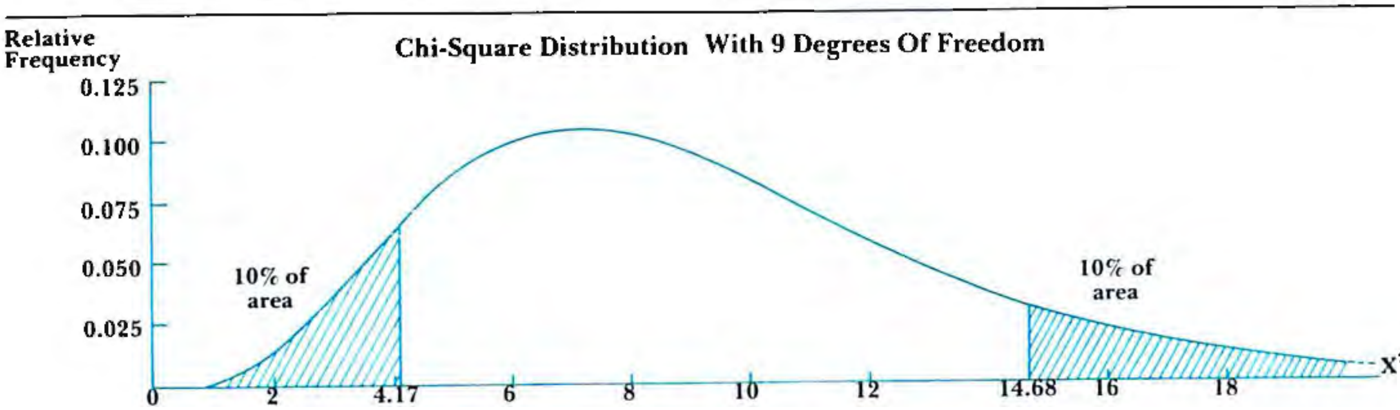


Figure 2.

Program 1.

```
10  REM EXAMINING THE RANDOMNESS OF A SEQUENCE OF FRACTIONS
20  REM CHI-SQUARE TEST, RUNS TEST, DESCRIPTIVE STATISTICS
30  REM BRIAN J. FLYNN; WINTER 1980/81
40  REM MOD 1: INITIALIZE
50  GOSUB 1000
60  REM MOD 2: GENERATE RANDOM NUMBERS & TALLY STATISTICS
70  GOSUB 2000
80  REM MOD 3: PRINT DISTRIBUTION OF FRACTIONS
90  GOSUB 3000
100 REM MOD 4: PRINT TEST STATISTICS
110 GOSUB 4000
120 REM MOD 5: PRINT DESCRIPTIVE STATISTICS
130 GOSUB 5000
140 END
```

```
1000 REM MODULE 1
1010 REM VARIABLES
1020   REM NOTE: 'BLK' MEANS 'BLOCK.' 'CUM' MEANS 'CUMULATIVE.'
1030   REM          FOR EVERY 'BLK,' THERE IS A 'CUM' ANALOGUE.
1040   REM          N = TOTAL NUMBER OF FRACTIONS TO GENERATE
1050   REM          B = NUMBER OF FRACTIONS PER BLOCK
1060   REM          K = NUMBER OF CELLS FOR CHI-SQUARE TEST
1070   REM QBLK( ) = VECTOR OF CELL COUNTS
1080   REM          I = NUMBER OF FRACTIONS GENERATED
1090   REM         RN = RANDOM NUMBER
1100   REM       CELL = INTERVAL WHICH A FRACTION FALLS INTO
1110   REM       BBLK = NUMBER OF FRACTIONS FALLING BELOW 0.5
1120   REM       ABLK = NUMBER OF FRACTIONS FALLING ABOVE 0.5
1130   REM      SIGN$ = "-" OR "+" FOR FRACTION BELOW OR ABOVE 0.5
1140   REM       RBLK = NUMBER OF RUNS
1150   REM      SBLK$ = 'SIGN$' OF PREVIOUS FRACTION GENERATED
1160   REM SBLK(1) = SUM OF FRACTIONS
1170   REM SBLK(2) = SUM OF SQUARED FRACTIONS
1180   REM SBLK(3) = SUM OF PRESENT x PREVIOUS FRACTIONS
1190   REM       PBLK = VALUE OF PREVIOUS FRACTION
1200   REM      F$( ) = VECTOR OF OUTPUT FORMATS
1210   REM          L = LINE COUNT
1220   REM       XBLK = CHI-SQUARE STATISTIC
1230   REM       FBLK = CORRECTION FACTOR FOR RUNS TEST
1240   REM        NUM = NUMERATOR OF RUNS TEST STATISTIC
1250   REM        DEN = DENOMINATOR OF RUNS TEST STATISTIC
1260   REM       ZBLK = RUNS TEST STATISTIC
1270   REM         E1 = EXPECTED MEAN OF THE FRACTIONS
1280   REM         E2 = EXPECTED VARIANCE OF THE FRACTIONS
1290   REM         E3 = EXPECTED COVARIANCE OF SUCCESSIVE FRACTIONS
1300   REM         E4 = EXPECTED IST-ORDER SERIAL CORRELATION COEFFICIENT
1310   REM       MBLK = ACTUAL MEAN
1320   REM       VBLK = ACTUAL VARIANCE
1330   REM       CBLK = ACTUAL COVARIANCE
1340   REM       HBLK = ACTUAL CORRELATION COEFFICIENT
1350 REM # OF FRACTIONS, BLOCK SIZE, & # OF CHI-SQUARE CELLS
1360   DATA 100000,10000,10
1370   READ N,B,K
1380 REM EXPECTED VALUES
1390   DATA .5,.083,0,0
1400   READ E1,E2,E3,E4
1410 REM INITIALIZE
1420   DIM QBLK(K),QCUM(K)
1430   FOR J=1 TO K:QBLK(J)=0:QCUM(J)=0:NEXT
1440   FOR J=1 TO 3:SBLK(J)=0:SCUM(J)=0:NEXT
```

THE MANAGER

The first truly user-friendly Database Management System available at reasonable cost.

This suite of programs is ideally suited for both the businessman and programmer, for use with the CBM 8032.

For the Business User

- Uses Menu Options – no programming experience needed.
- Lets you enter data in the form you wish, then lets you recall it using any search criteria.
- Performs predefined calculations on the record in realtime as record is displayed on the screen.
- Reports can be produced using any search criteria and/or arithmetic functions.
- Useful applications can be developed quickly.

Typical Applications include –

- Inventory Control
- Mailing Lists
- Accounting systems
- Personnel
- Costing
- Gathering test data
- Budgeting
- Scheduling
- Examples of use included on disk supplied.

As Programmers Tool

- Uses standard PET ASCII files.
- Software interface is in Basic and available to the programmer.
- No special disk formatting so that word processing or other programs can be stored on the same disk.
- No ROM Based Security thus no need to open CPU.
- Fast 'n' key Sort/Merge included.
- Full realtime intra & inter record arithmetic performed on the screen as record is displayed.
- Professional software support including unique security available.

TRY IT!
IF YOU ARE
NOT SATISFIED WITHIN 30 DAYS
WE WILL RETURN YOUR MONEY

$250.00

CMD CANADIAN MICRO DISTRIBUTORS LTD.
365 Main St., Milton, Ontario L9T 1P7
416 878-7277

DISTRIBUTORS OF:
MUPET • DOUBLE-MUPET • SPOOLER
THE MANAGER • I/O PRODUCTS

```
1450   I=0:ABLK=0:ACUM=0:BBLK=0:BCUM=0:SBLK$=" ":SCUM$=" ":SIGN$=" "
1460   RBLK=0:RCUM=0:PBLK=0:PCUM=0
1470 REM HEADING
1480   CLS:PRINT TAB(7)"EXAMINING THE RANDOMNESS OF A SEQUENCE OF FRACTIONS"
1490   PRINT:PRINT"NOTES:"
1500   PRINT" 1. TOTAL NUMBER OF FRACTIONS TO GENERATE = ";N
1510   PRINT" 2. NUMBER TO PUT INTO EACH 'BLOCK'        = ";B
1520   PRINT" 3. NUMBER OF CELLS FOR CHI-SQUARE TEST    = ";K
1530   PRINT" 4. CHANGE LINE 1360 FOR DIFFERENT VALUES THAN THESE.
1540   PRINT
1550   PRINT"HIT 'ENTER' TO PROCEED."
1560   INPUT"READY ";Z
1570 RETURN
```

```
2000 REM MODULE 2
2010 REM RANDOM FRACTION
2020   CLS:PRINT"GENERATING FRACTIONS ...
2030   RN=RND(0):I=I+1
2040 REM CHI-SQUARE TALLY
2050   REM IDENTIFY THE CELL THAT THE FRACTION FALLS INTO
2060     CELL = INT(RN*K) + 1
2070   REM ADD TO CELL COUNT
2080     QBLK(CELL) = QBLK(CELL) + 1
2090     QCUM(CELL) = QCUM(CELL) + 1
2100 REM RUNS-TEST TALLY
2110   REM ADD TO COUNTS OF FRACTIONS FALLING BELOW/ABOVE MEDIAN
2120     IF RN<.5 THEN BBLK = BBLK + 1:BCUM = BCUM + 1:SIGN$ = "-"
2130     IF RN>.5 THEN ABLK = ABLK + 1:ACUM = ACUM + 1:SIGN$ = "+"
2140   REM ADD TO RUN COUNT, IF APPROPRIATE
2150     IF SIGN$ <> SBLK$ THEN RBLK = RBLK + 1
2160     IF SIGN$ <> SCUM$ THEN RCUM = RCUM + 1
2170   REM RECORD SIGN
2180     SBLK$ = SIGN$
2190     SCUM$ = SIGN$
2200 REM DESCRIPTIVE-STATISTICS TALLY
2210   REM SUM OF FRACTIONS
2220     SBLK(1) = SBLK(1) + RN
2230     SCUM(1) = SCUM(1) + RN
2240   REM SUM OF SQUARED FRACTIONS
2250     SBLK(2) = SBLK(2) + RN*RN
2260     SCUM(2) = SCUM(2) + RN*RN
2270   REM SUM OF CURRENT x PREVIOUS FRACTIONS
2280     SBLK(3) = SBLK(3) + RN*PBLK
2290     SCUM(3) = SCUM(3) + RN*PCUM
2300   REM PREVIOUS VALUE OF FRACTION NOW EQUALS CURRENT VALUE
2310     PBLK = RN
2320     PCUM = RN
2330 REM GENERATE ANOTHER FRACTION, IF APPROPRIATE
2340    IF I/B <> INT(I/B) THEN 2030
2350 RETURN
```

```
3000 REM MODULE 3
3010 REM FORMATS
3020   F$(1)="     NO. OF FRACTIONS IN 'BLOCK'      = ########"
3030   F$(2)="     NO. OF FRACTIONS IN 'CUMULATIVE' = ########"
3040   F$(3)="1. DISTRIBUTION OF FRACTIONS:"
3050   F$(4)="                           BLOCK                 CUMULATIVE"
3060   F$(5)="     INTERVAL          TALLY  % OF TOT.       TALLY  % OF TOT.
3070   F$(6)="     ------------      -----  ----------      -----  ----------"
3080   F$(7)=">= #.### AND <#.###    #####     ###.##       #####      ###.##"
3090 REM HEADING
```

Model EP-2A-88
EPROM Programmer

★ **Easy to use**
★ **Reliable**
★ **Field proven**

Fast as Jackrabbits . . . Well, almost!

In Australia, two rabbits can reproduce over 13 million offspring in three years . . At 105 seconds for 2716's, the **EP-2A-88** can reproduce 1,892,160 EPROMS in three years. Single push button control, the EP-2A-88 checks if EPROMS are erased, programs and verifies. Many features, including self test, diagnostics and audio prompt.

The **EP-2A-88-1** will accept Copy (CM) modules for the 2758, and 2716 EPROMS. The **EP-2A-88-2** will accept copy modules for the 2716, 2732 and TMS 2532 EPROMS. Power requirements are 115 VAC 50/60 Hertz at 15 watts.

Part No.	Description	Price
EP-2A-88-1	EPROM Programmer	$490.00
EP-2A-88-2	EPROM Programmer	490.00
CM-50	Copy Module for 2716, TMS 2516 EPROMS	25.00
CM-70	Copy Module for 2758, TMS 2508 EPROMS	25.00
CM-20	Copy Module for 2732 EPROMS	25.00
CM-20-A	Copy Module for 2732A EPROMS	33.00
CM-40	Copy Module for TMS 2532 EPROMS	25.00
	Non Standard Voltage Option (Specify 220v, 240v, or 100v)	15.00

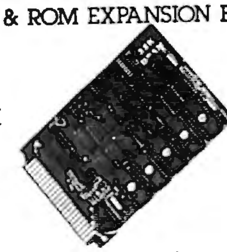
Optimal Technology, Inc.

Phone (804) 973-5482
Blue Wood 127 Earlysville, VA 22936

NEED MORE MEMORY?

32K BYTE DYNAMIC RAM & ROM EXPANSION BOARD

Expand Your 4K/8K PET
SYM/KIM/AIM −65 to 32K



- Easily connected to your computer via the expansion connector
- Build huge and complex programs!
- Need 64K of RAM? Buy two boards on board configuration circuitry will allow you to expand to 64K easily!
- New dynamic RAM technology brings you more memory in less space and at a lower cost!
- RAM chips are upgraded, compatible with the new 64K RAM chips for future expansion!
- Operates on +5 volts only supplied from your computer power supply no on board generators to go bad
- Requires A LOT less power than static RAM!
- Has full invisible refresh operation does not interfere with processor operation
- Fully buffered DATA BUSS
- 5 on board sockets for 2716/2732 (2K/4K) type EPROMS addressable anywhere
- Great for designing a two board computer system (CPU I/O-RAM ROM)
- Other specifications
  Disable any 4K block of RAM for I/O
  place RAM above or below 8000 HEX
  KIM-4 BUSS COMPATIBLE FOR CARD RACKS Adapter cables available for non rack use
- All these features on a 6 x 4.5" board!

**ASSEMBLED & TESTED BOARDS—GUARANTEED FOR 6 MONTHS
PURCHASE PRICE IS FULLY REFUNDABLE IF RETURNED
UNDAMAGED WITHIN 14 DAYS**

List Price — **$289.88**
Introductory Price — **$269.88**
Include $2.00 for S&H — Allow 4 weeks for delivery

Full informative documentation included with all our products.
C.O.D. Orders Accepted (702) 361-6331 Mail Order Only.

PROTRONICS
COMPUTER INNOVATIONS

**1516 E. Tropicana, Suite 7A
Las Vegas, Nevada 89109**

IBM COMPATIBLE 8" FLOPPY DISK for CBM/PET,* Aim, Sym

- HIGH SPEED — at least FOUR times faster than the 488 bus disks.
- HIGH CAPACITY — up to 850K bytes of storage in a three drive system.
- RELIABILITY — wide timing margins insure long trouble-free operation.
- SOFTWARE — the PDOS II disk software provides a full set of BASIC commands with standard syntax.

5¼" MINI FLOPPY DISK SYSTEMS:

Model 540-1 Single Drive, Double Density (143K)	$595.00
Model 540-2 Dual Drive, Double Density (286K)	$895.00
Model 580-1 Single Drive, Quad Density (286K)	$795.00
Model 580-2 Dual Drive, Quad Density (572K)	$1,195.00

8" MINI FLOPPY DISK SYSTEMS:

Model 877-1 Single Drive, IBM standard (295K)	$1,095.00
Model 877-2 Dual Drive, IBM standard (590K)	$1,695.00

PEDISK II

fullFORTH +

FULL FEATURE "FORTH" FOR 6502 SYSTEMS
Interpreter — Cross-Compiler — Cond. Assembler —
Screen Editor — String Handling — Floating Point
SPECIFY PEDISK II, PET 2040 or 4040 DISK, OR APPLE $75.00

FOR INFORMATION, SEE YOUR DEALER OR:

cgrs MICROTECH

DEALER INQUIRIES INVITED

ADD-ON DRIVES FOR TRS-80*

Model I, 5¼", 40 Track (102K)	$329.00
Model I, 5¼", 80 Track (204K)	$429.00

P.O. BOX 102 • LANGHORNE, PA 19047 • (215) 757-0284

*PET IS A TRADEMARK OF COMMODORE

```
3100    CLS
3110    LPRINT F$(3)
3120    LPRINT USING F$(1);B
3130    LPRINT USING F$(2);I
3140    LPRINT:LPRINT F$(4):LPRINT F$(5):LPRINT F$(6)
3150 REM DISTRIBUTION
3160    REM LINE COUNT
3170      L=7
3180    REM FIGURES
3190      FOR J=1 TO K
3210      LPRINT USING F$(7);(J-1)/K,J/K,QBLK(J),QBLK(J)*100/B,QCUM(J),QCUM(J)*10
0/I
3220      L=L+1
3230      NEXT J
3240      LPRINT:LPRINT:LPRINT
3250 RETURN
```

```
4000 REM MODULE 4
4010 REM FORMATS
4020    F$(3)="2. TEST STATISTICS:"
4030    F$(4)="      STATISTIC                      BLOCK           CUMULATIVE"
4040    F$(5)="      ---------                      -----           ----------"
4050    F$(6)="      CHI-SQUARE                   ####.##              ####.##"
4060    F$(7)="      RUNS                         ####.##              ####.##"
4070 REM HEADING
4080    CLS
4090    LPRINT F$(3)
4100    LPRINT USING F$(1);B
4110    LPRINT USING F$(2);I
4120    LPRINT:LPRINT F$(4):LPRINT F$(5)
4130 REM CHI-SQUARE STATISTICS
4140    XBLK=0:XCUM=0
4150    FOR J=1 TO K
4160    XBLK = XBLK + (QBLK(J)-B/K)*(QBLK(J)-B/K)/(B/K)
4170    XCUM = XCUM + (QCUM(J)-I/K)*(QCUM(J)-I/K)/(I/K)
4180    NEXT J
4190    LPRINT USING F$(6);XBLK,XCUM
4200 REM RUNS-TEST STATISTICS
4210    REM YATES' FACTORS TO BETTER APPROXIMATE  A NORMAL DISTRIBUTION
4220      FBLK=0:FCUM=0
4230      IF RBLK < 2*ABLK*BBLK/B THEN FBLK = .5
4240      IF RBLK > 2*ABLK*BBLK/B THEN FBLK = -1.5
4250      IF RCUM < 2*ACUM*BCUM/I THEN FCUM = .5
4260      IF RCUM > 2*ACUM*BCUM/I THEN FCUM = -1.5
4270    REM STATISTICS
4280      NUM = ABS(2*ABLK*BBLK/B - RBLK) + FBLK
4290      DEN = SQR((4*ABLK*ABLK*BBLK*BBLK-2*B*ABLK*BBLK)/(B*B*B-B*B))
4300      ZBLK = NUM/DEN
4310      NUM = ABS(2*ACUM*BCUM/I - RCUM) + FCUM
4320      DEN = SQR((4*ACUM*ACUM*BCUM*BCUM-2*I*ACUM*BCUM)/(I*I*I-I*I))
4330      ZCUM = NUM/DEN
4340      LPRINT USING F$(7);ZBLK,ZCUM
4350      LPRINT
4360 REM DEGREES OF FREEDOM
4370    LPRINT"      NOTE: DEGREES OF FREEDOM FOR CHI-SQUARE TEST =";K-1
4380    LPRINT:LPRINT:LPRINT
4390 RETURN
```

```
5000 REM MODULE 5
5010 REM FORMATS
5020    F$(3)="3. DESCRIPTIVE STATISTICS:"
```

```
5030   F$(4)="                                          COVARIANCE OF        SERIAL"
5040   F$(5)="                                            SUCCESSIVE     CORRELATION"
5050   F$(6)="    VALUES          MEAN      VARIANCE      FRACTIONS      COEFFICIENT"
5060   F$(7)="    -------         ----      --------     -----------    -----------"
5070   F$(8)="     ###.###      ###.###          ###.###          ###.###"
5080 REM HEADING
5090   CLS
5100   LPRINT F$(3)
5110   LPRINT USING F$(1);B
5120   LPRINT USING F$(2);I
5130   LPRINT:LPRINT F$(4):LPRINT F$(5):LPRINT F$(6):LPRINT F$(7)
5140 REM EXPECTED VALUES
5150   LPRINT"  EXPECTED";
5160   LPRINT USING F$(8);E1,E2,E3,E4
5170 REM MEANS
5180   MBLK = SBLK(1)/B
5190   MCUM = SCUM(1)/I
5200 REM VARIANCES & COVARIANCES
5210   VBLK = (SBLK(2)-SBLK(1)*SBLK(1)/B)/(B-1)
5220   VCUM = (SCUM(2)-SCUM(1)*SCUM(1)/I)/(I-1)
5230   CBLK = (SBLK(3)-SBLK(1)*SBLK(1)/B)/(B-1)
5240   CCUM = (SCUM(3)-SCUM(1)*SCUM(1)/I)/(I-1)
5250 REM SERIAL CORRELATION COEFFICIENT
5260   HBLK = CBLK/VBLK
5270   HCUM = CCUM/VCUM
5280 REM 'BLOCK' RESULTS
5290   LPRINT
5300   LPRINT"     BLOCK";
5310   LPRINT USING F$(8);MBLK,VBLK,CBLK,HBLK
5320   LPRINT"      BIAS";
5330   LPRINT USING F$(8);MBLK-E1,VBLK
  -E2,CBLK-E3,HBLK-E4
5340 REM 'CUMULATIVE' RESULTS
5350   LPRINT
5360   LPRINT"CUMULATIVE";
5370   LPRINT USING F$(8);MCUM,VCUM,
  CCUM,HCUM
5380   LPRINT"      BIAS";
5390   LPRINT USING F$(8);MCUM-E1,VCUM
  -E2,CCUM-E3,HCUM-E4
5400   FOR J=1TO 15:LPRINT:NEXT J
5410 REM RESET 'BLOCK' VARIABLES &
  GENERATE MORE FRACTIONS, IF
  APPROPRIATE
5420   IF I = N THEN 5460
5430   FOR J=1 TO K:QBLK(J) = 0:NEXT J
5440   ABLK=0:BBLK=0:SBLK$=" ":SBLK(1)
  =0:SBLK(2)=0:SBLK(3)=0:RBLK=0:PBLK
  =0
5450   GOSUB 2020 :GOSUB 3020 :GOSUB
  4020 :GOTO 5020
5460 RETURN
```

©

TELECOMMUNICATIONS

Getting Outside The Computer

Michael E. Day
Chief Engineer
Edge Technology

Getting a computer to communicate with the outside world is not an easy task. In fact, many of the "internals" of the typical computer are devoted to the task of converting information from or to a form that the computer can understand.

Sometimes attached equipment (peripherals) is designed to meet the computer part way. This helps to reduce the circuitry and/or work that the computer needs to convert the information into or out of a form it can understand.

Keyboards are sometimes set up this way. Other times, due to the complexity of the work involved, a large amount of the work must be done by the computer.

Monitors, for example, can cause some difficulty since, if the computer is spending too much time "servicing" the attached devices, too little time is left to run the program. An example of an extreme case of this is the SINCLAIR ZX80 which actually spends all of its time servicing the monitor and keyboard. Because of this, it must stop servicing the display in order to run a program (causing the display to go blank). The ZX81 cures this problem by having a slightly improved display service routine which gives the computer a little time to squeeze in the program.

In order to solve this problem, IC manufacturers came up with a "Video Controller" IC. This little chip (a computer in its own right!) does all the service work for the computer, and allows the computer to do more important things like running your program.

The UART

When the computer is to communicate over the phone line, the same problem occurs. The computer can spend a large amount of time doing the required work, or we can bring in another device to do it for the computer. This is called the UART (Universal Asynchronous Receiver Transmitter) or USART (Universal Synchronous / Asynchronous Receiver Transmitter) depending on which flavor you like.

A UART accepts information from the computer in a form which the computer understands and converts it to the form necessary to transmit the information out of the computer. Additionally, it accepts information sent to the computer and converts it to a form which the computer uses.

Inside the computer, we deal with data in a form called *byte*. When this is translated to the form it takes on the outside it becomes a *character*. A *byte* is made up of eight *bits*, with *bit* being the simplest form of data representation inside a computer. A bit consists of nothing more than an ON or OFF condition. When a computer is using the information it works with all eight bits of the byte at once. This is called *parallel* operation since eight bits are used simultaneously. Since each bit has two possible conditions (*on* or *off*) and since we are working with eight bits at once, this means that by using these bits together we can represent two to the eighth (256) possible conditions.

By taking some of these 256 states and defining them as representing something such as characters in the alphabet, we provide a means for the computer to work with information as we humans understand it.

Since the computer must know whether data is your information or its own, one of the bits is usually set aside to indicate this. This leaves us two to the seventh (128) possible things which we can represent as our own information. When using the computer to communicate to other equipment, the equipment generally requires certain "control codes" to perform some of its functions — returning the carriage on a printer or clearing the screen on a video terminal. Generally, 34 of the possible representations* are set aside for the purpose of controlling equipment. This leaves 94 possibilities left to represent all the characters in the alphabet (both uppercase and lowercase letters) the numbers (0 through 9) and some of the more commonly used symbols.

When we want to send this information over the phone lines, we run into a problem. The phone network is an entirely different environment, and is not at all compatible with computerized information.

Digital Into Audio

This is where the MODEM comes in. The MODEM changes the digital signals which the computer likes into the audio signals the phone network likes. (For this discussion we will assume that a BELL 103 compatible MODEM is being used in the originate mode.)

Introducing the M line . . .

Now! Drive Systems for AIM, KIM and SYM Computers — from PERCOM.

Assembled and tested systems start at only $599.95, including the drive controller circuit card, disk-operating system, interconnecting cable, drive and comprehensive users manual.

- **The right storage capacity** – Available in 1-, 2- and 3-drive systems, with either 40- or 80-track drives.
- **Flippy storage** – Flippy drives (optional) let you flip a diskette and store data and programs on the second recording surface.
- **High Storage Capacity** – Formatted, one-side storage capacity is 102 Kbytes (40-track drive), 205 Kbytes (80-track drive).
- **Proven Controller** – The drive controller design is the same as the design used in the Percom 680X LFD mini-disk system. This system – introduced in 1977 – has given reliable service in thousands of applications. Two versions are available: the MFD-C65 for the AIM-65 expansion bus, and the MFD-C50 for the System-50 (SS-50) bus.
  - Includes an explicit data separator circuit that's reliable even at the highest bit densities.
  - Provides for on-card firmware.
  - Includes a motor inactivity time-out circuit.
  - Capable of handling up to four drives.
  - Capable of reading both hard- and soft-sectored diskettes.
- **DOS included** – The MFD disk-operating system works with the AIM monitor, editor, assembler, Basic and PL/65 programs. Interface is direct, through user I/O and F1, F2 keys. Diskette includes DOS source code and library of 20 utility commands.
- **Reliability assurance** – Drives are burned-in 48 hours, under operating conditions, to flag and remove any units with latent defects.
- **Full documentation** – Comprehensive hardware and software manuals are included with each system.

Now! Expand your AIM-65 with Low-cost System-50 Modules.

The Percom M65/50 Interface Adapter connects your M-65 bus to Percom's System-50 (SS-50) motherboard, allowing you to expand your AIM, KIM or SYM with proven System-50 modules. You can add disk storage, memory modules,even a video display system. The M65/50 provides buffer-amplification of address, data and control lines. On-card decode circuitry lets you allocate address space either to the computer or to the expansion motherboard. Price: only $89.95, *including* System-50 motherboard.

System Requirements: AIM-65, KIM or SYM computer with expansion bus and four Kbytes RAM (min).

Yes . . . I'd like to know more about Percom MFD drive systems. Rush me free literature.

Send to
PERCOM DATA COMPANY, Inc., Dept. 65-C
11220 Pagemill Rd Dallas, TX 75243

name

address

city state

zip phone number

MAIL TODAY!

PERCOM

PERCOM DATA COMPANY, INC.
11220 PAGEMILL RD DALLAS, TX 75243
(214) 340-7081

Toll-Free Order Number: **1-800-527-1222**

PRICES AND SPECIFICATIONS SUBJECT TO CHANGE WITHOUT NOTICE

When an *on* condition is sent to the MODEM, it transmits a 1270 Hz signal. When an *off* condition is sent to the MODEM, it transmits a 1070 Hz audio signal.

This allows us to send digital signals, but we still have a problem. The computer is dealing with 256 possible combinations at any one time, but the modem can only handle two conditions at any one time. This is what the UART is for. The UART takes a byte that the computer feeds it and breaks it down into bits that can be sent through the MODEM. In order to do this it takes the eight bits which make up the byte and transmits them one at a time. The timing of this is critical. The computer at the other end (the remote unit) that is receiving this information has to have some way to reconstruct this sequence back into the byte that the computer can understand. It must have a UART to do this. We can't just send the information whenever we feel like it.

Timing Is Critical

The first thing we have to do is define what a bit will be. This is defined as being an *on* or *off* condition for a specified period of time (This is referred to as the *bit time* or *bit rate*.) This way, the remote UART can know that, once it starts to receive the information, the first bit will be presented to it for one *bit time*. Then, the next bit will be presented to it for another *bit time*. This continues until the last bit has been sent.

We also must specify which bit is to be sent first. By taking the bits that make up a byte and labeling one of them as the Least Significant Bit (LSB), and another as the Most Significant Bit (MSB) we can define that we will send the LSB first followed by the next to the least significant bit until we reach the most significant bit which is the last one that gets sent. This allows the remote UART to know what order the bits are being sent and it can reconstruct a byte properly.

Since this is a time-dependent activity, we need to have a way to synchronize the two UARTs so that the one that is receiving the information is looking at the right bit at the right time. To do this, the UART adds an extra bit to the byte that is being sent called a *start bit*. When no information is being sent, the UART will send a continuous *on* signal to indicate that it is in an *idle* condition (sometimes referred to as a *marking* condition). When something is to be sent, the UART will send a single *off* bit to let the remote UART know that it should begin collecting bits.

Since the computer only uses seven bits to represent a character, the UART makes use of the eighth bit for itself. This is usually referred to as the *parity* bit. The parity bit is created by adding up the number of *on* bits in the character that is being sent and, if there are an even number of *on* bits in the character, the parity bit is turned *on*. If there is an odd number, it is turned off (assuming that the standard *even parity* convention is being used). This lets the remote UART know if the information sent was lost or damaged during transmission. The remote UART does this by adding up the *on* bits it receives and then compares this sum to the parity bit that the first UART sends it.

One Final Problem

Now we have only one final problem with which to deal. The remote UART knows that the transmission has begun when it receives the start bit, but what happens if the parity bit (which is the last bit sent) is an *off* bit like the start bit? The remote UART has to have some way of being able to recognize the next start bit. To do this, we have to insure that an *idle* condition always precedes the start bit so that there will always be an *on* condition prior to the *off* condition generated by the start bit. So, the UART adds one final bit to the information called the *stop bit*. The stop bit is always an *on* bit insuring that there is always an idle condition generated before the next is sent. This means that a sum total of ten bit times is required for the computer to send each character.

In the next column, I will discuss the interreaction of the UART and the computer with regard to timing.

* Note: the "space" as an alphabetic character does not exist. It is, in reality, a control function. It has come to be used and referred to as a *character* because it is simpler to represent this non-character condition in the context in which real characters are used. This is analogous to the number *zero* which is not really a number, but is used as one to represent the condition of nothingness. ©

DYNACOMP

Quality software for:*

ATARI	**TRS-80 (Level II)****
PET	**NORTH STAR**
APPLE II Plus	**CP/M Disks/Diskettes**

DYNACOMP OFFERS THE FOLLOWING

- Widest variety
- Guaranteed quality
- Fastest delivery
- Friendly customer service
- Free catalog
- 24 hour order phone

CARD GAMES

BRIDGE 2.0 (Available for all computers) — **Price: $17.95 Cassette/$21.95 Diskette**
An all-inclusive version of this most popular of card games This program both BIDS and PLAYS either contract or duplicate bridge Depending on the contract, your computer opponents will either play the offense OR defense If you bid too high, the computer will double your contract' BRIDGE 2 0 provides challenging entertainment for advanced players and is an excellent learning tool for the bridge novice See the software review in 80 Software Critique Rated #1 by Creative Computing.

HEARTS 1.5 (Available for all computers) — **Price: $15.95 Cassette/$19.95 Diskette**
An exciting and entertaining computer version of this popular card game Hearts is a trick-oriented game in which the purpose is not to take any hearts or the queen of spades Play against two computer opponents who are armed with hard-to-beat playing strategies HEARTS 1 5 is an ideal game for introducing the uninitiated (your spouse) to computers See the software review in 80 Software Critique

STUD POKER (Atari only) — **Price: $11.95 Cassette/$15.95 Diskette**
This is the classic gambler's card game The computer deals the cards one at a time and you (and the computer) bet on what you see. The computer does not cheat and *usually* bets the odds. However, it sometimes bluffs! Also included is a five card draw poker betting practice program. This package will run on a 16K ATARI Color, graphics, sound. See review in COMPUTE.

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

CRIBBAGE 2.0 (TRS-80 only) — **Price: $14.95 Cassette/$18.95 Diskette**
This is simply the best cribbage game available It is an excellent program for the cribbage player in search of a worthy opponent as well as for the novice wishing to improve his game The graphics are superb and assembly language routines provide rapid execution. See the software review in 80 Software Critique.

THOUGHT PROVOKERS

MANAGEMENT SIMULATOR (Atari, North Star 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 review in 80 Software Critique.

BACKGAMMON 2.0 (Atari, North Star 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.

CHECKERS 3.0 (PET only) — **Price: $16.95 Cassette/$20.95 Diskette**
This is one of the most challenging checkers programs available It has 10 levels of play and allows the user to change skill levels at any time. Although providing a very tough game at level 4-8, CHECKERS 3.0 is practically unbeatable at levels 9 and 10.

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.

LEM LANDER (32K Apple Disk only) — **Price: $16.95 Diskette**
Pilot your LEM LANDER to a safe landing on any of nine different surfaces ranging from smooth to treacherous. The game paddles are used to control craft attitude and thrust This is a real-time high res challenge!

FOREST FIRE! (Atari only) — **Price: $16.95 Cassette/$20.95 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.

NOMINOES JIGSAW (Atari, Apple and TRS-80 only) — **Price: $16.95 Cassette/$20.95 Diskette**
A jigsaw puzzle on your computer! Complete the puzzle by selecting your pieces from a table consisting of 80 different shapes NOMINOES JIGSAW is a virtuoso programming effort The graphics are superlative and the puzzle will challenge you with its three levels of difficulty Scoring is based upon the number of guesses taken and by the difficulty of the board set-up. See review in ELECTRONIC GAMES

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

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

SPACE LANES (Available for all computers) — **Price: $14.95 Diskette**
SPACE LANES is a simple but exciting space transportation game which involves up to four players (including the computer). The object is to form and expand space transportation companies in a competitive environment. The goal is to amass more net worth than your opponent. The economics include stock purchases and company mergers. Watch your wealth grow!

**ATARI, PET, TRS-80, NORTHSTAR, CP/M and IBM are registered tradenames and/or trademarks*

***Except where noted, all model I software is available for the Model III. TRS-80 diskettes are not supplied with DOS or BASIC.*

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

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.

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.

MOVING MAZE (Apple and Atari only) — **Price: $10.95 Cassette/$14.95 Diskette**
MOVING MAZE employs the games paddles to direct a puck from one side of a maze to the other However, the maze is dynamically (and randomly) built and is continually being modified The objective is to cross the maze without touching (or being hit by) a wall. Scoring is by an elapsed time indicator, and three levels of play are provided.

ALPHA FIGHTER (Atari only) — **Price: $14.95 Cassette/$18.95 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: $16.95 Cassette/$20.95 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: $16.95 Cassette/$20.95 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.

GIANT SLALOM (Atari only) — **Price: $14.95 Cassette/$18.95 Diskette**
This real-time action game is guaranteed addictive! Use the joystick to control your path through slalom courses consisting of both open and closed gates Choose from different levels of difficulty, race against other players or simply take practice runs against the clock. GIANT SLALOM will run on 16K systems.

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.

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.

ADVENTURE

CRANSTON MANOR ADVENTURE (North Star and CP/M only) — **Price: $21.95 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!

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!

Price: $329.95 (Please add $4.00 for shipping and handling)

TNT Software

The following DYNACOMP programs are available for use with TNT:

STUD POKER (Atari, 24K)
NOMINOES JIGSAW (Atari, 24K)
TEACHER'S PET I (Atari and North Star)
BRIDGE 2.0 (North Star)
CHOMPELO (Atari, 24K)

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

Please specify 'TNT' versions when ordering.

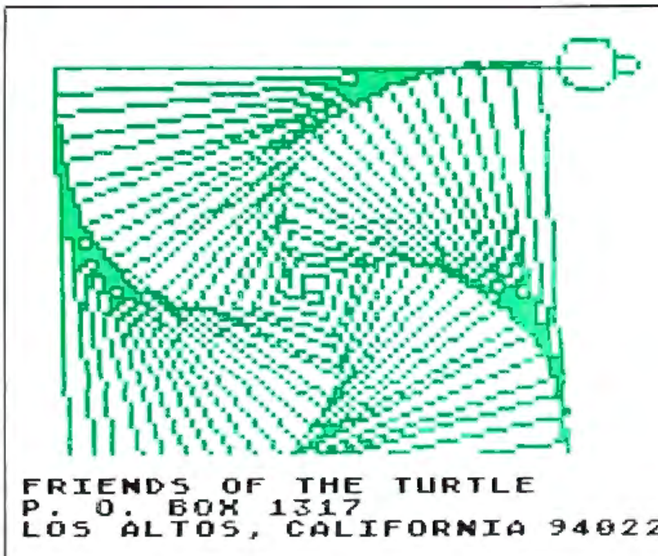
ABOUT DYNACOMP

DYNACOMP is a leading distributor of small system software with sales spanning the world (currently in excess of 40 countries). During the past two years we have greatly enlarged the DYNACOMP product line, but have maintained and improved our high level of quality and customer support. The achievement in quality is apparent from our many repeat customers and the software reviews in such publications as COMPUTRONICS, 80 Software Critique and A.N.A.L.O.G. Our customer support is as close as your phone. It is always friendly. The staff is highly trained and always willing to discuss products or give advice.

Education

Friends Of The Turtle

David D Thornburg
Innovision
Los Altos CA



Welcome to a new Society – the Friends of the Turtle. A free membership in this society is available to all subscribers to this magazine, and our meetings will be held on these pages every issue. The goal of this society is to promote the type of computer graphics and robot environment that uses what is called "turtle geometry."

Turtle geometry is a key element in several user-friendly computer languages such as Atari PILOT, TI LOGO, and Apple LOGO. It may come as a surprise to some of you, but the types of graphics commands used in these languages are also obeyed by a programmable toy – the Milton Bradley Big Trak. Turtle geometry encourages exploration. It can be learned by first-time computer users of almost any age, and its power is so great that it can keep full-fledged computer wizards engrossed for years. The turtle is a graphics tool that makes it easy for you to get the computer to do what *you* want it to do.

In these pages we will share programs that illustrate many interesting ideas and developments in this field. Most of all, we will share beautiful designs that have come out of this computer environment.

Background – What Is A Turtle?

If you have ever played with a Milton Bradley Big Trak, or used computer languages like Atari PILOT, LOGO, or WSFN, you have encountered a very special device called a turtle. Basically, a turtle is a "robot" that can move around the floor (or display screen) in response to messages you send it. Display turtles often have "pens" with which they can leave traces of their path as they move. This makes the turtle a handy tool for drawing pictures.

The difference between turtle graphics and conventional coordinate graphics can be demonstrated by drawing a square in both systems.

In coordinate geometry, the pen is moved to various coordinates on a grid. To draw a square 40 units on a side, we could use these five steps:

GOTO 0,0 (put the pen at the origin)
DRAWTO 0,40 (draw the left vertical line)
DRAWTO 40,40 (draw the top horizontal line)
DRAWTO 40,0 (draw the right vertical line)
DRAWTO 0,0 (draw the bottom horizontal line)

This is illustrated below.

Figure 1a.

Figure 1b.

Figure 1c.

Figure 1d.

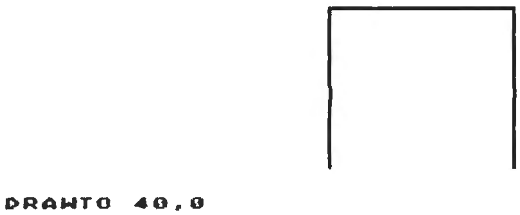
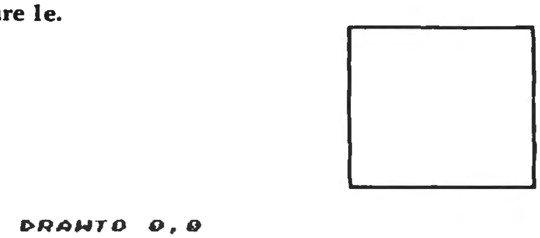


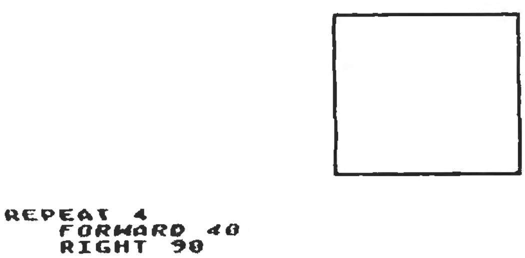
Figure 1e.



Next, let's see how a square would be drawn in turtle geometry. We make the turtle draw lines by giving a sequence of instructions like this:

REPEAT 4 (repeat the following commands 4 times)
FORWARD 40 (draw a line 40 units long)
RIGHT 90 (turn 90 degrees to the right)

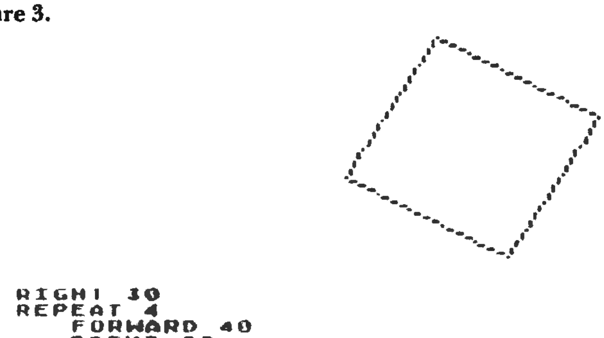
Figure 2.



While the turtle commands that draw a square are much simpler than the commands in coordinate geometry, this is far from being their only power. The coordinate representation we showed only describes a square with vertical and horizontal sides. Suppose you wanted to draw a square tilted at some angle (say 30 degrees). How would you draw that in coordinate geometry?

In turtle geometry, the description of one square is just the same as that for any other square, independent of its orientation. To draw a square tilted at 30 degrees, you first must turn the turtle by 30 degrees before having it draw the square.

Figure 3.



The commands look like this:

RIGHT 30
REPEAT 4
FORWARD 40
RIGHT 90

The power of turtle geometry is so great that we cannot begin to touch it in this first column. If you want more information between now and the next "meeting," you should read "Picture This! PILOT's Turtle Graphics for Atari" in the May-June 1981, issue of *Recreational Computing.* Two important books on this topic have recently been published – *Mindstorms: Children, Computers, and Powerful Ideas* by Seymour Papert (Basic Books), and *Turtle Geometry: The Computer as a Medium for Exploring Mathematics* by Harold Abelson and Andrea diSessa (MIT Press). A new book (by the author), *Picture This!,* will be published by Addison Wesley in early 1982. This book focuses on the Turtle Language incorporated into Atari PILOT.

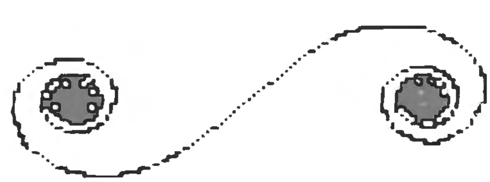
Why Do We Need Friends Of The Turtle?

As we said before, turtle geometry is being incorporated in many of the computer languages that are just now beginning to be available on low-cost personal computers. Each of the various implementations of this environment has its special features and limitations. To the extent that the graphics environments in all these implementations are similar to each other, Friends of the Turtle will be a place where we can explore the turtle world in a machine independent fashion. We will describe all sorts of interesting experiments to do with turtles (since experimenting is probably the best way to learn geometry anyway), share our programs, provide a "Rosetta Stone" for various dialects of turtle languages, keep track of recent developments in the field, and generally have a good time. This last point is the most important, since the turtle is a marvelous device to play with.

So, welcome to friends of the Turtle. Please write to me with your ideas and programs. If you are new to this field you should know that we will spend a great deal of time dealing with the basics.

Turtles are for everyone, and so is this society. Please write to me at the following address:

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



©

Learning With Computers

Glenn M. Kleiman and Mary M. Humphrey
Teaching Tools: Microcomputer Services
P.O. Box 50065
Palo Alto, CA 94303

How might existing computer technology change schools in the near future?

In this column, we recount a hypothetical visit to the Charles Babbage School, circa 1985. Our tour guide was the principal, Ada Lovelace, who told us the school has been using computers since 1982.

At Babbage School, children move about a great deal, working individually and in groups on different lessons and projects. The children have a lot of flexibility in which lessons they do when, and in how they approach studying a given topic. Everyday attendance is not compulsory, and some children often take lesson disks home to work on their own computers. Teachers generally work with individuals or small groups of children. Ms. Lovelace told us the teachers spend most of their time tutoring and directing children's learning. The students have a lot of choice, but the teachers make sure that each child engages in a balanced variety of activities each week. Very little time is spent in record keeping or grading – computers take care of that. Since computers make truly individualized instruction possible, grading is not emphasized as it once was.

Many lessons are very different from those in schools of 1980. For example, nine-year-old Jane showed us a computer lesson on ecology and pollution. The computer showed a lake with a variety of plants and fish. It also provided information about the food chain and reproduction rates of the species within the lake. Jane then told the computer that a certain pollutant had entered the lake. The computer responded that the pollutant had killed 50% of the "glod" plants, and asked Jane to predict the effect of this on the other life in the lake over the next five years. Jane then compared her predictions to the actual effects calculated by the computer, finding that she had estimated much less damage than would have occurred.

This simulation certainly seemed to teach her the basic principles of an ecological system. Computer simulations are available at Babbage School for many science lessons. Ms. Lovelace told us that she hopes to get simulation programs to teach principles of economics and social psychology. She pointed out that software development has lagged behind hardware advances ever since she first worked with computers.

Lessons As Games

Other lessons take a more game-like format, often with two or more players. Competitive games requiring (and providing practice in) math and language skills are very popular. Several children were playing an adventure game in which they explore a complicated world created within the computer. They search through castles, caves, and mazes for treasures, while trying to avoid the dangers of creatures such as wizards, dragons, and gremlins. Lessons in reading comprehension, logic, and map reading were embedded within the game.

Ms. Lovelace said that some children spend a lot of time with these game-lessons, and that completing one adventure can take several weeks. Teachers can instruct the computer to modify the game as it is being played. They use this capability to introduce new vocabulary words and other educational material, and to encourage the children to do other lessons. For example, 12 year-old Jim (who told us that "adventure is a real classic computer game") often neglected his science lessons. A quick modification by one teacher added a wizard to the adventure. This wizard gave Jim instructions for finding a treasure which required knowledge about certain star constellations. We later saw Jim engrossed in an astronomy lesson.

Ms. Lovelace told us that the children learn a great deal by exploring environments simulated on the computer. For example, one program creates computer screen representations of gears, pulleys, wheels, levers and so on. The child can combine these simple machines on the screen to create devices to perform various jobs, such as moving heavy objects. The device created can be tested through computer simulations to see if it works as planned. The child can then modify and re-test the device, or build a new one.

Creating, testing, and modifying devices in this simulated environment produces an understanding of the principles of simple mechanical machines. Other programs available at Babbage School create environments in which children can explore geometry, physics, and simple computer operations. Ms. Lovelace expressed the hope that more such programs would be available soon since this type of learning makes abstract concepts more concrete and manageable for children. Also, children learn through active exploration, rather than just passively remembering information given to them.

Writing And Typing Skills

Several students were engaged in writing projects. All the writing was done using word processing programs. The children easily entered and then revised their writing. Everything from correcting spelling errors and adding or deleting words to rearranging paragraphs was done quickly on the screen. Using word processors makes writing more enjoyable and children are willing to revise their own work many times – something they are reluctant to do when they have to rewrite by hand each time.

... teachers have time for individual tutoring ...

We expressed surprise that all the children knew how to type so well. Ms. Lovelace told us that they had learned from a computer program. The program presents typing drills and measures how long it takes to complete the drill on the computer keyboard. Later drills are designed to give practice with letters or letter combinations the child has typed incorrectly or too slowly. Since practice is directed at specific problems, learning is very rapid.

Some of the children were writing articles or stories for the school newspaper. One child told us he was writing a science fiction story about what the world would be like without any computers. When he finished his story, he stored a copy on disk so the newspaper editor could edit it later. We were told that, after being approved by the editor, the newspaper was automatically formatted and printed by the computer.

Other children were writing letters. They told us the letters were for their pen-pals in Japan. The letters were sent via electronic mail and the children expected to receive answers the next day. One child asked us why they were called "pen-pals." After we explained, another child added "it's like why we say 'dial the phone' – it's left over from the old days."

Speech Synthesis For A Blind Student

Later, we noticed a child wearing headphones attached to a small box next to a computer. The box was a speech synthesizer. At the push of a button, it would convert the text on the screen to speech. Ms. Lovelace told us that John has been blind since birth, but with the speech synthesizer, a special keyboard, and some other electronic devices, he is able to progress with his lessons very well. She emphasized that computers have been a tremendous help in educating children with all types of handicaps and in making it possible for handicapped children to work in regular classroom settings.

Many lessons were about computers themselves. Computer studies are a standard part of the curriculum. All the children learn how to control computers to permit creative work. For some, this consists of writing computer programs. One group of children was working on a math drill program to be used by younger children in the school. After testing it on some five-year-olds, they told us that it was "a neat program, but some of the instructions mixed up the little kids. It still has to be more user-friendly."

Other children used a computer to write music. The program allowed them to enter musical notation, listen to the music, alter its pitch and tempo, and change the notes. It was like a word processor for music. Their work was to be transmitted via a computer network, to be entered into a statewide computer music contest.

We also saw a group of three children working on a computer art project. Each child would take a turn adding something to the computer display by drawing on a board next to the computer. They simply outlined what they wanted to draw and it appeared on the screen. After something was placed on the screen, it could be easily colored, moved, rotated, made larger or smaller, or erased. With a great deal of animated debate (one of the teachers had to ask them to settle down) a picture was gradually taking form. Later, a large version was printed to hang on the classroom wall, and three small copies were printed for the artists to take home.

We asked the teachers about the discipline problems so prevalent in schools a few years ago. One teacher, who had been teaching for 20 years, said that many problems have been minimized since education had become truly individualized. Students and teachers feel less frustration and a greater sense of accomplishment since there is so much flexibility in the content and methods of teaching and learning.

Children with learning problems receive a great deal of specific help. Teachers have time for individual tutoring, while computers provide unlimited practice at a level and pace appropriate to

NEW! EDUCATIONAL SOFTWARE FOR CHILDREN

from MICROGRAMS, INC.

- Designed to supplement curriculum in Grades K-8
- Classroom tested
- Programmed with sound and advanced motivational graphics

Expect more from your PET®. . .

- Tutorial and Review Programs available for Math, Reading, Language, Science, Social Studies, Vocabulary, Spelling, and special interest areas
- Testing and Grading programs for Teachers

MICROGRAMS

INCORPORATED

P.O. BOX 2146, LOVES PARK, IL 61130

PHONE 815/965-2464

☐ Please send me a free catalog.

☐ Please send me a sample program and a free catalog. I have enclosed $2.00 for postage and handling.

NAME ______________________

ADDRESS ______________________

CITY ______________________

STATE ______________ ZIP ________

PET is a registered trademark of Commodore Business Machines.

each child. The problems that could lead to a child being labeled as "learning disabled" have been reduced. Debates among educators about such things as which is the best method of teaching reading have also decreased, since an optimal method can be used for each individual.

Is This Science Fiction?

Is Babbage School science fiction? Such a school doesn't exist today, but the technology to do everything we have mentioned does exist. We believe that Babbage School could be a reality within the next few years.

Will your school take advantage of computers and other technological innovations? The aim of our columns is to help you make good use of these new and powerful tools for teaching and learning. In each column, we will discuss a general issue about learning with computers, issues such as: what is computer literacy? How can computers facilitate the education of handicapped individuals? What training is required for teachers to make good use of computers?

We will also point out some articles, books, software, hardware, and sources of information you may find useful. Relevant to this column, there are many books about the influence of computers in the near future. We particularly recommend the following four:

1. *The Micro Millenium,* by Christopher Evans (Pocket Books, 1979). Discusses computers of the past, present, and future and their effects on society. Includes an account of the roles of Charles Babbage and Ada Lovelace in the history of computers.

2. *The Third Wave,* by Alvin Toffler (Bantam Books, 1980). Toffler's thesis, developed in some detail, is that our society is in the midst of a Computer Revolution, comparable in scope of its effects to the Agricultural Revolution (the first wave) and the Industrial Revolution (the second wave).

3. *The Electronic Cottage,* by Joseph Deken (William Morrow & Co., 1981). A wide-ranging discussion of things computers can do, how they work, and how they may change our everyday lives.

4. *Mindstorms: Children, Computers and Powerful Ideas,* by Seymour Papert (Basic Books, 1980). A detailed description of some computer-created environments for children to explore, and the effects on the children's understanding of mathematical concepts. ©

THE apple® GAZETTE

Apple Addresses

Bill Grimm
Mountain View, CA

The Apple II uses three types of addressing depending upon the language being used. Apple's machine language uses hexadecimal addresses in the range from $0000 to $FFFF. Its Floating Point BASIC language uses decimal addresses in the range from 0 to 65535. Its Integer BASIC uses decimal addresses in the range from 0 to 32767 to -32767 to -1. This means that, if you want to address a particular memory location, you must choose the correct address for the language you are using. Since I program in all three languages and my references are a mixture from all three, I needed an address cross-reference program. So I wrote "Apple Addresses."

"Apple Addresses" can be used "as is" to convert one language's address to another's, and to give the high and low byte values which need to be POKEd into a BASIC program to store that address. Alternatively, you could extract the subroutines in Apple Addresses which convert between hex and decimal numbers and insert them in your own program. See the last paragraph of this article for more details.

The program begins by asking the user which of the six possible conversions he would like to make. This is followed by a request to select the way the results of the conversions are to be displayed. There are four possible displays:

1. single conversions displayed on the monitor one at a time.

2. single conversions printed out on a Silentype printer* one at a time.

3. a range of conversions displayed on the monitor.

4. a range of conversions printed out on a Silentype printer*.

*With slight program modifications other printers could be used.

Subroutines

"Apple Addresses" makes extensive use of subroutines. This helps in organizing the program as well as making it shorter and easier to debug. The controlling or EXECutive routine is called Apple Addresses – Exec. It starts on line 100 and goes to line 310. Since a picture is worth a thousand words, I made what I call a *balloon diagram* (Figure 1) to show how data flows through the program. These are the conventions I used to make the diagram:

1. Each balloon represents a subroutine. The name of the subroutine and the line numbers where it is located are placed in the balloon.

2. Data flows through a subroutine in the direction of the arrows on the outside of the balloon.

3. Data flows between subroutines in the direction of the arrows on the *strings*.

4. If conditions are placed on what data flows through a subroutine, these conditions are written in along the *strings*.

As an additional aid for understanding how the program works I have included the following variable descriptions list:

A( ) — each A(I) holds the decimal equivalent value of the Ith hexadecimal numeral in the hex number being created from a decimal number — appropriate numbers are then added to convert these to ASCII codes.

A$( ) — holds the characters represented by the ASCII codes in A( ).

CHOICE — holds the number of the conversion chosen — see lines 120 to 178.

DVL — holds the decimal value of the number being converted — may be either FP or INT decimal.

DVL$ — is the string equivalent of DVL and is used in the output routines.

FLAG — if FLAG = 1 then an invalid number was entered and the program returns to get a new number.

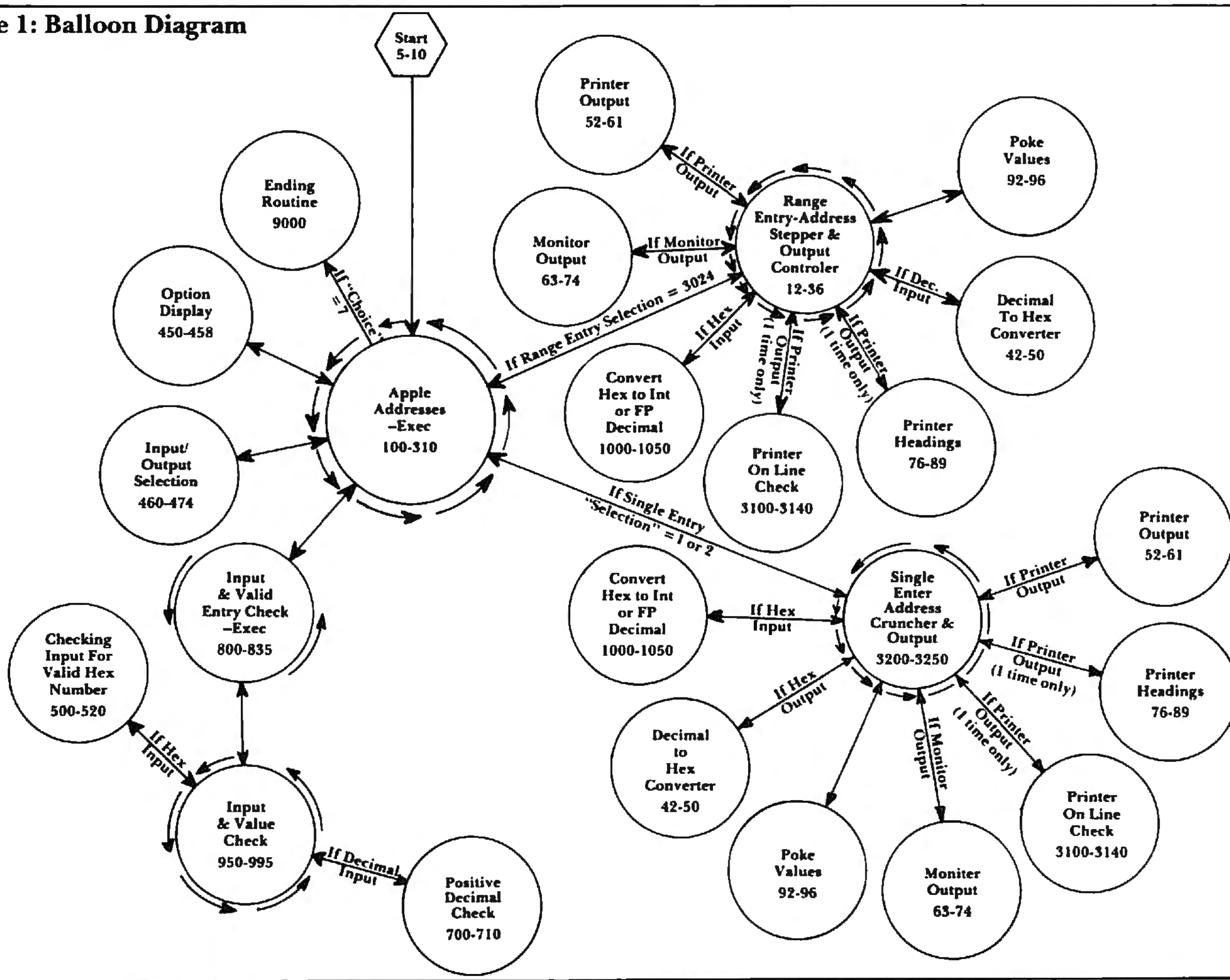
FRST — holds the FP BASIC address equivalent of the lowest address in the selected range.

FRST$ — holds the smallest address chosen — this address is then processed and stored in FRST.

HVL$ — holds the hex number selected or the hex number resulting from the conversion — if no hex numbers are involved then it holds the converted decimal number.

LST — holds the FP BASIC address equivalent

Figure 1: Balloon Diagram



of the largest address in the selected range.

LST$ — holds the largest address chosen — this address is then processed and stored in LST.

N — holds the decimal equivalent of each hex numeral in a hex number being converted to a decimal number.

PHI% — holds the number that would be POKEd into the high byte when placing the address into memory.

PLO% — holds the number that would be POKEd into the low byte when placing the address into memory.

POK — holds the address from which PLO% and PHI% are derived.

SELECT — holds the type of output selected — see lines 462 to 470.

STP — holds the positive decimal stepping interval chosen.

STP$ — holds the stepping interval chosen which is later changed and stored in STP.

TB — the horizontal tab value desired.

TN — holds the intermediate numbers of the decimal address that is being converted into a hex address. VTB — used to control the vertical tabbing of the monitor output.

Some Suggestions

I have found that the easiest way to debug a program while I am entering it is to first type in the EXEC program. Then, if I place return statements at all the branching locations, I can check the EXEC for bugs. Once the EXEC is free of bugs, I add one subroutine at a time in the order that the EXEC uses them, checking for bugs as I go.

If you have a need for subroutines which convert numbers from hex to decimal or from decimal to hex, two subroutines in this program may be of help. The first is called "decimal to hex converter" (lines 42 to 50). The input to this routine is TN which must hold a positive decimal number <65536. The output is HVL$ which holds the hex equivalent to the number in TN. The second is called "convert hex to INT or FP decimal" (lines 1000 to 1050). The input to this routine is HVL$ which must hold a hex number < = $FFFF and choice. If choice = 1 then you get the positive decimal equivalent. Otherwise you get Int BASIC's equivalent. The output is a decimal number in DVL. ©

NEW ROM BOARD FOR THE APPLE II*
$125.00 WITH UTILITY ROM.

With Andromeda's new ROM Board, you can plug many useful utility programs into your Apple II. Because ROM memory never forgets, you can access these utilities instantly without having to load them from disk.

The ROM Board comes with the utility ROM, which gives you five powerful options to apply to your Applesoft* programs. With the Utility ROM, you can do automatic line numbering, control a program list with a page mode, restore a crashed Applesoft* program in memory, alphabetize a disk catalogue and create a disk without DOS, giving you an extra 8K on your disk. Any of Soft Control Systems' other ROMS can be used, such as the Dual DOS in ROM, and Your'ple ROM.

You can install 2K PROMS, 4K PROMS, or even 2K RAM chips in each of the two memory sockets. So you can even have the Read - Write capability of RAM to develop PROM Programs yourself, or just have an extra 2K RAM for your machine - Language programs. Two 2732 PROMS allow a total of 8K of memory on the Board.

Now with One Year Warranty.

Don't forget the Andromeda 16K RAM Expansion Board $195.00

ANDROMEDA

INCORPORATED
P.O. Box 19144
Greensboro, NC. 27410
919 852-1482

Distributed By

COMPUTER DATA SERVICES

P.O. Box 696
Amherst, NH. 03031
603 673-7375

*Apple II and Applesoft are trademarks

More Apple Hi-Res Shape Writer

Chris Dupuy
Gonzales, LA

Countless hours spent plugging ones and zeros on graph paper are now history, thanks to Mr. Hennig's "Hi-Res Shape Writer." [**COMPUTE!** #14] Shload miseries are not missed and drawing shapes other than right angles are now a breeze.

After creating one star cruiser after another, I was soon struck with the harsh realization that I could not SAVE these cosmic creations on my cassette recorder. Unfortunately for me, I belong to the one percent club of Apple owners who cannot afford the luxuries of a disk drive. Undaunted with the PEEKs and POKEs ahead of me, I proceeded to write a routine that would put all the bytes from the shape table into trusty DATA statements.

The program is intended to be added to the original "Apple Hi-Res Shape Writer" by Doug Hennig. However, the routine used to POKE DATA in DATA statements can be adapted to other programs where the user does not want to be bothered with the rules of STOREing and RECALLing arrays.

Program Operation

REMark statements were omitted from the program in order to save valuable space, since memory size becomes a problem with complex shapes.

5–1084 Sets an array to the bytes POKEd into the shape table in original program.

13900–13906 Searches for the memory locations of the first blank DATA statement and sets Y equal to this.

13910–13970 POKEs Q to first item in DATA statement.

13930 Separates Q into individual digits.

13975 POKEs number of shape tables and reference numbers for shape tables.

14000–14075 POKEs bytes of shape table into the succeeding locations of the DATA statements.

14004 Searches DATA statement for a period (CHR$(46)), in order to find location to insert next value.

14550–14630 Demonstration program to verify information in DATA statements.

14572 Checks DATA statement to verify additional space on current statement. If not, then READ asterisks and jump to next DATA statement.

14700–14710 DELetes main portion of program and leaves demo program with DATA statements to be SAVEd.

15000–15005 DATA statements with 184 periods (quantity is at your discretion), and 4 asterisks.

Variables Used

Q Holds the number of bytes in the shape table.

V() Stores individual bytes of shape table.

Y Keeps track of the DATA statement memory locations.

R Used to check memory locations for a period.

F,FF Holds LENgth of strings and uses that value in FOR-NEXT statements.

T(),L() Arrays that hold the individual digits of bytes from shape table.

E$ User input.

X The location for bytes to be POKEd into shape table.

Y$ Stores the DATA being READ from demo program. String is used to prevent error message when asterisk is READ.

Hints And Changes

Those who have 32K Apples will encounter space problems when trying to run this longer program. DELeting the instructions, REMarks, disks subroutines, and combining statements will help avoid this obstacle.

Once all changes are made to your program, lines 13904 and 13906 may be DELeted. However, the memory location for the first DATA statement must be found. In machine language, the three bytes to look for are: 83 00 2E. The decimal location of 2E should then be set to Y in line 13900. Remember — if this change is done, no other changes can be made in the program (except for DATA statements), without the information being POKEd into the wrong locations. If searching for memory locations is too tedious, then you might want to experiment by raising the value in line 13900. Either one of these changes will save time in program execution.

Providing you have shaved off a good portion of the program, the value in line five may be raised to accomodate more complex shapes.

The major shortcoming in this program is the

inability to store more than one shape table at a time. Though a small amount of effort could change this, it would not be feasible if you are running low on memory. I hope this program brought some relief and enjoyment to you cassette owners out there.

```
5 DIM V(250)
1082 Q=Q+1
1084 V(Q)=X
13040 TEXT: HOME
13042 VTAB 10: HTAB 5
13045 PRINT "MEMORY LOCATIONS ARE BEING
      SCANNED"
13900 Y=3500
13904 IF PEEK(Y)=131 AND PEEK (Y+2)=46 THEN
      Y=Y+2: GOTO 13910
13906 Y=Y+1:GOTO 13904
13910 FF=LEN(STR$(Q))
13920 FOR X=1 TO FF
13930 T(X)=VAL(MID$(STR$(Q),X,1))
13940 POKE Y,T(X)+48
13945 Y=Y+1
13950 NEXT
13970 POKE Y,44
13975 POKE Y+1,49:POKE Y+2,44:POKE Y+3,48:POKE
      Y+4,44:POKE Y+5,52:POKE Y+6,44:POKE
      Y+7,48:POKE Y+8,44
13997 TEXT: HOME
13998 VTAB 10: HTAB 2
13999 PRINT "DATA IS NOW BEING POKED INTO
      MEMORY"
14000 FOR QQ=1 TO Q
14003 R=PEEK (Y)
14004 IF R<>46 THEN Y=Y+1: GOTO 14003
14005 F=LEN(STR$(V(QQ)))
14010 FOR T=1 TO F
14019 L(T)=VAL(MID$(STR$(V(QQ)),T,1))
14040 POKE Y,L(T)+48
14050 Y=Y+1
14055 NEXT
14060 POKE Y,44
14070 Y=Y+1
14075 NEXT
14100 HOME
14500 PRINT "TYPE 'ESC' KEY TO DEMONSTRATE
      PROGRAM"
14510 GET E$: IF E$<>CHR$(27) THEN END
14550 POKE 232,0: POKE 244,64
14555 READ Q
14560 FOR X=16384 TO 16387+Q
14570 READ Y$
14572 IF Y$="*" OR Y$="**" OR Y$="***" OR Y$=
      "****" THEN 14570
14575 Y=VAL(Y$)
14580 POKE X,Y
14590 NEXT
14600 POKE 16388+Q,0
14610 HGR: SCALE=1: ROT=0
14620 HCOLOR=3
14630 DRAW 1 AT 140,80
14700 VTAB 22
14702 PRINT "TYPE 'ESC' TO FORM NEW PROGRAM"
14704 GET E$: IF E$<>CHR$(27) THEN END
14705 TEXT: HOME
14706 PRINT "PROGRAM IS NOW READY TO BE
      SAVED"
14710 DEL 5,14510
15000 DATA . . . . . . . . . . . . . . . . . . . . . . . . .
      . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
      . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
      . . . . . . . . . . . . . . . . . . . . . . . . ****
15001 DATA . . . . . . . . . . . . . . . . . . . . . . . . .
      . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
      . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
      . . . . . . . . . . . . . . . . . . . . . . . . ****
15001 DATA . . . . . . . . . . . . . . . . . . . . . . . . .
      . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
      . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
      . . . . . . . . . . . . . . . . . . . . . . . . ****
15002 DATA . . . . . . . . . . . . . . . . . . . . . . . . .
      . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
      . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
      . . . . . . . . . . . . . . . . . . . . . . . . ****
15003 DATA . . . . . . . . . . . . . . . . . . . . . . . . .
      . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
      . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
      . . . . . . . . . . . . . . . . . . . . . . . . ****
15004 DATA . . . . . . . . . . . . . . . . . . . . . . . . .
      . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
      . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
      . . . . . . . . . . . . . . . . . . . . . . . . ****
15005 DATA . . . . . . . . . . . . . . . . . . . . . . . . .
      . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
      . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
      . . . . . . . . . . . . . . . . . . . . . . . . ****
```

©

Lower Case With Unmodified Apple

Joseph Wrubel
Aberdeen, NJ

This article describes a program called LC.EDIT which can be used to build, modify, and print text files using both upper and lower case letters on an unmodified 48K APPLE II Plus. The editor supports most of the commonly used edit commands including find, locate, change, append, insert and delete. Also included are read and write disk commands.

Uppercase letters are entered by preceding them with a CTRL-A. Internally, the program adds 32 to the ASCII value of each lower case letter, thus setting up the string for output to the printer. On the screen, capital letters are converted to the inverse mode while the lowercase letters are converted back to uppercase for display only.

I purchased my APPLE II early in December 1980, and quickly realized that the BASIC language had changed a lot since I had used it last in 1968. The biggest change I noticed was the string handling ability of the new BASIC.

The first application program I decided to write required the use of strings. I quickly found the "write text" and "read text" programs on the master disk and as quickly decided I didn't like them. At work, I make use of text editors on PRIME and UNIVAC computers and find that each of them has certain features which the other doesn't support. So I backed myself into writing a text editor for my APPLE and decided to incorporate the features I liked best from each system.

The program is used the first time to create a text file. The procedure is to hit a carriage return when prompted for "FILE NAME." This puts the program in the input mode. Once the text is entered, a CR puts the program into the EDIT mode. The options available in the EDIT mode are described below. Note that a single letter followed by a space and then any needed parameters is the usual format within the program. In this version, capital letters are typed in by preceding them with a CTRL-A.

The edit options are as follows:

I — Insert new line behind the present line.

C — Change the first sub-string to the second sub-string in this line of text. Sub-strings are separated by /'s. Double //'s can be used to enter a new substring in front of the existing string or to delete the last part of the original string.

A — Append new string to the end of the original string on this line.

P — Print a number of lines. Options include printing all lines from the present position to the end of the file by typing P*.

S — Save file. It is saved with its original name if one has been previously entered. Otherwise, a file name is requested via a prompt. If you give a file name when using S, the new name is used. This is a way of making an image of a text file for backup or modification.

N — If alone the next line is displayed. N +/– NUMB moves the pointer back and forth within file limits.

L — Locate sub-string at any location in any line from the present line to the end of the file.

Q — Quit. Normal program exit.

F — Find sub-string at beginning of any line from the present line to the end of file.

R — Retype present line completely.

H — Help if you have forgotten how to use the program. Can be used at any time.

E — Enter new file name to be edited. Can be used to edit when finished with the first without having to re-run the program.

NN — NN is any valid line number in the file. This is a direct line number access to the entire file.

The program is well REMarked to help any new programmer understand not only what the program does, but also how it does it.

The printer I have is an EPSON MX-80, but I believe this program will work for any printer which supports lower case characters. Until the day this article was written, I had no idea that I could take advantage of the printer's lower case abilities, but my son persisted. This program was modified from my original upper-case only version in about four hours.

One necessary feature of this program is the amount of user error-checking which takes place. As of this writing, I am unaware of any way to make the program bomb. Most of the checks were installed originally, but a few were added when bomb-outs indicated an unexpected pitfall such as typing "DELETE" instead of "D" to delete one line.

If anyone would like to save the effort of typing in the program send me a disk, $3, and an SASE mailer and I will provide a copy of this version and the upper-case only version. My mailing address is:

Joseph N. Wrubel
27 Norwood Lane
Aberdeen, NJ 07747

```
1  REM  **********************
2  REM  *                    *
3  REM  *  LC.EDIT PROGRAM   *
4  REM  *                    *
5  REM  *        BY          *
6  REM  *                    *
7  REM  *    J. N. WRUBEL    *
8  REM  *                    *
9  REM  *                    *
10  REM  *   REV. AUG 1981    *
11  REM  *                    *
12  REM  **********************
13  REM
15  HOME : DIM T$(500)
20  INPUT "FILE NAME :";Z$
24 D$ =  CHR$ (4): REM  CTRL-D
25  REM
26  REM ************************
27  REM  CR TO ENTER BUILD MODE
28  REM ************************
29  REM
30  IF  LEN (Z$) = 0 GOTO 1000
32  IF Z$ = "H" THEN 9400: REM LIST INSTRUCTIONS
34  REM
35  REM ************************
36  REM   LOAD FILE FROM DISK
37  REM ************************
38  REM
40  PRINT D$;"OPEN";Z$
50  PRINT D$;"READ";Z$
52  REM
53  REM ************************
54  REM  FIRST ELEMENT FROM DISK IS FILE LENGTH (NUMERIC)
55  REM
57  REM  REMAINDER OF FILE IS IN STRING FORMAT
58  REM ************************
59  REM
60  INPUT I
70  FOR J = 1 TO I
75  INPUT T$(J): NEXT
80  PRINT D$;"CLOSE";Z$
90 J = 1
100  REM ************************
101  REM  MAIN REENTRY POINT FROM MOST PROGRAM OPTIONS
102  REM ************************
103  REM
105  PRINT J;":";
110  GOSUB 250:R$ = T$
112  IF  LEN (T$) = 0 THEN W$ = "": GOTO 121
115 W$ =  CHR$ ( ASC ( LEFT$ (R$,1)) - 32)
116  REM
117  REM ************************
```

```
118  REM   CONVERT SINGLE LETTER ENTRY TO NUMERIC
119  REM **********************
120  REM
121  FOR W = 1 TO 13
130  IF W$ =  MID$ ("ICADPSNLQFRHE",W,1) THEN 190
140  NEXT
150  GOTO 1200: REM ENTRY WAS NOT A VALID LETTER
190  ON W GOTO 2000,3000,4000,5000,6000,7000,8000,9000,200,500,2500,9400
     ,750
200  PRINT "PROGRAM COMPLETE": END
210  GOTO 90: REM  CONTINUE RE-ENTRY
250  REM
251  REM **********************
252  REM INPUT STRING SUBROUTINE
253  REM **********************
254  REM
257 T$ = ""
260  GET A$: IF A$ =  CHR$ (13) THEN  PRINT A$;: RETURN
270  IF A$ =  CHR$ (8) AND  LEN (T$) > 1 THEN T$ =  LEFT$ (T$, LEN (T$) -
     1): PRINT A$;: GOTO 260
275  IF A$ =  CHR$ (8) THEN T$ = "": PRINT A$;: GOTO 260
280  IF A$ =  CHR$ (1) THEN  GET A$: INVERSE : PRINT A$;: NORMAL :T$ = T
     $ + A$: GOTO 260
290  IF A$ <  CHR$ (65) OR A$ >  CHR$ (90) THEN T$ = T$ + A$: PRINT A$;:
      GOTO 260
300  PRINT A$;:T$ = T$ +  CHR$ ( ASC (A$) + 32): GOTO 260
350  REM
351  REM **********************
352  REM PRINT A LINE SUBROUTINE
353  REM **********************
354  REM
360  FOR L = 1 TO  LEN (T$):B$ =  MID$ (T$,L,1)
370  IF B$ >  CHR$ (64) AND B$ <  CHR$ (91) THEN  INVERSE : PRINT B$;: NORMAL
     : GOTO 395
380  IF B$ >  CHR$ (96) AND B$ <  CHR$ (123) THEN B$ =  CHR$ ( ASC (B$) -
     32)
390  PRINT B$;
395  NEXT : PRINT : RETURN
450  REM
451  REM ***********************
452  REM STRING DECODE SUBROUTINE
453  REM ***********************
454  REM
460  FOR M = 3 TO  LEN (R$)
470  IF  MID$ (R$,M,1) >  CHR$ (95) AND  MID$ (R$,M,1) <  CHR$ (124) THEN
     R$ =  LEFT$ (R$,M - 1) +  CHR$ ( ASC ( MID$ (R$,M,1)) - 32) +  MID$
     (R$,M + 1)
480  NEXT : RETURN
500  REM ********************
501  REM  FIND STRING ROUTINE
502  REM ********************
503  REM
510  IF  LEN (R$) < 3 THEN 580
520 F$ =  MID$ (R$,3): REM  STRING TO BE FOUND
530  FOR K = J + 1 TO I
540  IF  LEFT$ (T$(K), LEN (F$)) = F$ THEN 570
550  NEXT
560  PRINT "NO FIND": GOTO 90
570 J = K: GOTO 6300
```

```
580  PRINT "YOU MUST ENTER STRING": GOTO 100
750  REM
751  REM *********************
752  REM  ENTER NEW FILE NAME
753  REM *********************
754  REM
755  HOME
760  IF  LEN (R$) < 3 THEN 20
770  GOSUB 450:Z$ =  MID$ (R$,3): GOTO 25
999  REM
1000  REM *******************
1001  REM   BUILD FILE MODE
1002  REM *******************
1003  REM
1005 I = 0:J = 0
1007  PRINT "INPUT"
1010  PRINT J + 1;":";
1020  GOSUB 250:T$(J + 1) = T$
1030  IF  LEN (T$(J + 1)) = 0 GOTO 1100
1040 J = J + 1:I = I + 1
1050  GOTO 1010
1090  REM
1091  REM ******************
1092  REM  ENTER EDIT MODE
1093  REM ******************
1094  REM
1100  PRINT "EDIT": GOTO 100
1200  REM
1201  REM **********************
1202  REM CR TO ENTER INPUT MODE
1203  REM **********************
1204  REM
1205  IF  LEN (R$) = 0 THEN 1500
1206  REM
1207  REM  *********************
1208  REM  VALIDATE LINE POINTER
1209  REM  *********************
1210 W =  VAL (R$)
1215  IF W < 1 OR W > I GOTO 1240
1220 J = W
1230 T$ = T$(J): GOSUB 350: GOTO 100
1240  PRINT "ILLEGAL ENTRY": GOTO 100
1500  REM
1501  REM  ********************
1502  REM       INPUT MODE
1503  REM  ********************
1504  REM
1505  REM  IF AT END OF FILE DO EASY WAY
1507  IF J = I GOTO 1007
1509  REM  THE HARD WAY
1510  PRINT "INPUT"
1515  PRINT J + 1;":";
1520  GOSUB 250
1530  IF  LEN (T$) = 0 GOTO 1100: REM   RETURN TO EDIT MODE
1540  FOR K = I TO J STEP  - 1
1550 T$(K + 1) = T$(K)
1560  NEXT
1570 T$(J + 1) = T$
1580 J = J + 1:I = I + 1
```

```
1590  GOTO 1515
2000  REM
2001  REM *********************
2002  REM   INSERT NEW LINE
2003  REM *********************
2004  REM
2005  IF  LEN (R$) < 3 THEN  PRINT "BAD I": GOTO 100
2010 I = I + 1
2020  FOR K = I - 1 TO J STEP  - 1
2030 T$(K + 1) = T$(K)
2040  NEXT
2050 T$(J + 1) =  MID$ (R$,3)
2060 J = J + 1
2070  GOTO 100
2500  REM
2501  REM *******************
2502  REM      RETYPE LINE
2503  REM *******************
2504  REM
2505  IF  LEN (R$) < 3 THEN  PRINT "BAD R": GOTO 100
2510 T$(J) =  MID$ (R$,3)
2520  GOTO 100
3000  REM
3001  REM *********************
3002  REM  CHANGE PART OF LINE
3003  REM *********************
3004  REM
3005  IF  LEN (R$) < 3 THEN  PRINT "BAD C": GOTO 100
3010 W$ =  MID$ (R$,3)
3020  IF  LEFT$ (W$,1) <  > "/" OR  RIGHT$ (W$,1) <  > "/" THEN 3060
3030  FOR K = 2 TO  LEN (W$) - 1
3040  IF  MID$ (W$,K,1) = "/" GOTO 3070
3050  NEXT
3060  PRINT "MISSING DELIMITERS": GOTO 100
3070 F$ =  MID$ (W$,2,K - 2)
3075 H =  LEN (T$(J))
3080  FOR M = 1 TO H
3090  IF  MID$ (T$(J),M,K - 2) = F$ GOTO 3120
3100  NEXT
3110  PRINT "NO FIND": GOTO 100
3120 G$ =  MID$ (W$,K + 1, LEN (W$) - K - 1)
3125  IF H - M + 1 -  LEN (F$) = 0 GOTO 3160
3127  IF K = 2 GOTO 3170
3128  IF M = 1 GOTO 3190
3130 T$(J) =  LEFT$ (T$(J),M - 1) + G$ +  RIGHT$ (T$(J),H - M + 1 -  LEN
     (F$))
3140  GOTO 6300
3160 T$(J) =  LEFT$ (T$(J),M - 1) + G$: GOTO 3140
3170 T$(J) =  MID$ (W$,3, LEN (W$) - 3) + T$(J): GOTO 3140
3190 T$(J) = G$ +  RIGHT$ (T$(J),H - M + 1 -  LEN (F$)): GOTO 3140
4000  REM
4001  REM **********************
4002  REM APPEND TO PRESENT LINE
4003  REM **********************
4004  REM
4005  IF  LEN (R$) < 3 THEN  PRINT "BAD A": GOTO 100
4010 T$(J) = T$(J) +  MID$ (R$,3)
4020  GOTO 6300
5000  REM
```

```
5001  REM ********************
5002  REM    DELETE LINE(S)
5003  REM ********************
5004  REM
5007 L =  LEN (R$)
5010  IF L > 1 GOTO 5050
5012  REM A "D" ALONE DELETES ONE LINE ONLY
5020  FOR K = J TO I
5030 T$(K) = T$(K + 1): NEXT
5040 I = I - 1:J = J - 1: GOTO 100
5050  IF L = 2 GOTO 5110
5055 N =  VAL ( MID$ (R$,3))
5060  IF N > I - J + 1 THEN 5100
5065  IF N = 0 THEN  PRINT "BAD D": GOTO 100
5070  FOR K = J TO I - N
5080 T$(K) = T$(K + N): NEXT
5090 J = J - 1:I = I - N: GOTO 100
5100  PRINT "DELETE TOO BIG": GOTO 100
5110  PRINT "ILLEGAL DELETE": GOTO 100
6000  REM
6001  REM *******************
6002  REM  PRINT SOME LINES
6003  REM *******************
6004  REM
6007  IF  LEN (R$) < 2 THEN 6300
6010 NUM$ =  MID$ (R$,2)
6020  IF NUM$ = "*" GOTO 6150
6030 NUM =  VAL (NUM$)
6035  IF NUM = 0 THEN T$ = T$(J): GOSUB 350: GOTO 100
6040  FOR K = J TO J + NUM - 1
6050 T$ = T$(J): GOSUB 350:J = J + 1
6060  IF J > I GOTO 6100
6070  NEXT
6075 J = J - 1
6080  GOTO 100
6100  PRINT "EOF:";I;" LINES"
6104  REM
6105  REM  THE END OF FILE WAS FOUND
6106  REM
6110  GOTO 90
6150  REM
6151  REM ********************
6152  REM  IS PRINTOUT WANTED
6153  REM ********************
6154  REM
6160  PRINT : INPUT "PRINTOUT?";PR$
6170  IF  LEFT$ (PR$,1) = "Y" THEN 6350
6180  IF  LEFT$ (PR$,1) = "N" THEN 6200
6190  PRINT : PRINT "TRY AGAIN": GOTO 6160
6200  FOR K = J TO I
6210 T$ = T$(K): GOSUB 350: NEXT
6220  GOTO 6100
6300 T$ = T$(J): GOSUB 350: GOTO 100
6350  REM
6351  REM *********************
6352  REM   PRINT ENTIRE FILE
6353  REM *********************
6354  REM
6360  PRINT D$;"PR#1": PRINT  CHR$ (9);"80N"
```

```
6370  FOR K = J TO I
6374  REM  IF PERIOD SKIP A LINE
6375  IF T$(K) = "." THEN  PRINT : GOTO 6385
6378  IF  LEFT$ (T$(K),4) = ".    " THEN T$(K) = " " +  MID$ (T$(K),2)
6380  PRINT T$(K)
6385  NEXT
6390  PRINT D$;"PR#0": GOTO 6100
7000  REM
7001  REM ******************
7002  REM      SAVE FILE
7003  REM ******************
7004  REM
7006  IF  LEN (R$) > 2 THEN  GOSUB 450:Z$ =  MID$ (R$,3)
7008  IF  LEN (Z$) <  > 0 THEN 7015
7010  PRINT : INPUT "FILE NAME ?";Z$
7012  IF  LEN (Z$) = 0 THEN 7010
7015  PRINT D$;"OPEN";Z$
7020  PRINT D$;"DELETE";Z$
7030  PRINT D$;"OPEN";Z$
7040  PRINT D$;"WRITE";Z$
7050  PRINT I
7060  FOR J = 1 TO I
7070  PRINT T$(J): NEXT
7080  PRINT D$;"CLOSE";Z$
7090  GOTO 90
8000  REM
8001  REM ********************
8002  REM   RELATIVE MOVEMENT OF POINTER
8003  REM ********************
8004  REM
8005  IF R$ <  > CHR$ (110) THEN 8030: REM   A TRANSLATED "N"
8010 J = J + 1
8015  IF J > I THEN  PRINT "EOF:";I;" LINES": GOTO 90
8020 T$ = T$(J): GOSUB 350: GOTO 100
8030 V =  VAL ( MID$ (R$,2))
8040  IF V + J > I OR V + J < 1 GOTO 8100
8050 J = J + V
8060 T$ = T$(J): GOSUB 350: GOTO 100
8100  PRINT "MOVE TOO BIG": GOTO 100
9000  REM
9001  REM *********************
9002  REM    LOCATE STRING
9003  REM *********************
9004  REM
9007  IF  LEN (R$) < 3 THEN  PRINT "BAD L": GOTO 100
9010 F$ =  MID$ (R$,3)
9020  FOR K = J + 1 TO I
9030  FOR M = 1 TO  LEN (T$(K)) -  LEN (F$) + 1
9040  IF F$ =  MID$ (T$(K),M, LEN (F$)) GOTO 9070
9050  NEXT M: NEXT K
9060  PRINT "NO FIND": GOTO 90
9070 J = K: GOTO 6300
9400  REM
9401  REM *********************
9402  REM        HELP USER
9403  REM *********************
9404  REM
9405  HOME
9407  PRINT : PRINT  SPC( 9);"TEXT EDITING PROGRAM"
```

```
9410  PRINT : PRINT "EACH SINGLE CHARACTER INSTRUCTION SHOWN"
9415  PRINT "BELOW IS TO BE FOLLOWED BY A SPACE AND"
9420  PRINT "AND THEN ANY NEEDED PARAMETERS."
9425  PRINT : PRINT "TO START A NEW FILE, PUSH RETURN WHEN"
9430  PRINT "YOU ARE PROMPTED FOR THE FILE NAME."
9435  PRINT "YOU MAY THEN ENTER YOUR TEXT FILE LINE
9440  PRINT "BY LINE.  WHEN DONE, PUSH RETURN AGAIN"
9445  PRINT "TO ENTER THE EDIT MODE.
9450  PRINT : PRINT  SPC( 4);"** PUSH ANY KEY TO CONTINUE **"
9460  GET G$
9505  HOME
9510  VTAB 2: HTAB 10
9515  PRINT "TEXT EDITING PROGRAM"
9520  PRINT : PRINT "CODE      FUNCTION"
9525  PRINT : PRINT " I        INSERT NEW LINE OF TEXT"
9527  PRINT " ","BEHIND THE PRESENT LINE"
9530  PRINT : PRINT " C        CHANGE THE FIRST STRING TO "
9535  PRINT  SPC( 9),"THE SECOND, USE /'S TO"
9540  PRINT " ","SEPARATE STRINGS"
9545  PRINT : PRINT " A"; SPC( 7);"APPEND STRING TO END OF LINE"
9550  PRINT " ","LEAVE 1 SPACE BETWEEN"
9555  PRINT " ","THE A AND THE STRING"
9560  PRINT : PRINT " D"; SPC( 7);"DELETE 'N' LINES, IF N OMITTED,";
9565  PRINT " ","JUST THIS LINE IS DONE"
9570  PRINT : PRINT " P"; SPC( 7);"PRINT 'N' LINES FROM HERE"
9575  PRINT " ","USE P* TO LIST ALL"
9580  PRINT : PRINT  SPC( 6);"** PUSH ANY KEY TO CONTINUE **"
9585  GET G$
9590  HOME : PRINT : PRINT "CODE      FUNCTION"
9595  PRINT : PRINT " S"; SPC( 7);"SAVE FILE WITH NAME ENTERED"
9600  PRINT " ","IF NO NAME IS ENTERED"
9605  PRINT " ","USE ORIGINAL FILE NAME"
9610  PRINT : PRINT " N"; SPC( 7);"NEXT LINE +/- NUMB IS PRINTED"
9615  PRINT : PRINT " L"; SPC( 7);"LOCATE STRING FROM HERE"
9620  PRINT " ","TO END OF FILE"
9625  PRINT : PRINT " Q"; SPC( 7);"QUIT"
9630  PRINT : PRINT " F"; SPC( 7);"FIND STRING AT START OF ANY"
9635  PRINT " ","LINE FROM HERE TO END"
9640  PRINT : PRINT " R"; SPC( 7);"RETYPE PRESENT LINE"
9645  PRINT : PRINT " H"; SPC( 7);"HELP PROVIDED VIA THIS LIST"
9650  PRINT : PRINT  SPC( 7);"** PUSH ANY KEY TO CONTINUE **"
9651  GET G$
9653  HOME : PRINT : PRINT "CODE      FUNCTION"
9655  PRINT : PRINT " E"; SPC( 7);"NAME FILE TO BE EDITED"
9660  PRINT : PRINT "(CR)"; SPC( 5);"USE CARRIAGE RETURN TO
9665  PRINT  SPC( 9);"ENTER INPUT MODE"
9690  PRINT : PRINT : HTAB 5: PRINT "** PUSH ANY KEY TO CONTINUE **"
9695  GET G$: GOTO 100
```

©

COMPUTE! OVERVIEW:

Individual Tax Plan

The "Individual Tax Plan" program by Aardvark Software, Incorporated is a highly sophisticated piece of computer software for the Apple computer system (II or Plus) with at least 48K of RAM and two disk drives, DOS 3.3 or PASCAL. It also nicely lends itself to the computerist who, in essence, does not have a working knowledge of computers. As long as the manual is at least previewed, one will not have any trouble running this program.

It is a well-designed, easy to use system for comparing different filing alternatives in order to minimize the income tax liability for an individual taxpayer. It does an effective job of allowing a comparison of numerous different tax preparation schemes at one time. It does not, however, do all of the work and calculations necessary to complete a tax return. Perhaps a better name for the software package would have been "Individual Tax Comparison Scheme."

Up to five alternative tax preparation schemes may be entered at once. One alternative, for example, could include income averaging with schedule G while others could compare filing jointly vs. filing singly for a married couple. Side-by-side comparison of the calculated taxes for each of the alternatives is effectively done by the program. The program is only of value, however, after an individual has calculated many of the numbers that belong on the tax return. For example, tax credits is a single item to be entered. The taxpayer (or tax return preparer) must determine the tax credits for child care expenses and energy-saving expenses (each a percentage of actual expenses and each subject to dollar limitations and other limiting factors), and add them together. This sum is the value that is entered into the "Individual Tax Plan."

It should be stressed that this program is not oriented towards layman use, but towards the tax professional, who has had previous tax preparation exposure. To effectively use this powerful tool one must have a working knowledge of possible tax alternatives to pursue.

Updates

Should changes in federal tax law occur in a calendar year, Aardvark Software will make available revised programs reflecting these changes. Revisions will cost $50.00 and can be obtained from local Aardvark Software dealers. Annual updates reflecting changes in tax law and including program enhancements will be made available on or before November 1st of each calendar year.

Back-up copies of the included program and data disks are allowed using the standard Apple copy program. You should be able to save between 50 and 75 Tax Plan cases on each copy of the data diskette.

Using The Program

During operation of the program the user enters data for up to 74 categories, such as filing status, interest, charitable contributions, and "long term capital gains-post 6/8/81." Unfortunately, the documentation does not follow the program exactly in the identification of the different categories. Items 12 through 32 are misidentified, most of the numbers being off by one. Once the changes are marked on two of the four pages which identify the various categories, there is no difficulty finding the various items, but the problem should never have occurred.

For each category a value can be independently entered for each alternative, or programming options can be used to calculate values for different alternatives. For example, if $10,000 is entered for the first filing alternative, then the remaining alternatives are calculated by the program at 20% increments by simply entering "P20" for percent-20. Other options include "X" if only the next alternative is to be calculated on a percentage basis or "I" for "increment" if all subsequent alternatives are to differ from each other by a specific dollar amount.

After all of the data is entered, the program takes a few seconds to calculate the taxes for all of the alternatives. Any two alternatives (in any order) may be printed as hard copy for easy comparison of the alternatives in different columns. In a strange departure from the easy to use options, here "999" must be entered to indicate that the numbers of all of the desired options have selected. RETURN would have been far easier to use.

Flexibility

One of the strengths of this software package is the ease with which a user can move from one part of the program to another. From a main menu single digit numbers are used to reach further menus which identify specific activities. Several options are offered for moving from category to category for data entry. To "select" a specific category "S" can be typed followed by the number of the category. To move "forward" to the next category "F" is used and "B" is used to "back up." For many of the categories up to ten numbers can be entered — five for the taxpayer and five for the spouse. The program is smart enough to fill up all of the alternatives with the value given for the first alternative unless it is specifically given new values for subsequent alternatives. To move from one specific alternative to another "U" is used to go "up" and "D" is used to go "down." No control keys are required here — the editing is very easy to use.

ESCape can be used at nearly any time to exit from data entry and save on disk all of the values that have been entered for all of the alternative schemes. One minor irritant here is that the Pascal volume numbers are used to specify the disk drives. The documentation explains that disk drive #1 needs to be specified as volume four and so on, but the program should have been written to accept simple drive numbers. The name that is given for the file is first checked against those currently on the disk in order to prevent inadvertent over writing of a file that should be maintained. An option is also provided to see the directory of items that have been stored.

There are no charts included to indicate which of the 74 possible tax input questions are to be entered if, for example one were filing "married with a joint return." A glossary of terms would also be a welcome addition. However, execution speed is an outstanding feature of this program. All calculations are performed in under 60 seconds, regardless of complexity. The program appears to be written entirely in machine code, which would account for its exceptional speed.

While the ranges of input data appear to be sufficiently checked, disk error codes are vaguely defined. If RESET is pressed, all existing data not saved on disk is lost and the program requires rebooting for continued operation. This can be most annoying and could possibly prove fatal if done during a disk storage operation.

The Documentation

Documentation for the individual tax plan program consists of an attractive 3-ring binder with a 31 page illustrated instruction manual which includes a simple appendix and printouts. The documentation, although sufficient for the tax professional, is not designed to be a comprehensive overview of tax preparation for the layman.

With the exception of the misnumbered categories, the documentation is clear and complete. About ten pages are used to lead the user through two simple examples that do a good job of demonstrating how to move about in the program. Sample printed output for each of these examples is given in an Appendix (misidentified as Appendix "B"). About ten more pages are used to specifically describe the program options and to identify the various categories for data entry. Throughout the instructions, 27 photographs of screen images appear. The photographs were apparently taken with a wide angle lens and therefore appear distorted, but they are readable and provide an accurate representation of what the program displays.

General Overview

Panelist #1: "Negative and detracting hindrances:

(1) There should be a subroutine within the program which would enable the user to enter directly into a mini-directory to review a directive or procedure.

(2) The ability to only do the filing status routine should be looped so that only an individual taxpayer entry is verified and utilized when there is no spouse involved.

(3) Provision to exercise the use of only one disk drive should be available when only one is involved or necessary.

(4) An ending directive within the program (other than in the manual) should be provided after all statistics have been entered.

(5) A 'short form' alternative option could be incorporated.

Positive and useful aspects:

(1) Exceedingly fast access time.

(2) Ease of use in the main menu parameters.

(3) Printer parameters and linefeed status changes.

(4) Aardvark's updating procedures)annually or when the tax structure/laws change)

(5) Comparative analysis of defined numerical statistics to take advantage of the lowest tax amount to be paid.

(6) The 'step' feature: accessing forward and backward through the program via a single keystroke.

(7) Ability to access any part of the program by entering the input of the area and return.

(8) User defined changes: save data (Y/N), screen or printer display, program user return (ability to re-enter your numerical statistics and make any changes necessary in any of the alternative figures prior to executing the calculations).

(9) Ability to handle positive and negative integers as well as figuring out its compound percent.

(10) User ability to make any and all necessary backup copies in the event of catastrophes."

Panelist #2: "The software is easy to use and effectively compares calculations done on the basis of different tax preparation schemes. It does not do all of the calculations that a taxpayer needs to do, nor does it identify a correspondence between specific line numbers on form 1040 and the categories within the program. The software package could be very useful for professional tax preparers, but is not likely to be worth the expense for an ordinary taxpayer. For someone with substantial capital gains to declare, it could be helpful, but that person is probably going to benefit from advice from a professional anyway. Whom should you select as that professional? Someone who has an Apple and Aardvark's Individual Tax Plan."

Panelist #3: "This program was designed by a group of CPA's with over 17 years of "Big Eight" experience to meet the needs of the professional tax practitioner.

This program is not, nor was it designed to be, everyone's answer to H & R Block. With some additional documentation, a much wider range of people could benefit from it. While not intended for the layman, the professional tax preparer should find this program an outstanding value."

Sample Output

Table 1.

```
1981                              ALTERNATIVE
                                       1
FILING STATUS                        JOINT
EXEMPTIONS                               2
WAGES, SALARIES                     28,480
INTEREST AFTER EXCLUSION               350
DIVIDENDS AFTER EXCLUSION                0
CAPITAL GAIN/LOSS                        0
PARTNERSHIP INCOME/LOSS                  0
OTHER INCOME/LOSS                    2,000
                                 ---------
TOTAL INCOME                        30,830
ADJUSTMENTS TO INCOME                1,600
                                 ---------
ADJUSTED GROSS INCOME               29,230

DEDUCTIONS
MEDICAL & DENTAL EXPENSES              170
STATE & LOCAL INC TAXES              1,681
OTHER TAXES                              0
INTEREST EXPENSE                     1,690
CHARITABLE CONTRIBUTIONS               943
CASUALTY LOSS                        1,090
MISCELLANEOUS                          787
                                 ---------
TOTAL DEDUCTIONS                     6,361
ZERO BRACKET AMOUNT                  3,400
                                 ---------
EXCESS ITEM. DEDUCTIONS              2,961
```

TAX TABLE INCOME	26,269
EXEMPTIONS TIMES $1,000	** N/A
TAXABLE INCOME	** N/A
TAX - TAX TABLES/XYZ	4,359
TAX - QUAL. CAP. GAINS	** N/A
TAX - INCOME AVERAGING	*
TAX - MAXIMUM TAX	** N/A
TAX SELECTED	4,359
ADDITIONAL TAXES	0
GROSS REGULAR TAX	4,359
CREDITS	0
NET REGULAR TAX	4,359
MINIMUM TAX	0
ALTERNATIVE MINIMUM TAX	0
OTHER TAXES	0
TOTAL TAX LIABILITY	4,359
FEDERAL PAYMENTS	4,998
BALANCE DUE (REFUND)	-639

Individual Tax Plan. Aardvark Software, Inc., 783 North Water Street, Milwaukee, WI 53202. 48K Apple, two disk drives, DOS 3.3 or Pascal, $250. ©

THE MOSAIC 32K RAM FOR ATARI

THE BEST:

✓ DESIGN
✓ MATERIALS
✓ SAFETY
✓ GUARANTEE

THE ONLY CHOICE

You own the best micro computer available, the Atari*. At Mosaic we've made Atari computers our only business and have made our products the best anywhere. You've seen the advantage of having a 32K RAM board. The Mosaic 32K RAM is the only board designed to meet your needs now and in the future too. It has designed-in flexibility to be compatible with products available in the near future. See for yourself, Mosaic is the only choice for the serious Atari owner.

THE BEST SCREEN CLARITY

Here's what A.N.A.L.O.G. magazine had to say: "Mosaic uses, what we feel are the highest quality components, which may improve reliability." and "The Mosaic showed no sign of interference and gave the best screen clarity."

THE MOSAIC ADVANTAGE	MOSAIC 32K RAM	OTHER 32K RAMS
Works in both Atari 400 & 800	■	■
Gold edge connectors for better reliability	■	■
Fits Atari 400 without modification	■	■
Custom components for better performance & reliability	■	
Highest quality components for the best screen clarity	■	
Full year warranty	■	
Designed to take advantage of Atari 800's superior bus structure.	■	
Can be used with 8K, 16K and future products.	■	
Allows Atari 800 to have 2 slots for future expansion	■	
Designed so there's no danger of damaging your computer	■	
Designed for inter-board communication in Atari 800	■	
Easy to follow instructions for simpler no-solder installation in Atari 400	■	
Available companion board ($5) to allow running 32K board independent of other boards	■	
Full flexible memory configuration	■	

Atari 800 Memory Configuration	with the MOSAIC 32K RAM	with other 32K Boards	Atari 800 Memory Configuration	with the MOSAIC 32K RAM	with other 32K Boards
Empty 32K 16K	48K RAM 40K With BASIC Cartridge	48K RAM 40K With BASIC Cartridge	Empty 8K 32K	40K RAM	**Danger!** This Configuration Can Damage Computer
Empty 32K 8K	40K RAM	**Danger!** This Configuration Can Damage Computer	8K 32K 8K	48K RAM 40K With BASIC Cartridge	**Danger!** This Configuration Can Damage Computer
Empty 16K 32K	48K RAM 40K With BASIC Cartridge	**Danger!** This Configuration Can Damage Computer	Empty Companion 32K	32K RAM	**Danger!** This Configuration Can Damage Computer

Now from your nearest Mosaic dealer **$179.95**

Direct from Mosaic $189.95

MOSAIC
ELECTRONICS
P.O. Box 748 Oregon City, Oregon 97045 503/655-9574

*Trade Mark of Atari, Inc.

THE ATARI® GAZETTE

Cryptogram

Ronald and Lynn Marcuse
Freehold, NJ

Word games are just one of many applications which can be programmed on the versatile ATARI computers. While not challenging your reflexes as does Space Invaders or Star Raiders, they do challenge your mind. With the number of graphic action games approaching infinity in our house, we predicted that our two school-age offspring will be competent space shuttle pilots by the time they reach 18. Unfortunately, they may not be able to read. An educational computer word game may be capable of swinging the pendulum the other way, or at least slow down the onrushing invaders.

Most of the electronic word games currently being marketed are variations of the "hangman" game, where you are required to guess an unknown word by specifying its letter content within a certain number of tries. If you fail, you are then "hanged" or punished in some similar manner. A *cryptogram*, however, challenges your ability to decipher coded phrases or messages. Not wanting to generate an "unbreakable" code, we used a simple letter substitution. Each letter in the statement is replaced uniformly by some other alphabetic character. For example, all of the A's may appear as G's, the B's as Y's, etc. Don't bother to memorize these relationships because the structure of the code changes each game. Spaces between words and punctuation remain intact.

There are two skill levels in the game, selectable through the OPTION key. With the first, the program will decipher the vowels for you, leaving you only the consonents to decode. With the second option you are on your own. We had originally programmed three options, the third being a compression of the statement into one long string (removal of spaces), but deciphering of the phrase became rather difficult. The SELECT key is utilized to vary the number of participants. For the two person version, one player would enter a statement for the other to decode. In the one player game, the computer will randomly select one of fifty popular (?) expressions stored in the program. These phrases appear as the DATA statements in lines 1010-1500. You may change this list if you desire. Just make sure that you wind up with exactly 50 statements and that each one is no longer than 75 characters (including spaces and punctuation). You may use any punctuation with the exception of the comma.

How To Play

A game in progress may be saved to either disk or tape. Program 1 contains the disk version. Program 2 displays changes required for the recorder. In saving the game, the disk version will request a three character (or less) name which will be appended to the file name. The tape version will require you to insert a blank tape in the recorder. Make a note of the recorder counter. The procedure for loading a saved game is similar, but you must supply either the file name extender or tape that has been positioned (using the counter) in the recorder.

To start a game type "N" to the program prompt "SHOULD I LOAD A SAVED GAME" and then pick the skill level and number of players by pressing the OPTION and SELECT keys. Press the START key to begin. You may need to depress the keys for a second or two to register your action. This may be speeded up by shortening the timing loop at line 990. If you had chosen the two player option, the program will prompt you to enter a phrase or message to be encoded. This must be from 20 to 75 characters in length. Shorter phrases are actually harder to decipher than long ones. In the one player option, the program will randomly select one of the fifty DATA statements.

After the encoding process is completed (it takes approximately 15 seconds to generate the code and substitute the letters), the game screen is displayed. At the top of the display is a table showing the code letters and values that you have assigned to the code. The next group of lines contain the "secret phrase" and your working translation. These alternate if the phrase is longer than one line in length. If skill level 1 had been selected, the vowels would have already been translated for you. At the bottom of the display is the input area for code letters and values. Enter a code letter and then the substitution you would like to perform. An arrow cursor alternates between the two input lines. To erase a previous entry, first type the code letter and then press the space bar.

When you have correctly substituted all of the characters, the program will notify you graphically. You may also press the ESCAPE key to end the game. This will allow you to save the game, quit, or try a new phrase. If you are short on RAM (under 24K), the REMARK statements may be omitted with no ill effects.

Lines 18 through 30 comprise the "house-keeping" section of the program. The left screen margin is set to 1 (POKE 82,1) for those TV sets that overscan, the keyboard is OPENed and the variables are DIMensioned here. The alphabet is stored in A$, the substitution code in B$, and the table entries for the game display in T$, P$, C$, and Q$ are the actual phrase, the coded phrase, and the working translation, respectively. The array X (with 26 elements) is used by the code generation routine.

If a saved game is being reloaded (prompt in line 40), the data is input and control is sent directly to the main game display at line 400. Otherwise, the variables are cleared (lines 80-90) and the option screen is generated (starting at line 100). Memory location 53279 is the register used to read the console keys on the ATARI computer. The address is first cleared by POKE 53279,8 and then queried by PEEKing at it in the loop from lines 120 through 180. We are concerned with the binary value that is stored in that address.

The START key is assigned to bit 0, the SELECT key to bit 1, and the OPTION key to bit 3. A value of zero in the bit position means that the key was pressed. For example, if the START key is hit, the SELECT and OPTION keys would register decimal values of 2 and 4 in their respective bit positions. The START key would return a zero in the low order bit, giving a total of 6 (decimal). Likewise, the SELECT key would equal a decimal 5 (4 + 1) and the OPTION key would be 3 (2 + 1).

If the two player option was selected, the phrase would be input in line 220, otherwise the program will randomly select one of the fifty data statements in line 240. In lines 250 through 290, the program generates the substitution code. A random letter (from A$) is selected and, if that element of the X array is still set to zero, the B$ sub-string position is equated to the letter. The array is used to check off letters that have already been used. This type of algorithm could easily be expanded to a card shuffling routine if you prefer poker to word games.

The substitution of the code letters into the phrase is done in lines 300 to 380. If skill level 1 was selected (SK = 1) then the ATASCII value of the phrase letter is checked to see if it is a vowel (values of 1,5,9,15, and 21). If it is, the letter is moved into the translation line Q$, otherwise the character "–" occupies that position. The program must also count backwards from 38 looking for the first space to break the line on.

The game board is displayed in lines 400-430 and the input of code letters and substitutions is performed in lines 500-520. After the data is received, the modified table elements are redisplayed in line 530 and the revised translation line in 540-560. If the translation is the same as the phrase (line 560), you are sent to the winners circle at line 700, otherwise you go back to 510 for more data. Pressing the ESCAPE key (an ATASCII value of 27) would cause a jump to line 800 for your exit options. The POP statement in line 915 is necessary to reset the stack pointer for the non-RETURN exit out of the subroutine.

The remainder of the program is routines for the winning and losing displays, input and printing of data, the exit options, and the saving of games in progress. The variables saved, either on disk or tape, are P$, C$, and Q$ (the original phrase, the coded phrase the the current translation), T$ (the assignment table), and the lengths of the phrase (L) and its first line segment (L1). For the disk version of the program, the format of the saved game is D:CRYPTG. + the 3 character name that was entered.

Program 1.

```
10 REM ********    CRYPTO-GRAM    ******
*******
11 REM *
      *
12 REM *  RONALD & LYNN MARCUSE, FREEHOL
D NJ  *
13 REM *
      *
14 REM ********************************
*******
15 REM
18 POKE 82,1:OPEN #4,4,0,"K:"
20 DIM A$(26),B$(26),T$(26),I$(1),D$(1),
P$(81),C$(80),Q$(80),X(26)
25 DIM N$(3),F$(12)
30 A$="ABCDEFGHIJKLMNOPQRSTUVWXYZ":GOSUB
 900:R=40
36 REM
37 REM LINES 40 THRU 70 ALLOW THE LOADIN
G OF A SAVED GAME ON DISK
38 REM SEE LISTING 2 FOR TAPE VERSION
39 REM
40 ? "{DOWN} SHOULD I LOAD A SAVED GAME
(Y/N) ";
45 GOSUB 970:IF I$<>"Y" THEN 80
50 GOSUB 960:TRAP 70:OPEN #2,4,0,F$:INPU
T #2,P$
```

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

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 **$5.95**

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! **$14.95**

#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. **$14.95**

#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 **$14.95**

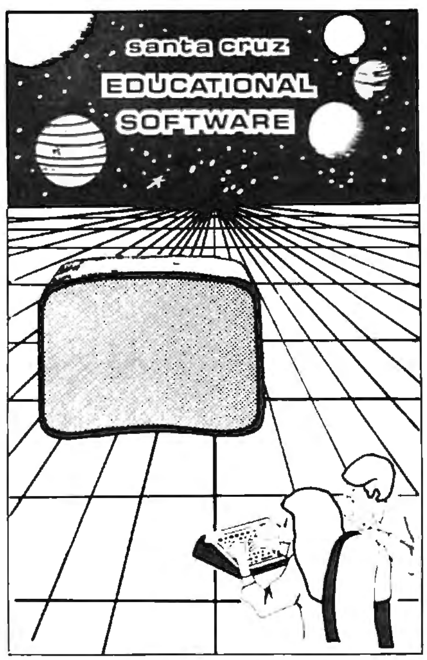
#4: BASICS OF ANIMATION — Shows you how to animate simple shapes using the PRINT and PLOT commands. and also has nice little PLAYER/MISSILE Graphics demo to learn This would be an excellent way to start making your programs come alive on the screen Recommended for new owners. **$14.95**

#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 **$14.95**

Tricky Tutorials (except #5) require 16K memory for cassette orders and 24K for disk. The price is $14.95 each. You may order 1,2,3, & 4 for $49.95! All six in a colorful binder cost $89.95.

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 **$14.95**

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 keeeping 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

***Include $2.00 postage ($.75 for Memory Map)**
***In California, include 6.5% tax**

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

```
60 INPUT #2,C$:INPUT #2,Q$:INPUT #2,T$:I
NPUT #2,L:INPUT #2,L1
65 CLOSE #2:XIO 33,#2,0,0,F$:TRAP 40000:
GOTO 400
70 CLOSE #2:? "{TAB}!DISK ERROR!":GOSUB
990:GOTO R
80 FOR I=1 TO 26:X(I)=0:T$(I)="_":NEXT I
:PL=1:SK=1:RESTORE
90 FOR I=1 TO 80 STEP 10:C$(I)="
  ":Q$(I)="          ":NEXT I
97 REM
98 REM OPTION SELECTION MENU
99 REM
100 GRAPHICS 17:SETCOLOR 0,3,10:SETCOLOR
 4,3,2
105 POSITION 4,2:? #6;"CRYPTO-GRAM":POKE
 53279,8
110 POSITION 3,5:? #6;"* * * * * * *"
120 POSITION 2,11:? #6;"SKILL LEVEL - ";
SK
130 POSITION 1,14:? #6;"# OF PLAYERS - "
;PL
140 POSITION 4,20:? #6;"PRESS START":? #
6;"     TO BEGIN"
150 GOSUB 990:A=PEEK(53279):IF A=6 THEN
200
160 IF A=5 THEN PL=PL+1:IF PL>2 THEN PL=
1
170 IF A=3 THEN SK=SK+1:IF SK>2 THEN SK=
1
180 GOTO 120
200 GOSUB 900:ON PL GOTO 240,210
207 REM
208 REM TWO PLAYER OPTION
209 REM
210 ? "{2 DOWN} ENTER PHRASE (20 TO 75 C
HARACTERS){DOWN}"
220 INPUT P$:IF LEN(P$)<20 OR LEN(P$)>75
 THEN 210
230 GOTO 250
237 REM
238 REM ONE PLAYER OPTION, COMPUTER PICK
S RANDOM PHRASE
239 REM
240 J=INT(RND(0)*50)+1:FOR I=1 TO J:READ
 P$:NEXT I
247 REM
248 REM ALPHABETIC SUBSTITUTION CODE GEN
ERATED
249 REM
250 GOSUB 910:? "{2 DOWN}   PLEASE WAIT
WHILE I GENERATE"
260 ? "{DOWN}{TAB}AN ! UNBREAKABLE ! COD
E":FOR I=1 TO 26
270 J=INT(RND(0)*26)+1:IF X(J)=1 THEN 27
0
280 IF SK=1 THEN IF I=1 OR I=5 OR I=9 OR
 I=15 OR I=21 THEN T$(J,J)=A$(I,I)
290 B$(I,I)=A$(J,J):X(J)=1:NEXT I
297 REM
298 REM CHARACTERS IN PHRASE SUBSTITUTED
 WITH CODE LETTERS
299 REM
300 L=LEN(P$):FOR I=1 TO L:J=0:I$=P$(I,I
):IF I$=" " THEN 360
310 J=J+1:IF J>26 THEN C$(I)=I$:Q$(I)=I$
:GOTO 360
320 IF I$<>A$(J,J) THEN 310
340 C$(I)=B$(J,J):Q$(I)="_"
350 IF SK=1 THEN IF J=1 OR J=5 OR J=9 OR
 J=15 OR J=21 THEN Q$(I)=I$
360 NEXT I
365 L1=L:IF L<38 THEN 400
367 REM
368 REM FIRST LINE SPACING MEASURED
369 REM
370 FOR I=38 TO 18 STEP -1:IF C$(I,I)="
" THEN L1=I:GOTO 400
380 NEXT I
397 REM
398 REM MAIN GAME BOARD DISPLAYED
399 REM
400 GOSUB 910:POSITION 2,3:? "CD/LT":FOR
 N=1 TO 26:I$=A$(N,N)
410 POSITION 3+INT(N/5)*6,3+N-INT(N/5)*5
:? I$;" ";T$(N,N):NEXT N
420 POSITION 1,9:? C$(1,L1):IF L>L1 THEN
 POSITION 1,13:? C$(L1+1)
430 GOSUB 950
497 REM
498 REM PROMPTS FOR INPUT OF CODE AND LE
TTER
499 REM
500 POSITION 3,20:? "ENTER CODE LETTER":
? "{DOWN}{TAB}    AND VALUE"
510 K=20:GOSUB 920:N=A-64:D$=CHR$(A)
520 K=22:GOSUB 920:T$(N,N)=CHR$(A):IF A=
32 THEN T$(N,N)="_"
530 POSITION 5+INT(N/5)*6,3+N-INT(N/5)*5
:? T$(N,N)
540 FOR I=1 TO L:IF C$(I,I)=D$ THEN Q$(I
,I)=T$(N,N)
550 NEXT I
560 GOSUB 950:IF P$=Q$ THEN 700
590 POSITION 24,22:? " ":GOTO 510
697 REM
698 REM WINNER SCREEN DISPLAYED
699 REM
700 FOR J=0 TO 14 STEP 2:GRAPHICS 18:SET
COLOR 4,J,2:POSITION 3,5
710 ? #6;"* * * * * * *":? #6;"   *  COR
RECT  *":? #6;"  * * * * * * *"
```

```
720 FOR K=0 TO 250 STEP 10:SOUND 0,K,10,
15:NEXT K
730 GOSUB 900:GOSUB 950
740 FOR K=250 TO 0 STEP -10:SOUND 0,K,10
,15:NEXT K
750 NEXT J:SOUND 0,0,0,0
760 GOSUB 900:GOSUB 950:? "{2 DOWN}{TAB}
PLAY AGAIN (Y/N) ";
PLAY AGAIN (Y/N) ";
765 GOSUB 970:IF I$="Y" THEN 80
770 GOTO 880
797 REM
798 REM END OF GAME (NO WINNER) OPTIONS
799 REM
800 GOSUB 910:? "{DOWN}{TAB}OPTIONS ARE:
"
810 ? "{DOWN}{TAB}S - SAVE GAME":? "{DOW
N}{TAB}Q - QUIT":? "{DOWN}{TAB}P - PLAY
AGAIN"
820 ? "{DOWN}{TAB}SELECT => ";:GOSUB 970
:IF I$="S" THEN 850
830 IF I$="Q" THEN GOSUB 980:GOTO 880
840 IF I$="P" THEN GOSUB 980:GOTO 80
845 GOTO 800
847 REM
848 REM GAME IS SAVED TO DISK, SEE LISTI
NG 2 FOR TAPE VERSION
849 REM
850 R=800:GOSUB 960:TRAP 70:OPEN #2,8,0,
F$:TRAP 40000
870 ? #2;P$:? #2;C$:? #2;Q$:? #2;T$:? #2
;L:? #2;L1:CLOSE #2
880 GRAPHICS 0:END
897 REM
898 REM PRINT CRYPTO-GRAM TITLE
899 REM
900 GRAPHICS 0:SETCOLOR 1,3,10:SETCOLOR
2,3,2:POKE 752,1
910 ? "{CLEAR}{DOWN}{2 TAB}{2 LEFT}CRYPT
O-GRAM":RETURN
917 REM
918 REM CHARACTER INPUT EDITING
919 REM
920 POSITION 21,K:? "==> {LEFT}";:GET #4
,A
925 IF A=27 THEN POP :GOTO 800
930 IF A=32 AND K=22 THEN 940
935 IF A<65 OR A>90 THEN 920
940 POSITION 21,K:? "   ";CHR$(A):RETURN

947 REM
948 REM PRINT ANSWER
949 REM
950 POSITION 1,11:? Q$(1,L1):IF L>L1 THE
N POSITION 1,15:? Q$(L1+1)
955 RETURN
957 REM
958 REM DISK FILE SAVED WITH PERSONS NAM
E (3 CHAR)
960 ? "{DOWN} ENTER 3 LETTERS OF NAME =>
 ";:INPUT N$:F$="D:CRYPTG."
965 F$(10)=N$:RETURN
967 REM
968 REM GET AND PRINT CHARACTERS
969 REM
970 GET #4,A:I$=CHR$(A):? I$:RETURN
977 REM
978 REM LOOSER DISPLAY SCREEN
979 REM
980 ? "{3 UP}{TAB}THE ANSWER WAS:":Q$=P$
:GOSUB 950:? "{2 DOWN}{TAB}I LOOK HOW EA
SY IT WAS !"
985 FOR I=10 TO 250:SOUND 0,I,10,I/20+2:
NEXT I:SOUND 0,0,0,0:RETURN
987 REM
988 REM DELAY LOOP
989 REM
990 FOR I=1 TO 100:NEXT I:RETURN
997 REM
998 REM THE STORED PHRASES FOLLOW, MAXIM
UM OF 50 ALLOWED
999 REM EACH MUST BE UNDER 75 CHARACTERS
 LONG
1000 REM
1010 DATA A STITCH IN TIME SAVES NINE
1020 DATA EARLY TO BED AND EARLY TO RISE
 MAKES A MAN HEALTHY WEALTHY AND WISE
1030 DATA THE EARLY BIRD CATCHES THE WOR
M
1040 DATA DO UNTO OTHERS AS YOU WOULD HA
VE OTHERS DO UNTO YOU
1050 DATA PLOP PLOP FIZZ FIZZ OH WHAT A
RELIEF IT IS
1060 DATA A LONG TIME AGO IN A GALAXY FA
R FAR AWAY
1070 DATA WHY DID THE CHICKEN CROSS THE
ROAD?
1080 DATA TO BE OR NOT TO BE. THAT IS TH
E QUESTION
1090 DATA THOU SHALT NOT COVET THY NEIGH
BORS WIFE
1100 DATA MAY THE FORCE BE WITH YOU
1110 DATA BEGINNERS ALL-PURPOSE SYMBOLIC
 INSTRUCTION CODE
1120 DATA WE THE PEOPLE OF THE UNITED ST
ATES OF AMERICA
1130 DATA DON'T FIRE UNTIL YOU SEE THE W
HITES OF THEIR EYES
1140 DATA YOU CAN FOOL SOME OF THE PEOPL
E ALL OF THE TIME
1150 DATA PROGRESS IS OUR MOST IMPORTANT
 PRODUCT
```

```
1160 DATA ONE SMALL STEP FOR MAN; ONE GI
ANT STEP FOR MANKIND
1170 DATA EVERY ACTION HAS AN EQUAL AND
OPPOSITE REACTION
1180 DATA I HAVE NOT YET BEGUN TO FIGHT
1190 DATA FASTER THAN A BULLET; MORE POW
ERFUL THAN A SPEEDING LOCOMOTIVE
1200 DATA WHO WAS THAT MASKED MAN
1210 DATA THEIRS NOT TO REASON WHY; THEI
RS BUT DO OR DIE
1220 DATA TO ERR IS HUMAN. TO FORGIVE DI
VINE
1230 DATA A LITTLE LEARNING IS A DANGERO
US THING
1240 DATA HE'D FLY THROUGH THE AIR WITH
THE GREATEST OF EASE
1250 DATA LAUGH AND THE WORLD LAUGHS WIT
H YOU; CRY AND YOU CRY ALONE
1260 DATA MARRIED IN HASTE; WE MAY REPEN
T AT LEISURE
1270 DATA O CAPTAIN! MY CAPTAIN! OUR FEA
RFUL TRIP IS DONE
1280 DATA THESE ARE THE TIMES THAT TRY M
EN'S SOULS
1290 DATA TIGER! TIGER! BURNING BRIGHT I
N THE FORESTS OF THE NIGHT
1300 DATA TALL OAKS FROM LITTLE ACORNS G
ROW
1310 DATA I THINK THAT I SHALL NEVER SEE
 A POEM LOVELY AS A TREE
1320 DATA FOR FOOLS RUSH IN WHERE ANGELS
 FEAR TO TREAD
1330 DATA WHEN IN ROME DO AS THE ROMANS
DO
1340 DATA FRIENDS ROMANS COUNTRYMEN LEND
 ME YOUR EARS
1350 DATA FROM THE HALLS OF MONTEZUMA TO
 THE SHORES OF TRIPOLI
1360 DATA IN FOURTEEN HUNDRED NINETY TWO
 COLUMBUS SAILED THE OCEAN BLUE
1370 DATA MARY HAD A LITTLE LAMB ITS FLE
ECE AS WHITE AS SNOW
1380 DATA I SHOT AN ARROW INTO THE AIR;
IT FELL TO EARTH I KNEW NOT WHERE
1390 DATA IF AT FIRST YOU DON'T SUCCEED
TRY TRY AGAIN
1400 DATA HE THAT FIGHTS AND RUNS AWAY M
AY TURN AND FIGHT ANOTHER DAY
1410 DATA IT TAKES A HEAP OF LIVING IN A
 HOUSE TO MAKE IT HOME
1420 DATA OH WHAT FUN IT IS TO RIDE IN A
 ONE HORSE OPEN SLEIGH
1430 DATA IT WAS A ONE-EYED ONE-HORNED F
LYING PURPLE PEOPLE EATER
1440 DATA I.CAN'T BELIEVE I ATE THE WHOL
E THING
1450 DATA SAY THE SECRET WORD AND WIN A
HUNDRED DOLLARS
1460 DATA DAMN THE TORPEDOES; FULL SPEED
 AHEAD
1470 DATA TWAS THE NIGHT BEFORE CHRISTMA
S AND ALL THROUGH THE HOUSE
1480 DATA HEY MISTER TAMBOURINE MAN PLAY
 A SONG FOR ME
1490 DATA EVERYTHING THAT GOES UP MUST C
OME DOWN
1500 DATA HICKORY DICKORY DOCK THE MOUSE
 RAN UP THE CLOCK
```

Program 2.

```
15 REM FOR TAPE VERSION, USE THESE
   LINES INSTEAD
16 REM YOU MAY ALSO DELETE LINES
   960-965
17 REM
50 TRAP 70:OPEN #2,4,0,"C:":INPUT
   #2,P$
65 CLOSE #2:TRAP 40000:GOTO 400
70 CLOSE #2:? "[B]tape error":GOSUB
   990:GOTO R
850 R=800:TRAP 70:OPEN #2,8,0,"C:":
    TRAP 40000
```

©

SuperFont

Charles Brannon
Editorial Assistant

The ability to redefine the character set is one of the more useful features of the Atari. In a previous article, "Character Generation on the Atari," (**COMPUTE!** #9) I explained the principles and techniques of customizing the character set. Basically, it involves the plotting of a character on an eight by eight matrix and then converting each row into a binary number.

This process, however, is slow and tedious for the programmer. Fortunately, it is an obvious candidate for computerization. The computer could display a grid, let you set and clear points on it, and then do the binary-to-decimal conversion for you. It could also let you save and load completed *fonts* (character sets) from tape or disk.

Although SuperFont may lack some of the features of commercial products, it is quite powerful and versatile. SuperFont is written in BASIC, but what makes it special is that it has several machine language subroutines as well. One of these (thanks to DLI) enables the redefined character set to be displayed on the screen at the same time as the regular one. This permits you to see the effects of your changes without letting the command menu or prompts turn into starships.

SuperFont uses player/missile graphics for fast updates and a colorful grid. Since the special character window is set off in a different color than the rest of the screen (again via DLI's), you get eight different colors to delight the eye. The human interface is enhanced with the use of a joystick to plot points in the eight by eight grid.

SuperFont has 18 commands:

E:Edit	**R:Restore**
F:Copy From	**X:Switch**
T:Copy To	**C:Clear**
O:Overlay	**I:Invert**
S:Save Font	**L:Load Font**
:Delete	**:Insert**
:Scroll Left	**:Scroll Right**
W:Write Data	**Q:Quit**
:Reverse	**G:Graphics**

This menu is displayed on the screen along with a "checkerboard" plotting grid, the 128 characters of the character set, and the 128 characters of the alternate character set. Some commands require you to select a character. A cursor will be placed on each of the character sets. You can move the cursors around the sets simultaneously. When the cursor is on the desired character, press the fire button to indicate it. An explanation of each command follows:

Edit: The basic editing command. The selected character is copied into the grid and a flashing cursor is homed into the grid. You move the cursor with the joystick. Pressing fire will set a point (if a point is clear) or reset (clear) a point (if a point is already set). You can draw lines by holding down the button while moving the joystick. Any changes are immediately visible in the character set and the character displayed in GRAPHICS mode one and two lines at the bottom of the screen. To completely redesign a character, use the Clear command, and then design the character from scratch.

Restore: This command will "fix" a character by copying the original bit pattern into it. Very useful if you have mangled a character or changed the wrong one.

Copy From: You select a character which is copied into the current character. The grid is updated, and you can further edit the character.

Copy To: The current character is copied to (replaces) the indicated character.

Switch: Exchanges the current character's bit pattern with the selected character.

Overlay: The selected character is overlaid upon the current character. This lets you combine two characters to form a new one.

Clear: Clears out the current character. For creating unique characters.

Invert: Turns the current character "upside down." For example a re-defined M could be inverted and copied to the W.

Save Font: Saves the alternate character set in compact form with a machine-language routine. Answer "Filename?" with either C: or D:filespec. If you see an error message, press a key to return to the menu.

Load Font: Retrieves a character set from tape or disk. Answer the "Filename" prompt as you did in Save Font.

Cursor-up or SHIFT-DELETE: Similar to Delete Line in BASIC. The line of dots the cursor rests on is deleted; the following lines are pulled up to fill the gap.

Cursor-down or SHIFT-INSERT: Similar to Insert Line in BASIC. A blank line is inserted at the cursor position. The bottom line is lost.

Scroll left: The bit pattern of the character is shifted to the left.

Scroll right: The bit pattern of the character is

shifted to the right.

Write Data: The internal code (0-127) of the current character is printed in reverse-field followed by the eight bytes (in decimal) of the character. If you want a printout of the entire character set, use the auxiliary program CHPRINT (Program 3). Pressing any menu selection key will erase the nine bytes.

Graphics: Toggles the TEXT/GRAPHICS option of the GRAPHICS mode one and two lines to let you see each half of the character set.

Reverse: Puts the character in reverse field: all dots become blanks, and all blanks become dots. Reverse field versions of the characters are not normally stored in the character set, but you may want this for special graphics, such as reverse-field text in GRAPHICS modes one or two.

Quit: Exits program.

The commands offer flexibility in working with character sets, but there may be other functions you may want to add. The program is modular in structure; just follow the branching IF statements after line 790 to 1370 and replace the 520 (IF K<> ASC("G") THEN 520) with a link to your additional command(s). You may also want to change the colors. Besides the SETCOLOR statements in line 170, change the zero in line 300 (POKE 1538,0) to COLOR (0-15)*16 + LUMINANCE (0-14). Similarly, you can play with the player/missile colors in line 360.

It is also possible to use the character set data on tape or disk directly. It is written as a series of 1024 bytes: the bytes of the character set — no more, no less. I have included two extra utility programs which access the character data. Program 2 simply loads the set into memory and changes CHBASE (756) to point to it. Program 3 produces a formatted hex or decimal dump of the character set. Both programs should have the "filespec" changed to the filename of your character set.

The code of the main program is fairly straight forward. It uses several machine language subroutines: (1) A Display List Interrupt handler to maintain the special character window. (2) Copies the ROM character table into the RAM CHSET table (avoids the 15 second delay in BASIC). (3) A LOGIC subroutine that permits AND, OR, EOR (and any other 6502 function such as ROR) to be used on a binary level (see also "Make Your Atari a Bit Wiser," **COMPUTE!** #12, p. 74). (4) Implements a fast machine language memory save thanks to the IOCB PUTREC and GETREC commands.

You can do a lot with this capability: custom fonts (Greek, "Computeristic," script), graphics characters (special line drawing characters, spaceships, "invaders," bombs, tanks, planes, ships, even little people! (INTRUDER ALERT! INTRUDER ALERT!) SuperFont makes your task easier, even fun!

Program 1.

```
100 REM |  *** SuperFont ***       |
110 REM | 11/10/81 Charles Brannon |
120 REM |  Character Set Editor    |
130 REM
140 DIM I(7),FN$(14),N$(3)
150 IF PEEK(1536)=0 THEN GOSUB 1400
160 GRAPHICS 0:POKE 752,1
170 SETCOLOR 2,7,2:SETCOLOR 4,7,2
180 DL=PEEK(560)+256*PEEK(561)+4
190 SD=PEEK(88)+256*PEEK(89)+13*40:ASD=S
D+5*40
200 A1=1630:FUNC=1631:A2=1632:LOGIC=1628

210 RAM=PEEK(106)-8:PMBASE=RAM*256
220 CHRORG=57344
230 POKE 559,46:POKE 54279,RAM
240 POKE 53277,3:POKE 53256,3
250 CHSET=(RAM-8)*256
260 POKE DL+23,6:POKE DL+24,7
270 POKE DL+18,130
280 POKE 512,0:POKE 513,6
290 POKE 54286,192
300 POKE 1549,RAM-8:POKE 1538,0
310 A=USR(1555,CHSET)
320 P0=PMBASE+512+20:P1=PMBASE+640+20:P2
=PMBASE+768+20:P=PMBASE+896+20:T=85
330 FOR I=0 TO 7:FOR J=0 TO 3:T=255-T:PO
KE P0+I*4+J,0:POKE P1+I*4+J,T:T=255-T
340 POKE P2+I*4+J,T:NEXT J:T=255-T:NEXT
I
350 POKE 53248,64:POKE 53249,64:POKE 532
50,64
360 POKE 704,198:POKE 705,240:POKE 706,6
8
370 POKE 53256,3:POKE 53257,3:POKE 53258
,3:POKE 623,1
380 ? " {Q}{8 R}{E}":FOR I=1 TO 8:? " {=
}        {=}":NEXT I:? " {Z}{8 R}{C}"
390 POKE 82,14:POSITION 14,1
400 ? "|E| Edit          |R| Restore"
410 ? "|F| Copy From     |X| Switch"
420 ? "|T| Copy To       |C| Clear"
430 ? "|O| Overlay       |I| Invert"
440 ? "|S| Save Font     |L| Load Font"
450 ? "{ESC}{DEL LINE} Delete      {ESC}
{INS LINE} Insert"
460 ? "{ESC}{CLR TAB} Scroll Left {ESC}{
SET TAB} Scroll{DOWN}{4 LEFT}Right"
```

```
470 ? "|W| Write Data   |Q|uit"
480 ? "{DOWN}|{F}{=}{G}| Reverse    |G| G
raphics"
490 FOR I=0 TO 3:FOR J=0 TO 31:POKE SD+J
+I*40+4,I*32+J:POKE ASD+J+I*40+4,I*32+J:
NEXT J:NEXT I
500 POKE 82,2:POSITION 0,0
510 OPEN #2,4,0,"K:"
520 P=PEEK(764):IF P=255 THEN 520
530 IF P=60 THEN 520
540 IF P=39 THEN POKE 764,168
550 GET #2,K
560 IF K<>ASC("E") THEN 790
570 GOSUB 1750
580 FOR I=0 TO 7:A=PEEK(CHSET+C*8+I):FOR
 J=0 TO 3:POKE P0+I*4+J,A:NEXT J:NEXT I
590 POKE ASD+169,C:POKE ASD+190,C
600 JX=0:JY=0
610 POSITION JX+4,JY+1
620 ? CHR$(32+128*FF);"{LEFT}";:FF=1-FF
630 IF STRIG(0)=0 THEN 750
640 IF PEEK(764)<255 THEN ? " ";:GOTO 52
0
650 ST=STICK(0):IF ST=15 THEN 620
660 IF STRIG(0) THEN FOR I=0 TO 100 STEP
 20:SOUND 0,100-I,10,8:NEXT I
670 POSITION JX+4,JY+1:? " ";
680 JX=JX+(ST=7)-(ST=11)
690 JY=JY+(ST=13)-(ST=14)
700 IF JX<0 THEN JX=7
710 IF JX>7 THEN JX=0
720 IF JY<0 THEN JY=7
730 IF JY>7 THEN JY=0
740 GOTO 610
750 POKE A1,PEEK(CHSET+C*8+JY):POKE A2,2
^(7-JX):POKE FUNC,73:A=USR(LOGIC)
760 POKE CHSET+C*8+JY,A:FOR J=0 TO 3:POK
E P0+JY*4+J,A:NEXT J
770 FOR I=1 TO 10:SOUND 0,I*4,8,8:NEXT I
:SOUND 0,0,0,0
780 GOTO 650
790 IF K<>ASC("F") THEN 830
800 S=C:GOSUB 1750
810 FOR I=0 TO 7:A=PEEK(CHSET+C*8+I):POK
E CHSET+S*8+I,A:NEXT I
820 C=S:GOTO 580
830 IF K<>ASC("T") THEN 870
840 S=C:GOSUB 1750
850 FOR I=0 TO 7:A=PEEK(CHSET+S*8+I):POK
E CHSET+C*8+I,A:NEXT I
860 C=S:GOTO 600
870 IF K<>ASC("O") THEN 920
880 S=C:GOSUB 1750
890 FOR I=0 TO 7:POKE A1,PEEK(CHSET+C*8+
I):POKE A2,PEEK(CHSET+S*8+I):POKE FUNC,9
:A=USR(LOGIC)
900 POKE CHSET+S*8+I,A:NEXT I
910 C=S:GOTO 580
920 IF K<>ASC("R") THEN 940
930 FOR I=0 TO 7:POKE CHSET+C*8+I,PEEK(C
HRORG+C*8+I):NEXT I:GOTO 580
940 IF K<>ASC("C") THEN 960
950 FOR I=0 TO 7:POKE CHSET+C*8+I,0:NEXT
 I:GOTO 580
960 IF K<>ASC("{R}") THEN 980
970 FOR I=0 TO 7:POKE CHSET+C*8+I,255-PE
EK(CHSET+C*8+I):NEXT I:GOTO 580
980 IF K<>ASC("X") THEN 1010
990 S=C:GOSUB 1750
1000 FOR I=0 TO 7:A=PEEK(CHSET+S*8+I):PO
KE CHSET+S*8+I,PEEK(CHSET+C*8+1):POKE CH
SET+C*8+I,A:NEXT I:GOTO 580
1010 IF K<>ASC("I") THEN 1030
1020 FOR I=0 TO 7:I(I)=PEEK(CHSET+C*8+I)
:NEXT I:FOR I=0 TO 7:POKE CHSET+C*8+I,I(
7-I):NEXT I:GOTO 580
1030 IF K<>ASC("{UP}") AND K<>ASC("{DEL
LINE}") THEN 1050
1040 FOR I=JY TO 6:POKE CHSET+C*8+I,PEEK
(CHSET+C*8+I+1):NEXT I:POKE CHSET+C*8+7,
0:GOTO 580
1050 IF K<>ASC("{DOWN}") AND K<>ASC("{IN
S LINE}") THEN 1070
1060 FOR I=7 TO JY STEP -1:POKE CHSET+C*
8+I,PEEK(CHSET+C*8+I-1):NEXT I:POKE CHSE
T+C*8+JY,0:GOTO 580
1070 IF K<>ASC("{LEFT}") THEN 1100
1080 FOR I=0 TO 7:A=PEEK(CHSET+C*8+I)*2:
IF A>255 THEN A=A-256
1090 POKE CHSET+C*8+I,A:NEXT I:GOTO 580
1100 IF K<>ASC("{RIGHT}") THEN 1130
1110 FOR I=0 TO 7:A=PEEK(CHSET+C*8+I)/2
1120 POKE CHSET+C*8+I,A:NEXT I:GOTO 580
1130 IF K<>ASC("Q") THEN 1150
1140 POKE 53248,0:POKE 53249,0:POKE 5325
0,0:POKE 53277,0:GRAPHICS 0:END
1150 IF K<>ASC("S") THEN 1210
1160 GOSUB 1610:POKE 195,0
1170 TRAP 1190:OPEN #1,8,0,FN$
1180 A=USR(1589,CHSET)
1190 CLOSE #1:TRAP 40000:IF PEEK(195) TH
EN 1260
1200 POKE 54286,192:GOTO 580
1210 IF K<>ASC("L") THEN 1290
1220 GOSUB 1610:POKE 195,0
1230 TRAP 1250:OPEN #1,4,0,FN$
1240 A=USR(1619,CHSET)
1250 CLOSE #1:TRAP 40000:IF PEEK(195)=0
THEN 1200
1260 POSITION 14,0:? "{BELL}* ERROR -";P
EEK(195);" *"
1270 IF PEEK(764)<255 THEN POSITION 14,0
```

```
:? "                          ":GOTO 1200
1280 GOTO 1270
1290 IF K<>ASC("W") THEN 1370
1300 POSITION 2,10:N$="    ":L=LEN(STR$(C
)):N$(1,L)=STR$(C):L=LEN(N$)
1310 FOR I=1 TO L:? CHR$(ASC(N$(I,I))+12
8);:NEXT I:? ">";
1320 FOR I=0 TO 2:FOR J=0 TO 1+(I>0):A=P
EEK(CHSET+C*8+J+I*3)
1330 SOUND 0,(I*3+J)*10+50,10,8
1340 PRINT A;",";:NEXT J:? "{BACK S}":NE
XT I:SOUND 0,0,0,0
1350 IF PEEK(764)=255 THEN 1350
1360 POSITION 2,10:FOR I=1 TO 3:? "
        ":NEXT I:GOTO 520
1370 IF K<>ASC("G") THEN 520
1380 CF=1-CF:POKE 1549,RAM-8+2*CF
1390 GOTO 520
1400 GRAPHICS 2+16:SETCOLOR 4,1,4:POSITI
ON 5,3:? #6;"SUPER|FONT|"
1410 POSITION 4,5:? #6;"patience{3 N}":P
OSITION 2,7:? #6;"|charles brannon|"
1420 FOR I=1536 TO 1639:READ A:POKE I,A:
POKE 709,A:SOUND 0,A,10,4:NEXT I
1430 SOUND 0,0,0,0:RETURN
1440 DATA 72,169,100,141,10,210
1450 DATA 141,24,208,141,26,208
1460 DATA 169,6,141,9,212,104
1470 DATA 64,104,104,133,204,104
1480 DATA 133,203,169,0,133,205
1490 DATA 169,224,133,206,162,4
1500 DATA 160,0,177,205,145,203
1510 DATA 200,208,249,230,204,230
1520 DATA 206,202,208,240,96,104
1530 DATA 162,16,169,9,157,66
1540 DATA 3,104,157,69,3,104
1550 DATA 157,68,3,169,0,157
1560 DATA 72,3,169,4,157,73
1570 DATA 3,32,86,228,96,104
1580 DATA 162,16,169,5,75,52
1590 DATA 6,9,104,169,0,9,0,133
1600 DATA 212,169,0,133,213,96
1610 POSITION 14,0:? "Filename?";
1620 FN$="":K=0
1630 POKE 20,0
1640 IF PEEK(764)<255 AND PEEK(764)<>39
AND PEEK(764)<>60 THEN 1670
1650 IF PEEK(20)<10 THEN 1640
1660 ? CHR$(21+11*K);"{LEFT}";:K=1-K:GOT
O 1630
1670 GET #2,A
1680 IF A=155 THEN ? " ";:FOR I=1 TO LEN
(FN$)+10:? "{BACK S}";:NEXT I:RETURN
1690 IF A=126 AND LEN(FN$)>1 THEN FN$=FN
$(1,LEN(FN$)-1):? " {LEFT}";CHR$(A);:GOT
O 1630
1695 IF A=126 AND LEN(FN$)=1 THEN ? CHR$
(A);:GOTO 1620
1700 IF A=58 OR (A>=48 AND A<=57) OR (A>
=65 AND A<=90) OR A=46 THEN 1720
1710 GOTO 1630
1720 IF LEN(FN$)<14 THEN FN$(LEN(FN$)+1)
=CHR$(A):? CHR$(A);
1730 GOTO 1630
1740 END
1750 REM GET CHOICE OF CHARACTER
1760 CY=INT(MRY/32):CX=MRY-32*CY
1770 C=CX+CY*32
1780 POKE SD+CX+CY*40+4,C+128
1790 POKE ASD+CX+CY*40+4,C+128
1800 IF STRIG(0)=0 OR PEEK(764)<255 THEN
 MRY=C:GOTO 1900
1810 ST=STICK(0):IF ST=15 THEN 1800
1820 POKE 53279,0
1830 GOSUB 1900
1840 CX=CX-(ST=11)+(ST=7):CY=CY-(ST=14)+
(ST=13)
1850 IF CX<0 THEN CX=31:CY=CY-1
1860 IF CX>31 THEN CX=0:CY=CY+1
1870 IF CY<0 THEN CY=3
1880 IF CY>3 THEN CY=0
1890 GOTO 1770
1900 POKE SD+CX+CY*40+4,C
1910 POKE ASD+CX+CY*40+4,C
1920 RETURN
```

Program 2.

```
100 REM CHLOAD--CHARACTER SET LOADER
110 CHSET=PEEK(106)-8:POKE 756,CHSET
120 CHSET=CHSET*256
130 TRAP 180
140 OPEN #1,4,0,"D:FONT":REM YOUR FILENA
ME HERE
150 FOR I=0 TO 1023
160 GET #1,A:POKE CHSET+I,A
170 NEXT I
180 CLOSE #1
```

Program 3.

```
100 REM CHPRINT--CHARACTER SET PRINTOUT
110 TRAP 340
120 OPEN #1,4,0,"D:FONT":REM YOUR FILENA
ME HERE
130 OPEN #2,8,0,"P:":REM CHANGE TO "E:"
FOR SCREEN
140 PRINT "|1| HEX OR |2| DECIMAL";:INPU
T TYPE
150 DIM HEX$(16),F$(3)
160 HEX$="0123456789ABCDEF"
165 LSB=-1
```

```
170 FOR I=0 TO 1023 STEP 8
180 F$="    ":C=INT(I/8)
190 IF TYPE=2 THEN F$(1,LEN(STR$(C)))=ST
R$(C):PRINT #2;F$;":";:GOTO 250
200 LSB=LSB+1:IF LSB=256 THEN LSB=0:MSB=
MSB+1
210 PRINT #2;"$";HEX$(MSB+1,MSB+1);
230 HINYB=INT(LSB/16):LONYB=LSB-16*HINYB

240 PRINT #2;HEX$(HINYB+1,HINYB+1);HEX$(
LONYB+1,LONYB+1);": ";
250 FOR J=0 TO 7
260 GET #1,A
270 F$="    ":IF TYPE=2 THEN F$(1,LEN(STR
$(A)))=STR$(A):PRINT #2;" ";F$;:GOTO 310

290 HINYB=INT(A/16):LONYB=A-16*HINYB
300 PRINT #2;HEX$(HINYB+1,HINYB+1);HEX$(
LONYB+1,LONYB+1);" ";
310 NEXT J
320 PRINT #2
330 NEXT I
340 CLOSE #1:CLOSE #2
```

©

Word Search

Bob Jones
Cranbury, NJ

Basically this program creates two matrices. The first matrix, the A matrix, is the one we shall hide the words in. Since the ATARI and many other BASICs I have run across do not permit the use of strings in a matrix, I have found that the next best thing to do is use the ASCII value of the characters instead. In this case it seems to be even simpler. The A matrix is initialized in line 10 to a random set of numbers between 65 and 91, (the ASCII value of the letters A thru Z). The C matrix will be our control matrix and our answer key. In line 10, all locations in C are initialized to 42, (the ASCII value for the character "*").

Next the user is asked to input 12 words, (the subroutine called by line 15). Lines 3015 through 3130 simply set A$ equal to the word to be processed, selected by the variable I. Line 45 sets L equal to the length of the word and if it is too long, (greater than ten letters) asks the user to input a shorter word. In line 50 we convert letters of the word to their ASCII values and place them in the B array, (a numeric array also initialized to all zero's by line 10). This array is our workhorse. Line 60 serves two functions: first, to generate a random starting location within both matrices and, second, to generate a random direction for the word to go in.

Now comes the math. Line 70 directs the program to one of eight subroutines, each one representing a different possible direction for the word to travel in. I shall only go over the first one, (lines 500 to 550) as the others work the same way. Line 500 checks to see that the word will fit within the matrix, if not the program is directed back to line 60 to generate a new starting location and direction. In line 510 we check the position of the word against the C or control matrix for possible conflicts with words already placed within that matrix. If a conflict exists the program is again directed back to line 60. Line 520 checks for a crossover with a previous word and if there is one it sets a flag, (the variable F) equal to 1. Line 630 directs the program to lines 2000 to 2020, these lines would have been repeated 8 times, once for each direction subroutine so in order to save memory they are only listed once and called upon when needed. The use of the 'GOTO' instead of the 'GOSUB' command is necessary in order to conditionally return to other portions of the program without confusing the computer by jumping in and out of subroutines. In these lines, (2000 to 2020) we continue to process our word, if there is a crossover (F = 1), or we have tried 300 times to find one, (determined by the variable R) we continue, otherwise we go back to line 60. Line 2020 gets us back into our original subroutine. Line 550 is the last line of our subroutine, it places our word into the A and C matrix's and sends us on to get a new word.

Line 80 determines if we have processed all of our words, and if so sends us on. In line 100 we print our hidden word matrix by printing the letters represented by our ASCII values, and when we are ready, line 110 prints our C matrix which is now our answer key.

This program requires more than 8K of memory as stands to run, though it will load into 8K of memory. It is a simple matter to shorten it by cutting out some of the possible direction subroutines. Also you can ask for the words to be INPUT as they are needed rather than storing them in string arrays. This program can be run on almost any computer using BASIC as stands, the only possible modifications that might be needed are with the GOTO statements like 'GOTO D*100'. These may be changed to 'ON D GOTO 500,600, 700,800,900,1000,1100,1200'. Or you could use the 'IF...THEN' statements, though the program won't be as much fun. A '?' is simply a PRINT command. The POKE statements are not necessary: they simply speed up the program. (Thanks to Ed Stewart, **COMPUTE!** #11.)

```
1 REM WORD SEARCH  WRITTEN BY BOB JONES
5 POKE 559,0:DIM A$(11),B$(11),C$(11),D$
(11),E$(11),F$(11),G$(11),H$(11),I$(11),
J$(11),K$(11),L$(11),M$(11)
10 DIM A(13,16),B(13),C(13,16):I=0:FOR X
=1 TO 13:B(X)=0:FOR Y=1 TO 16:A(X,Y)=INT
(26*RND(0)+65):C(X,Y)=42
15 NEXT Y:NEXT X:R=300:GOSUB 3000:GOTO 4
0
20 FOR X=1 TO 13:FOR Y=1 TO 16
30 R=0
40 GOSUB 3015
45 L=LEN(A$):IF L>10 THEN 3150
50 FOR S=1 TO L:B(S)=ASC(A$(S,S)):NEXT S

60 F=0:R=R+1:X=INT(13*RND(0)+1):Y=INT(16
*RND(0)+1):D=INT(8*RND(0)+5)
70 GOTO D*100
80 I=I+1:LPRINT A$:POKE 559,34:? A$:POKE
 559,0:IF I=12 THEN 100
90 GOTO 30
100 POKE 559,34:FOR X=1 TO 13:? :LPRINT
:FOR Y=1 TO 16:? CHR$(A(X,Y));:LPRINT CH
```

```
R$(A(X,Y));:NEXT Y:NEXT X
105 ? "TO SEE ANSWERS PRESS RETURN KEY":
INPUT A$
110 LPRINT :LPRINT :FOR X=1 TO 13:LPRINT
 :? :FOR Y=1 TO 16:LPRINT CHR$(C(X,Y));:
? CHR$(C(X,Y));:NEXT Y:NEXT X
120 LPRINT :LPRINT :LPRINT :LPRINT :LPRI
NT :END
500 IF Y+L-1>16 THEN 60
510 FOR Z=0 TO L-1:IF C(X,Y+Z)>42 AND C(
X,Y+Z)<>B(Z+1) THEN 60
520 IF C(X,Y+Z)=B(Z+1) THEN F=1
530 GOTO 2000
550 C(X,Y+S)=B(S+1):A(X,Y+S)=B(S+1):NEXT
 S:GOTO 80
600 IF Y-L+1<1 THEN 60
610 FOR Z=L-1 TO 0 STEP -1:IF C(X,Y-Z)>4
2 AND C(X,Y-Z)<>B(Z+1) THEN 60
620 IF C(X,Y-Z)=B(Z+1) THEN F=1
630 GOTO 2000
650 C(X,Y-S)=B(S+1):A(X,Y-S)=B(S+1):NEXT
 S:GOTO 80
700 IF X+L-1>13 THEN 60
710 FOR Z=0 TO L-1:IF C(X+Z,Y)>42 AND C(
X+Z,Y)<>B(Z+1) THEN 60
720 IF C(X+Z,Y)=B(Z+1) THEN F=1
730 GOTO 2000
750 C(X+S,Y)=B(S+1):A(X+S,Y)=B(S+1):NEXT
 S:GOTO 80
800 IF X-L+1<1 THEN 60
810 FOR Z=L-1 TO 0 STEP -1:IF C(X-Z,Y)>4
2 AND C(X-Z,Y)<>B(Z+1) THEN 60
820 IF C(X-Z,Y)=B(Z+1) THEN F=1
830 GOTO 2000
850 C(X-S,Y)=B(S+1):A(X-S,Y)=B(S+1):NEXT
 S:GOTO 80
900 IF X+L-1>13 OR Y+L-1>16 THEN 60
910 FOR Z=0 TO L-1:IF C(X+Z,Y+Z)>42 AND
C(X+Z,Y+Z)<>B(Z+1) THEN 60
920 IF C(X+Z,Y+Z)=B(Z+1) THEN F=1
930 GOTO 2000
950 C(X+S,Y+S)=B(S+1):A(X+S,Y+S)=B(S+1):
NEXT S:GOTO 80
1000 IF X-L+1<16 OR Y-L+1<1 THEN 60
1010 FOR Z=L-1 TO 0 STEP -1:IF C(X-Z,Y-Z
)>42 AND C(X-Z,Y-Z)<>B(Z+1) THEN 60
1020 IF C(X-Z,Y-Z)=B(Z+1) THEN F=1
1030 GOTO 2000
1050 C(X-S,Y-S)=B(S+1):A(X-S,Y-S)=B(S+1)
:NEXT S:GOTO 80
1100 IF Y-L+1<1 OR X+L-1>13 THEN 60
1110 FOR Z=0 TO L-1:IF C(X+Z,Y-Z)>42 AND
 C(X+Z,Y-Z)<>B(Z+1) THEN 60
1120 IF C(X+Z,Y-Z)=B(Z+1) THEN F=1
1130 GOTO 2000
1150 C(X+S,Y-S)=B(S+1):A(X+S,Y-S)=B(S+1)
:NEXT S:GOTO 80
1200 IF Y+L-1>16 OR X-L+1<1 THEN 60
1210 FOR Z=L-1 TO 0 STEP -1:IF C(X-Z,Y+Z
)>42 AND C(X-Z,Y+Z)<>B(Z+1) THEN 60
1220 IF C(X-Z,Y+Z)=B(Z+1) THEN F=1
1230 GOTO 2000
1250 C(X-S,Y+S)=B(S+1):A(X-S,Y+S)=B(S+1)
:NEXT S:GOTO 80
2000 NEXT Z:IF F>0 OR R>300 THEN 2020
2010 GOTO 60
2020 FOR S=0 TO L-1:GOTO D*100+50
3000 POKE 559,34:? "TYPE WORD AND THEN H
IT RETURN":INPUT B$,C$,D$,E$,F$,G$,H$,I$
,J$,K$,L$,M$:POKE 559,0:RETURN
3015 IF I=0 THEN A$=B$
3020 IF I=1 THEN A$=C$
3030 IF I=2 THEN A$=D$
3040 IF I=3 THEN A$=E$
3050 IF I=4 THEN A$=F$
3060 IF I=5 THEN A$=G$
3070 IF I=6 THEN A$=H$
3080 IF I=7 THEN A$=I$
3090 IF I=8 THEN A$=J$
3100 IF I=9 THEN A$=K$
3110 IF I=10 THEN A$=L$
3120 IF I=11 THEN A$=M$
3130 RETURN
3150 POKE 559,34:? A$;" TOO LONG MUST BE
 NO GREATER THAN 10 LETTERS TRY ANOTHER
WORD";:INPUT A$:POKE 559,0:GOTO 45
```

©

Review:

Screen Printer Interface (Version 2.0) From Macrotronics

David D Thornburg
Innovision
Los Altos, CA

More than anything else, I use my Atari computer for the creation of pictures. For various reasons, it is not enough for me to see these pictures on a TV screen — I also need copies of them on plain paper. Fortunately, there is an exceptionally well designed product which makes this a very simple task. That product is the screen printer Interface from Macrotronics. This program allows the user to transfer any image from the display screen to a suitable graphics printer with a single keystroke. The printed image can (if you choose) preserve grey scales, and can be printed in any size from a single sheet to a poster which would cover a wall. The user can choose among several printers (Trendcom, IDS, Centronics, Epson), and does not need the Atari 850 interface unit. Instead of the 850, Macrotronics provides a printer interface cable which connects to joystick ports 3 and 4. The screen printer software comes on a disk containing DOS 1, and they also provide a copy of the utility which is compatible with DOS 2.

The manual is clearly written and contains many examples showing the use of this interface with all language environments presently supported by Atari (BASIC, Assembler, PILOT).

Setting It Up

To use the system, one first connects the printer to the joystick ports with the cable provided and then boots the system from the disk supplied. During the boot process, the screen prompts the user for information on the printer being used. Once this is done, the rest of the program is loaded (the total utility occupies less than 3K bytes) and the familiar blue screen appears.

From this point on, the printer driver software is tucked safely inside the computer where it remains to do your bidding until the computer is turned off. Any command which sends information to device P: will cause this information to be printed. BASIC commands such as LPRINT behave just as they would for an Atari printer connected through the serial port.

While this system supports all text printing functions, the real value of this interface package is the power it gives as a graphics printing tool. Any time this system is in the computer you can get a dot-by-dot copy of the screen image by simply typing CTRL-P. Macrotronics has created some default printer conditions which cause most images to be printed quite nicely. The user has total control over the system parameters and can change the settings of various registers to create many different effects.

For example, the printed image can be scaled independently in both axes by POKEing a number between 1 and 16 in each of two memory locations. The default scale (16) produces a figure which fits nicely on 8.5" wide paper. As the scale values are decreased, the image size increases by 16/n where n is the scale value. Wide images are printed in multiple strips which can then be glued together. On multiple strip printouts, each strip overlaps the previous one by a little bit to make strip alignment simple. This attention to making life simple for the user is beautiful!

In addition to using the scale variables to make large pictures, they can also be used to adjust for the fact that most dot matrix printers have different inter-dot spacings on each axis. To get an accurate square on the Epson MX-100, for example, the vertical scale should be set to 14 (with the horizontal scale left at 16). The result is almost perfect.

In addition to scaling, the user can select positive or negative images, grey scale or black and white, determine grey scale from either hue or luminance data, print data which has been "fine scrolled," and print players and missiles.

In short, if your Atari computer can generate it, the Macrotronics screen printer can print it.

I use this software almost every day. So far I have used it to print the illustrations for three book manuscripts, numerous articles, several large posters and some custom bumper stickers.

The Only Error

The only error I have uncovered is that the default grey scale setting uses *hue* data rather than *luminance* data, but this is just a documentation error — the software works perfectly.

To see more examples of printouts made with this utility, look at any "Friends of the Turtle" column in **COMPUTE!**, or at the book *Picture This!*, soon to be published by Addison Wesley.

Figure 1.

Figure 2.

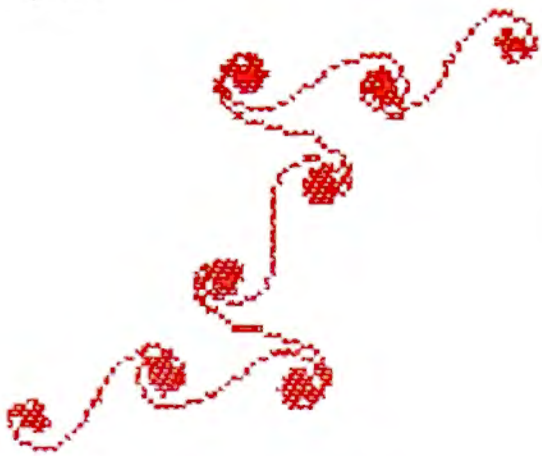
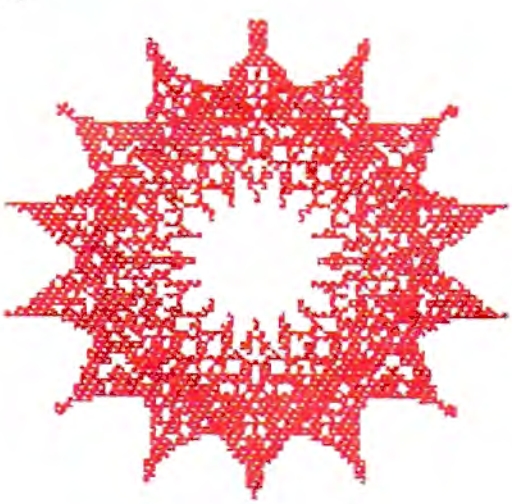


Figure 3.

Figure 4.



©

INSIGHT: ATARI

Bill Wilkinson
Cupertino, CA

I have recently seen a copy of the complete *De Re Atari* (by Atari's own Chris Crawford, author of SCRAM and EASTERN FRONT, et al). Since two out of three people I talk to say "Huh?" when I mention the name, I have personally subtitled it *Everything You Ever Wanted to Know About the Atari Computers But Didn't Know Enough to Ask.* The book concerns itself with foibles, tricks, innards, hardware, software, and everything in between: there are even tricks using Atari BASIC (that are "obvious" upon discovery) which we never thought about when we designed the thing! I must heartily recommend that every serious Atari programmer trade in his or her left thumb, if necessary, for a copy of this book.

"De Re" (the insiders' appellation) is currently being serialized in BYTE magazine (I guess Atari's trying to impress the non-Atari world), but seeing the book in one piece is somehow more instructive. "De Re" is generally a fantastic resource, but it does often assume that the reader has intimate knowledge and understanding of the Atari Hardware Reference Manuals, etc. This is *not* a fault (the authors forewarn the reader); and, besides, it does leave room for columns like this. I don't intend to duplicate material in either Atari's manuals or "De Re", but there is bound to be some overlap. I intend to present the "hows" and "whys" to supplement Atari's "whats."

I try to write this column for the programmer: the person who knows software, but is unfamiliar with Atari hardware and/or Atari's system level software. If this column stretches your understanding of the Atari and/or its software, that's probably good. And I am constantly amazed at the questions which beginners on the Atari come up with; they often show "insights" to solution methods that I wouldn't dream of. The first questions are arriving in my mailbox. Send more!

This month's column is part three of the series on the Atari Operating System. Next month we will cover screen output, including graphics, to formally end the series. I have a few ideas on what should come next for you non-BASIC Atari users, but I would welcome some input. Also, this month, we begin a series which will explore the inner workings of Atari BASIC.

Atari I/O, Part 3: Device Handlers

As we noted before, Atari's OS is actually a very small program (approximately 700 bytes). Even so, it is able to handle the wide variety of I/O requests detailed in the first two parts of this series with a surprisingly simple and consistent assembly language interface. Perhaps even more amazing is the purity and simplicity of the OS interface to its device handlers.

Admittedly, because of this very simplicity, Atari's OS is sometimes slower than one would wish (probably only noticeably so with PUT BINARY RECORD and GET BINARY RECORD) and the handlers must be relatively sophisticated. But not overly so, as we will show.

The Device Handler Table

Atari OS has, in ROM, a list of the standard devices (P:,C:,E:,S:, and K:) and the addresses thereof. So far, so good. But notice that, for example, the disk handler (D:) is not listed there; how does OS know about other devices? Simple. On SYSTEM RESET, the list is moved from ROM to RAM, and OS then utilizes only the RAM version. To add a device, simply tack it on to the end of the list: you need only specify the device's name (one character) and the address of its handler table (more on that in a moment). To reassure you that it is this simple, let me point out that this is exactly how the "D:" (Disk) handler is attached when the disk is booted.

In theory, all named device handlers under Atari OS may handle more than one physical device. Just as the disk handler understands "D1:" and "D2:", so could a printer handler understand "P1:" and "P2:". In practice, of all the standard Atari handlers only the Disk and Serial Port handlers can utilize the sub-device numbers. Incidentally, Atari OS supplies a default sub-device number of "1" if no number is given (thus "D:" becomes "D1:"). A project for those of you with two printers (there

```
        *=      $031A
HTABS                   ; the Printer device
        .WORD   PDEVICE ; and the address of its
                          driver
        .BYTE   'C'     ; the Cassette device
        .WORD   CDEVICE
        .BYTE   'E'     ; the screen Editor device
        .WORD   EDEVICE
        .BYTE   'S'     ; the graphics Screen
                          device
        .WORD   SDEVICE
        .BYTE   'K'     ; the Keyboard device
        .WORD   KDEVICE
        .BYTE   0       ; zero marks the end of
                          the table
        .WORD   0       ; ...but there's room for
                          up to
        .BYTE   0       ; ...9 more devices
        et cetera
```

Figure 1.

must be one or two of you): presumably one of them is connected via the MacroTronics interface; if so, try modifying the MacroTronics handler so that "P1:" refers to the Atari 850 interface while "P2:" refers to the MacroTronics. It's really a fairly easy project, presuming you have the listings of Atari's OS (which are available from Atari).

Rules For Writing Device Handlers

Each device which has its handler address placed into the handler address table (above) is expected to conform to certain rules. In particular, the driver is expected to provide six action subroutines and an initialization routine. (In practice, I believe the current Atari OS only calls the initialization routines for its own pre-defined devices. Since this may change in future OS's and since one can force the call to one's own initialization routine, I must recommend that each driver include one, even if it does nothing.) The address placed in the handler address table must point to, again, another table, the form of which is shown in Figure 2.

Notice the six addresses which must be specified; and note that, in the table, one must subtract one from each address (the "-1" simply makes CIO's job easier...honest). A brief word about each routine is in order.

The OPEN routine must perform any initialization needed by the device. For many devices, such as a printer, this may consist of simply checking the device status to insure that it is actually present. Since the X-register, on entry to each of these routines, contains the IOCB number being used for this call, the driver may examine ICAX1 (via LDA ICAX1,X) and/or ICAX2 to determine the kind of OPEN being requested. (Caution: Atari OS preempts bits 2 and 3, $04 and $08, of ICAX1 for read/write access control. These bits may be examined, but should normally not be changed.)

The CLOSE routine is often even simpler. It should "turn off" the device if necessary and if possible.

The PUTBYTE and GETBYTE routines are just what are implied by their names: the device handler must supply a routine to output one byte to the device and a routine to input one byte from the device. *However*, for many devices, one or the other of these routines doesn't make sense (ever tried to input from a printer?). In this case the routine may simply RTS and Atari OS will supply an error code.

The STATUS routine is intended to implement a dynamic status check. Generally, if dynamic checking is not desirable or feasible, the routine may simply return the status value it finds in the user's IOCB. However, it is *not* an error under Atari OS to call the status routine for an unOPENed device, so be careful.

The XIO routine does just what its name implies: it allows the user to call any and all special and wonderful routines that a given device handler may choose to implement. OS does nothing to process an XIO call except pass it to the appropriate driver.

Note: In general, the AUXilliary bytes of each IOCB are available to each driver. In practice, it is best to avoid ICAX1 and ICAX2, as several BASIC and OS commands will alter them at their will. Note that ICAX3 through ICAX5 may be used to pass and receive information to and from BASIC via the NOTE and POINT commands (which are actually special XIO commands). Finally, drivers should not touch any other bytes in the IOCBs, especially the first two bytes.

Notice that handlers need not be concerned with PUT BINARY RECORD, GET TEXT RECORD, etc.: OS performs all the needed housekeeping for these user-level commands.

```
HANDLER
          .WORD   <address of OPEN routine>-1
          .WORD   <address of CLOSE routine>-1
          .WORD   <address of GETBYTE routine>-1
          .WORD   <address of PUTBYTE routine>-1
          .WORD   <address of STATUS routine>-1
          .WORD   <address of XIO routine>-1
          JMP     <address of initialization routine>
```

Figure 2.

Rules For Adding Things To OS

We touched on this subject last month, in the section titled "The Easiest Way of Making Room?", but a review and an addition are in order. Both Atari FMS (File Manager System, otherwise known as DOS and/or the Disk Device Driver) and the serial port handlers follow the same scheme when they add themselves to OS, so it is safe to assume that this method may be considered the *de facto* Atari standard. We enumerate:

1. Inspect the system MEMLO pointer (at $2E7, I called it LOMEM last month, which is BASIC's name for it).

2. Load your routine (including needed buffers) at the current value of MEMLO.

3. Add the size of your routine to MEMLO.

4. Store the resultant value back in MEMLO.

5. Connect your driver to OS by adding its name and address into the handler address table.

6. Fool OS so that if SYSTEM RESET is hit steps 3 through 5 will be re-executed (because SYSTEM RESET indeed resets the handler address table and the value of MEMLO).

In point of fact, step 2 is the hardest of these to accomplish. In order to load your routine at wherever MEMLO may be pointing, you need a relocatable (or self-relocatable) routine. Since there is currently no assembler for the Atari which produces relocatable code, this is not an easy task. (*However*, I just happen to have a method which works. But it will have to wait for a later article.)

Step 6 is accomplished by making Atari OS think that your driver is the Disk driver for initialization purposes (by "stealing" the DOSINI vector) and then calling the Disk's initializer yourself when steps 3 through 5 are performed. This is a fairly simple process, but again, details must await a future article.

Yet Another Real Live Example

I promised last month that we would present a driver for a "peripheral" device found in every Atari, yet not supported by any Atari device handlers. I could have been cagey and presented a driver for a "Null" device. (A handy thing to have, actually: One can throw away one's output *very* fast when trying to debug a program. See *De Re Atari* for a simple implementation of one. Better yet, try to write one from the information presented herein.) Being a glutton for punishment, I undertook to write a truly useful handler for Atari's overlooked device: RAM memory!!

After the snickers and sarcastic comments die down, let me point out how truly useful such a device is to BASIC programs: program one can "write" data to RAM and then chain to program two, which then "reads" the same data back. Voila! Chaining with COMMON in Atari BASIC. So herewith the "M:" (Memory) driver, presented in its entirety in Figure 3.

Does It Work?

Some words of caution are in order. This driver does *not* perform step 6 as noted in the last section (but it may be reinitialized via a BASIC USR call). It does *not* perform self-relocation: instead it simply locates itself above all normal low memory usage (except the serial port drivers, which would have to be loaded *after* this driver). If you assemble it yourself, you could do so at the MEMLO you find in your normal system configuration (or you could improve it to be self-modifying, of course).

Other caveats pertain to the handler's usage: it uses RAM from the contents of MEMTOP ($2E5) downward. It does *not* check to see if it has bumped into BASIC's MEMTOP ($90) and hence could conceivably wipe out programs and/or data. To be safe, don't write more data to the RAM than a FRE(0) shows (and preferrably even less).

In operation, the M: driver reinitializes upon an OPEN for write access (mode 8). A CLOSE followed by a subsequent READ access will allow the data to be read in the order it was written. More cautions: don't change graphics modes between writing and reading if the change would use more memory (to be safe, simply don't change at all). The M: will perform almost exactly as if it were a cassette file, so the user program should be data sensitive if necessary: the M: driver will *not* itself give an error based on data contents. Note that the data may be re-READ if desired (via CLOSE and re-OPEN).

Installing The M: Driver

The most obvious way to install this driver (Program 1) is to type in the source and assemble it directly to the disk. Then simply loading the object file from DOS 2 (or OS/A+) will activate the driver and move LOMEM as needed. You could even name the resulting file "AUTORUN.SYS" so that it would be automatically booted on power up.

If you don't have an assembler and/or disk, the problem is a little more difficult. If you are comfortable writing BASIC programs that load assembly language data to memory, you migth use the techniques described in last month's "Make Room?" to reserve the required memory. Then a simple POKEr program which uses DATA statements would suffice.

But the assembly listing given here is designed for a disk system and would waste 5K bytes or so in a cassette system. So, if you can't reassemble it and/or write that POKEr program, you will just have to be patient: I will try to give you a simplified BASIC POKEr program next month.

A suggested set of BASIC programs is presented:

Ending of Program 1:

```
9900 OPEN #2,8,0,"M:"
9910 PRINT #2; LEN(A$)
9920 PRINT #2; A$
9930 CLOSE #2
9940 RUN "D:PROGRAM2"
```

Beginning of Program 2:

```
100 JUNK = USR( 7984 )
    [ to insure the M: driver is linked, in case of
    RESET ]
110 OPEN #4,4,0,"M:"
120 INPUT #4, SIZE
130 DIM STRING$(SIZE)
140 INPUT #4, STRING$
150 CLOSE #4
```

BASIC A+ users might find RPUT/RGET and BPUT/BGET to be useful tools here instead of PRINT and INPUT. And, of course, users of any other language(s) might find this a handy inter-program communications device.

BASIC, Part 1: Why?

The first "Why?" I usually hear is "Why not Microsoft BASIC?" After a little probing, I find that the question really boils down to "Why not string arrays?" There is no simple answer to that question, so I hope to save myself time in the future by pointing toward these articles. Because I intend to give the true and not-so-simple answer, along with some (hopefully) very interesting information.

Believe it or not, Atari BASIC pretty much works the way it was designed and specified. And yours truly must take a large part of the brickbats or roses you might throw because of those specifications. We (that is, at the time, Shepardson Microsystems) were just finishing the highly successful and very powerful Cromemco 32K Structured BASIC. And, while a few Cromemco users had carped about the lack of string arrays, on the whole the real power of the language is extraordinarily impressive. All this "power" probably went to our head(s), so of course we had to duplicate the feat for Atari.

Oops. A small problem: Cromemco gave us 32K bytes for Structured BASIC; Atari gave us 10K bytes. What comes out? Wrong question! What can stay in?! Of course, Atari had some ideas, too, and the important features that we ended up with include (in my opinion):

- Decimal Arithmetic
- Long Variable Names
- Long Strings (more than 255 bytes)
- Flexible I/O
- Reasonable Assembly Language Interface
- Syntax Check at entrh time

That last item won't be appreciated by those of you who haven't used a BASIC that doesn't do it, so I will try to describe the horrors to you: You type in a long program which includes a line such as:

```
3034 IF SYSTEMERROR THEN
     PINT "Bad Disk Drive":
     GOTO 4090
```

Did you catch it? It says 'PINT' where it should say 'PRINT'. Most microcomputer BASICs will happily gulp that line in with nary a burp. Now, 13 months later, when that dreaded 'systemerror' actually occurs, your user (who lives in Hong Kong, naturally) sees the helpful message

```
***SYNTAX ERROR at LINE 3037
```

When you have fathomed the implications of that, calm your nerves so we can continue.

Needless to say, we were more than happy to include the Syntax Check feature. However, this inclusion had implications that rippled throughout the rest of the design of BASIC. First, you don't get something for nothing: such syntax checking uses memory, perhaps one to two kilobytes. Second, pre-syntaxing implies that the user program will be "tokenized": that is, the user's source will be converted into internal tokens for ease of execution and efficiency. Even Microsoft BASICs tokenize the keywords of the language; Atari BASIC tokenizes *everything:* keywords, variables, constants, operators, etc. Thirdly, the decision to have strings longer than 255 characters (coupled with the tight memory requirements) simply precluded any implementation of string arrays. (In fact, I do not know of *any* small-machine BASIC that supports string arrays with elements longer than 255 characters.)

Before perusing some quickie programs to show the effects of tokenizing, I should like to give some credit where it is due. Though I participated in the specifications for Atari BASIC, I had little to do with the actual implementation. More history: Atari asked us (in September, 1978) to bid on producing a custom "consumer-oriented" BASIC for them. Sometime in October, the specifications were finalized and Paul Laughton and Kathleen O'Brien (with a very little help from three more of us) began to work in earnest. The contract called for delivery by April 6, 1979, and included delivery of a File Manager System (DOS 1). Atari planned to take an early, 8K Microsoft BASIC to the Consumer Electronics Show (in Las Vegas) in January, 1979, and then switch later. The actual purchase order took a while to get through Atari's red tape, and the final version thereof is dated 12/28/78 — about one week *after* both BASIC and DOS were delivered to Atari! Atari took Atari BASIC to CES.

Investigating BASIC's Tokens

There are three fundamental types of tokens in Atari BASIC, each of which occupies exactly one byte of RAM memory, with only two special cases. The token types are statement name tokens, operator name tokens (which include function names and some other miscellany), and variable name tokens. The special cases are numeric and string constants, which begin with an operator name token, but are followed by the actual value of the constant.

Statement name tokens can *only* occur as the first item of a statement and, thus, have their own keyword and tokenizing table. In theory, Atari BASIC's structure could support up to 256 types of statements. Variable name tokens and operator name tokens are intermixed throughout the rest of a statement and are distinguished by the state of their upper bit: variable name tokens have their upper bit on, operators don't.

A few of the statement types are also special cased in that they are not followed by operator and variable tokens. These special cases include the obvious REM and DATA and the not-so-obvious ERROR (the statement name given to lines containing a syntax error).

Since each variable is reduced to a single byte (with its upper bit set), there are a maximum of 128 different variable names per program. There is the further implication that BASIC must remember the association of name to token in order to LIST your program back to you. The actual ATASCII names are stored in the "Variable Name Table," and we investigated its structure in **COMPUTE!** #17 under the heading of "VARIABLE, VARIBLE, VARABLE." (Briefly, the names are simply stored one after the other, with the upper bit of the last character of each name turned on.)

The statement and operator names are obviously predefined in the BASIC ROM cartridge, and we offer herewith a program (Program 2) which prints out the token numbers and corresponding keywords. When you run the program, you will notice that some operators (especially the left parenthesis) appear to be repeated. They are. We will find out why next month.

Program 1.

```
A sample device driver for Atari's OS
--- general remarks ---

0000          1000          .PAGE  "--- general remarks ---"
              1010 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
              1020 ;
              1030 ; The "M:" driver --
              1040 ;    Using memory as a device
              1050 ;
              1060 ; Includes installation program
              1070 ;
              1080 ; Written by Bill Wilkinson
              1090 ;   for January, 1982, COMPUTE!
              1100 ;
              1110 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

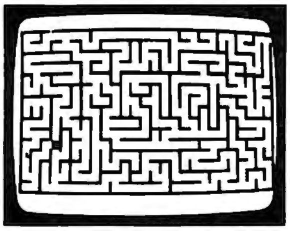
```
                 1120 ;
                 1130 ; EQUATES INTO ATARI'S OS, ETC.
                 1140 ;
034A             1150 ICAUX1 =    $34A       ; The AUX1 byte of IOCB
                 1160 ;
0008             1170 OPOUT  =    8          ; Mode 8 is OPEN for OUTPUT
                 1180 ;
02E7             1190 MEMLO  =    $2E7       ; pointer to bottom of free RAM
02E5             1200 MEMTOP =    $2E5       ; pointer to top of free RAM
                 1210 ;
00E0             1220 FR1    =    $E0        ; Fltg Pt Register 1, scratch
                 1230 ;
0001             1240 STATUSOK =  1          ; I/O was good
0088             1250 STATUSEOF = $88        ; reached an end-of-file
                 1260 ;
031A             1270 HATABS =    $31A
                 1280 ;
0100             1290 HIGH   =    $100       ; divisor for high byte
00FF             1300 LOW    =    $FF        ; mask for low byte
                 1310 ;

A sample device driver for Atari's OS
The installation routine

0000             1320        .PAGE "The installation routine"
                 1330 ;
0000             1340        *=   $1F00
                 1350 ; This first routine is simply
                 1360 ; used to connect the driver
                 1370 ; to Atari's handler address
                 1380 ; table.
                 1390 ;
                 1400 LOADANDGO
1F00 A200        1410        LDX  #0           ; We begin at start of table
                 1420 SEARCHING
1F02 BD1A03      1430        LDA  HATABS,X   ; Check device name
1F05 F00A        1440        BEQ  EMPTYFOUND ; Found last one
1F07 C94D        1450        CMP  #'M        ' ; Already have M: ?
1F09 F01A        1460        BEQ  MINSTALLED ; Yes, don't reinstall
1F0B E8          1470        INX
1F0C E8          1480        INX
1F0D E8          1490        INX             ; Point to next entry
1F0E D0F2        1500        BNE  SEARCHING  ; and keep looking
1F10 60          1510        RTS             ; Huh? Impossible!!!
                 1520 ;
                 1530 ; We found the current end of the
                 1540 ; table...so extend it.
                 1550 ;
                 1560 EMPTYFOUND
1F11 A94D        1570        LDA  #'M        ' ; Our device name, "M:"
1F13 9D1A03      1580        STA  HATABS,X   ; is first byte of entry
1F16 A93B        1590        LDA  #MDRIVER&LOW
1F18 9D1B03      1600        STA  HATABS+1,X ; LSB of driver addr
1F1B A91F        1610        LDA  #MDRIVER/HIGH
1F1D 9D1C03      1620        STA  HATABS+2,X ; and MSB of addr
1F20 A900        1630        LDA  #0
1F22 9D1D03      1640        STA  HATABS+3,X ; A new end for the table
                 1650 ;
                 1660 ; now change LOMEM so BASIC won't
                 1670 ; overwrite us.
```

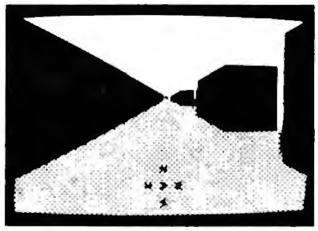
```
                 ;
                 MINSTALLED
1F25 A900              LDA  #DRIVERTOP&LOW
1F27 8DE702            STA  MEMLO       ; LSB of top addr
1F2A A920              LDA  #DRIVERTOP/HIGH
1F2C 8DE802            STA  MEMLO+1     ; and MSB therof
                 ;
                 ; and that's all we have to do!
                 ;
1F2F 60                RTS
                 ;
                 ;
                 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                 ;
                 ; This entry point is provided
                 ; so that BASIC can reconnect
                 ; the driver via a USR(RECONNECT)
                 ;
                 RECONNECT
1F30 68                PLA
1F31 F0CD              BEQ  LOADANDGO ; No parameters, I hope
1F33 A8                TAY
                 PULLTHEM
1F34 68                PLA
1F35 68                PLA              ; get rid of a parameter
1F36 88                DEY
1F37 D0FB              BNE  PULLTHEM  ; and pull another
1F39 F0C5              BEQ  LOADANDGO ; go reconnect
                 ;
```

```
A sample device driver for Atari's OS
The driver itself

1F3B                     .PAGE "The driver itself"
               ;
               ; Recall that all drivers must
               ; be connected to OS through
               ; a driver routines address table.
               ;
               MDRIVER
1F3B 4C1F                .WORD MOPEN-1   ; The addresses must
1F3D 6F1F                .WORD MCLOSE-1  ; ...be given in this
1F3F 921F                .WORD MGETB-1   ; ...order and must .
1F41 851F                .WORD MPUTB-1   ; ...be one (1) less
1F43 9F1F                .WORD MSTATUS-1 ; ...than the actual
1F45 491F                .WORD MXIO-1    ; ...address
1F47 4C4A1F              JMP  MINIT      ; This is for safety only
               ;
               ; For many drivers, some of these
               ; routines are not needed, and
               ; can effectively be null routines
               ;
               ; A null routine should return
               ; a one (1) in the Y-register
               ; to indicate success.
               ;
               MXIO
               MINIT
1F4A A001                LDY  #1         ; success
1F4C 60                  RTS
               ;
               ; If a routine is omitted because
               ; it is illegal (reading from a
               ; printer, etc.), simply pointing
               ; to an RTS is adequate, since
               ; Atari OS preloads Y with a
               ; 'Function Not Implemented' error
               ; return code.
               ;
```

```
A sample device driver for Atari's OS
The driver function routines

1F4D                     .PAGE "The driver function routines"
               ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
               ;
               ; Now we begin the code for the
               ; routines that do the actual
               ; work
               ;
               MOPEN
1F4D BD4A03              LDA  ICAUX1,X   ; Check type of open
1F50 2908                AND  #OPOUT     ; Open for output?
1F52 F00D                BEQ  OPENFORREAD ; No...assume for input
1F54 ADE502              LDA  MEMTOP
1F57 8DD21F              STA  MSTART     ; We start storing
1F5A ACE602              LDY  MEMTOP+1   ; ...the bytes
1F5D 88                  DEY             ; ...one page below
1F5E 8CD31F              STY  MSTART+1   ; the supposed top of mem
```

CAPTIVITY





the most advanced maze game available to the ATARI[*] owner

A new game by Craig Patchett which uses machine language routines, 5-color, high-resolution graphics and sound effects.

The player's goal is to guide an imaginary robot through and out of a 3-dimensional maze.

20 game variations insure countless hours of enjoyment.

CAPTIVITY can be used with any **ATARI[*]** computer with at least 24K. It is available in cassette format only. Price is $24.95 plus $2.00 shipping and handling.

Available at fine computer stores or write us directly.

PDI

Program Design, Inc. 11 Idar Court Greenwich, CT 06830
203-661-8799

ATARI[*] is the registered trademark of **ATARI**, Inc.

ATARI 400/800 OWNERS: THERE'S A MICROCONNECTION™ FOR YOU! —

Now you can direct-connect to the telephone with or without the 850™ interface, there's also a serial port to drive a printer, optional autodial and autoanswer, and smart terminal software! Prices start at $199.50 For more details write or phone:

the microperipheral corporation

2643A - 151st Pl. N.E.
Redmond, WA 98052
(206) 881-7544

master charge
VISA

SOFTWARE STREET
PRESENTS

ATARI

800 Computer 16k.....$729.00
800 Computer 32k.....$769.00
800 Computer 48k.....$839.00
825 Printer(80 col)..$599.00
810 Disk Drive.......$429.00
850 Intf. Mod........$155.00
822 Thermal Printer..$329.00
820 Printer..........$259.00
410 Recorder.........$65.00
830 Modem............$145.00

Special Software Packages

The Entertainer.......$69.00
Star Raiders, Missile Command, and Joysticks.
The Programmer........$54.00
BASIC Programming language, Self teaching guide to BASIC, and BASIC reference manual

The Educator..........$99.00
410 Recorder, BASIC programming language, and States and Capitals.

<u>TOP SELLING PROGRAMS</u>

Missile Command........$33.96
Asteroids..............$33.96
Invitation to Prog. 2..$16.96
Invitation to Prog. 3..$16.96
Assembler/Editor.......$48.96
Jaw Breaker............$25.95
Cranston Manor.........$29.95
Dodge Racer............$19.95
Atari Word Processor..$129.95
Atari World............$59.95
Star Raiders...........$39.96
Intruder Alert.........$14.95
Sammy the Sea Serpent..$14.95

savings from:
Atari
Atari Program Exchange
Adventure International
Crystal
P.D.I.
L.J.K.
Dynacomp
Quality Software
Avalon Hill
Epyx
Computer consultants
Synapse
Datasoft
United Software
On-Line
O.S.S.
Software Street

ATARI IS A REGISTERED TRADEMARK

Software Street
3392 Clipper Dr.
Chino, CA 91710
(714) 591-3061

Software Street is your <u>mail order</u> Atari discount center.

Shipping costs:
Software-minimum $2.00
Hardware-prices will vary
(please call)

prices subject to change

```
               2490 ;
               2500 ; now we join up with mode 4 open
               2510 ;
               2520 OPENFORREAD
1F61 ADD21F    2530        LDA   MSTART      ; simply move the
1F64 8DCE1F    2540        STA   MCURRENT    ; ...start pointer
1F67 ADD31F    2550        LDA   MSTART+1    ; ...to the current
1F6A 8DCF1F    2560        STA   MCURRENT+1  ; ...pointer, both bytes
               2570 ;
1F6D A001      2580        LDY   #STATUSOK
1F6F 60        2590        RTS               ; we don't acknowledge failure
               2600 ;
               2610 ;
               2620 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
               2630 ;
               2640 ; the routine for CLOSE of M:
               2650 ;
               2660 MCLOSE
1F70 BD4A03    2670        LDA   ICAUX1,X    ; check mode of open
1F73 2908      2680        AND   #OPOUT      ; was for output?
1F75 F00C      2690        BEQ   MCLREAD     ; no...close input 'file'
               2700 ;
1F77 ADCE1F    2710        LDA   MCURRENT    ; we establish our
1F7A 8DD01F    2720        STA   MSTOP       ; ...limit so that
1F7D ADCF1F    2730        LDA   MCURRENT+1  ; ...next use can't
1F80 8DD11F    2740        STA   MSTOP+1     ; ...go too far
               2750 ;
               2760 MCLREAD
1F83 A001      2770        LDY   #STATUSOK
1F85 60        2780        RTS               ; and guaranteed to be ok
               2790 ;
               2800 ;
               2810 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
               2820 ;
               2830 ; This routine puts one byte
               2840 ; to the memory for later
               2850 ; retrieval.
               2860 ;
               2870 MPUTB
1F86 48        2880        PHA               ; save the byte to be PUT
1F87 20B51F    2890        JSR   MOVECURRENT ; get ptr to zero page
1F8A 68        2900        PLA               ; the byte again
1F8B A000      2910        LDY   #0
1F8D 91E0      2920        STA   (FR1),Y     ; put the byte, indirectly
1F8F 20C01F    2930        JSR   DECCURRENT  ; point to nxt byte
1F92 60        2940        RTS               ; that's all
               2950 ;
               2960 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
               2970 ;
               2980 ; routine to get a byte put
               2990 ; in memory before.
               3000 ;
               3010 MGETB
1F93 20A01F    3020        JSR   MSTATUS     ; any more bytes?
1F96 B007      3030        BCS   MGETRTS     ; no...error
1F98 A000      3040        LDY   #0
1F9A B1E0      3050        LDA   (FR1),Y     ; yes...get a byte
1F9C 20C01F    3060        JSR   DECCURRENT  ; and point to next byte
               3070 MGETRTS
```

```
1F9F 60              RTS
               ;
               ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
               ;
               ; check the status of the driver
               ;
               ; this routine is only valid
               ; when READing the 'file'...
               ; "M:" never gets errors when
               ; writing.
               ;
               MSTATUS
1FA0 20B51F          JSR  MOVECURRENT ; current ptr to zero page
1FA3 CDD01F          CMP  MSTOP       ; any more bytes to get?
1FA6 D009            BNE  MSTOK       ; yes
1FA8 CCD11F          CPY  MSTOP+1     ; double chk
1FAB D004            BNE  MSTOK       ; yes, again
1FAD A088            LDY  #STATUSEOF ; oops...
1FAF 38              SEC              ; no more bytes
1FB0 60              RTS
               ;
               MSTOK
1FB1 A001            LDY  #STATUSOK ; all is okay
1FB3 18              CLC              ; flag for MGETB
1FB4 60              RTS
```

A sample device driver for Atari's OS
Miscellaneous subroutines

```
1FB5                 .PAGE "Miscellaneous subroutines"
               ;
               ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
               ;
               ; finally, we have a couple of
               ; short and simple routines to
               ; manipulate MCURRENT, the ptr
               ; to the currently accessed byte
               ;
               ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
               ;
               ; MOVECURRENT simply moves
               ;  MCURRENT to the floating
               ;  point register, FR1, in
               ;  zero page.  FR1 is always
               ;  safe to use except in the
               ;  middle of an expression.
               ;
               MOVECURRENT
1FB5 ADCE1F          LDA  MCURRENT
1FB8 85E0            STA  FR1         ; notice that we use
1FBA ACCF1F          LDY  MCURRENT+1 ; both the A and
1FBD 84E1            STY  FR1+1       ; Y registers...this
1FBF 60              RTS              ; is for MSTATUS use
               ;
               ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
               ;
               ; DECCURRENT simply does a two
               ;  byte decrement of the MCURRENT
               ;  pointer and returns with the
```

```
               3630 ;  Y register indicating OK status.
               3640 ; NOTE that the A register is
               3650 ;  left undisturbed.
               3660 ;
               3670 DECCURRENT
1FC0 ACCE1F    3680        LDY  MCURRENT   ; check LSB's value
1FC3 D003      3690        BNE  DECLOW     ; if non-zero, MSB is ok
1FC5 CECF1F    3700        DEC  MCURRENT+1 ; if zero, need to bump MSB
               3710 DECLOW
1FC8 CECE1F    3720        DEC  MCURRENT   ; now bump the LSB
1FCB A001      3730        LDY  #STATUSOK  ; as promised
1FCD 60        3740        RTS
```

A sample device driver for Atari's OS
RAM usage and clean up

```
1FCE           3750        .PAGE "RAM usage and clean up"
               3760 ;
               3770 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
               3780 ;
               3790 ; END OF CODE
               3800 ;
               3810 ;
               3820 ; Now we define our storage
               3830 ; locations.
               3840 ;
               3850 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
               3860 ;
               3870 ;
               3880 ; MCURRENT holds the pointer to
               3890 ; the next byte to be PUT or GET
1FCE 0000      3900 MCURRENT .WORD 0
               3910 ;
               3920 ; MSTOP is set by CLOSE to point
               3930 ; to the last byte PUT, so GET
               3940 ; won't try to go past the end
               3950 ; of data.
1FD0 0000      3960 MSTOP  .WORD 0
               3970 ;
               3980 ; MSTART is derived from MEMTOP
               3990 ; and points to the first byte
               4000 ; stored.  The bytes are stored
               4010 ; in descending addresses until
               4020 ; MSTOP is set by CLOSE.
1FD2 0000      4030 MSTART .WORD 0
               4040 ;
               4050 ; DRIVERTOP becomes the new
               4060 ; contents of MEMLO
2000           4070 DRIVERTOP = *+$FF&$FF00
               4080 ; (sets to next page boundary)
               4090 ;
               4100 ;
               4110 ; The following is how you make
               4120 ; a LOAD-AND-GO file under
               4130 ; Atari's DOS 2
               4140 ;
1FD4           4150        *=   $2E0
02E0 001F      4160        .WORD LOADANDGO
               4170 ;
```

```
              4180 ;
02E2          4190          .END

A sample device driver for Atari's OS
RAM usage and clean up

=034A ICAUX1        =0008 OPOUT         =02E7 MEMLO         =02E5 MEMTOP
=00E0 FR1           =0001 STATUSOK      =0088 STATUSEOF     =031A HADABS
=0100 HIGH          =00FF LOW            1F00 LOADANDGO      1F02 SEARCHING
 1F11 EMPTYFOUND     1F25 MINSTALLED     1F3B MDRIVER       =2000 DRIVERTOP
 1F30 RECONNECT      1F34 PULLTHEM       1F4D MOPEN          1F70 MCLOSE
 1F93 MGETB          1F86 MPUTB          1FA0 MSTATUS        1F4A MXIO
 1F4A MINIT          1F61 OPENFORREAD    1FD2 MSTART         1FCE MCERRENT
 1F83 MCLREAD        1FD0 MSTOP          1FB5 MOVECURRENT    1FC0 DECCURRENT
 1F9F MGETRTS        1FB1 MSTOK          1FC8 DECLOW
```

Program 2.

```
100 REM listing of a program to print token values
101 REM and their ATASCII equivalents
200 ? "The STATEMENT Token List" : ?
210 ADDR = 42161 : SKIP = 2 : TOKEN = 0
220 GOSUB 1000 : REM call the token printer
300 ? "The OPERATOR Token List" : ?
310 ADDR = 42979 : SKIP = 0 : TOKEN = 16
320 GOSUB 1000 : REM again call to print tokens
400 END

1000 REM Subroutine to print a keyword table
1001 REM On entry:
1002 REM   ADDR = the address of the keyword table
1003 REM   SKIP = number of bytes to skip
1004 REM          between keyword strings
1005 REM  TOKEN = the starting token number for
1006 REM          this table
1007 REM
1050 IF NOT PEEK(ADDR) THEN ?:?:RETURN

        [note: both tables end with a zero byte]

1060 PRINT TOKEN, : REM the token number
1100 REM Print the ATASCII string for this token
1110 BYTE = PEEK(ADDR) : ADDR = ADDR+1
1120 IF BYTE < 128 THEN ? CHR$(BYTE); : GOTO 1100
1130 PRINT CHR$(BYTE-128) : REM last character
        in keyword has upper bit on
1140 ADDR = ADDR + SKIP : REM an address for stmts
1150 TOKEN = TOKEN + 1  : REM to next keyword
1160 GOTO 1000
```

©

THE OSI® GAZETTE

Part I:

A Small Operating System: OS65D The Disk Routines

T. R. Berger
Coon Rapids, MN

Editor's Note: In this first part of a two-part series, Mr. Berger presents valuable information for all disk drive users. The article concludes next month with a memory map of the disk routines and flowcharts for all the major subroutines. — RTM

In this article I will examine the disk routines in OS65D (V3.2 NMHZ). To understand these subroutines, it is neither necessary to know precise details about the physical functioning of a disk drive, nor to know about various methods of storing data on a diskette. However, such background makes it easier to understand what is involved in an operating system, and why certain processes are done as they are. There are several articles [1-3] which offer very good general descriptions of disk drives. Further, manufacturers' drive manuals usually give fairly complete descriptions of individual drives. I only discuss those aspects which are immediately applicable to the functioning of OS65D.

The Disk Drive

The typical diskette looks as in Figure 1. A magnetically coated round piece of plastic is enclosed in a protective cardboard envelope which has an inner, slippery plastic liner. The hub of the disk drive engages the large hole in the middle of the plastic diskette causing it to spin very rapidly inside its envelope. There is a long slot in the envelope enabling the head of the drive to make contact with the plastic diskette.

Imagine a large number of concentric circles drawn on the plastic diskette so that part of each circumference is visible through the slot. We call each circle a track on the diskette. When the diskette is in the drive, the head is precisely positioned under one of these circular tracks, and contact is made with the diskette. The spinning of the diskette causes this track to continually pass over the head. If we imagine the track to be a continuous loop of magnetic cassette recorder head, we can appreciate how one might store data on the disk. If we envision each track as being a different loop of tape then we can begin to see the power of a disk drive.

In some minifloppies, inserting the diskette and closing the drive door brings the head into contact with the diskette. On other drives, there is a little lever with a soft pad attached directly above the head which is below the diskette. On drives with such a lever, there is a switch which causes this slapping. Turn the switch on and the head engages the diskette; turn it off and the head loses contact with the diskette.

The head can slide back and forth along the long slot in the diskette accessing all the concentric tracks on the diskette. This sliding motion is generally accomplished in one of two ways. The head may be on a screw. Spinning the screw one way or another moves the head in or out. The head may be on a flat metal band which is looped over some shafts, or it may be on a wire which is wound around some shafts. Spinning a shaft causes the head to move. The slide rule dial on most radios works by a similar principle: i.e. the dial pointer is mounted on a string strung over pulleys and wound around the tuning knob shaft. Twisting the knob moves the pointer across the dial. Thus the back and forth motion of the disk head is caused by the turning motion of a motor shaft.

Since the tracks on a disk are very close together, the motor only needs to spin a small fraction of a revolution in order to move the head one track. Very special motors called stepper motors are used for this purpose. When the motor is pulsed, it spins a fixed fraction of a revolution then stops. If pulsed again, it will spin that same fraction again. Clockwise or counterwise motion of the motor shaft translates into back or forth motion of the disk head. Consequently, there are two switches which control this motor: one to determine direction, the other an ON/OFF switch. If we set the direction switch as desired then flick the ON/OFF switch first to on then to off, the disk will move one track.

If we have a memory location in the computer which tells us the track number (say, Track 27) on

which the head is currently positioned, and we move the head outward one track on the diskette (the outermost track is Track 0) then we may decrease the number in memory by one (to Track 26). In other words, we may move from any track to any desired track just by stepping and counting. A single step occurs very rapidly, more than 100 steps per second are usually possible. Of course, this stepping method will only work if we know the current track number on which the head is located.

Most drives have a special indicator to tell when the head is positioned over the outermost track (Track 0). Moving the head out until this indicator comes on allows us to set a track counter to 0 precisely when the head is at Track 0.

When the head is down on a particular track, several operations are possible. The head can read (playback) data from the diskette, or it can write (record) data on the diskette. In addition, an erase function can be switched on. If we erase only, the track will be erased. However, if we erase and write at the same time, the erase function narrows the data stream keeping it from widening into neighboring tracks. The disk has a switch which causes the head to write if on and read if off. An additional switch turns the erase function on and off.

If you look down on some spinning circular object (e.g. a turning phonograph record), you will see that the outer edge is moving much more quickly than any inner part. In particular, on a diskette, each track moves at a different speed past the head. These radical changes in head speed from inner to outer tracks pose difficult problems in obtaining uniform recordings on all tracks of the diskette. Some drives compensate by having two possible recording levels: one for inner tracks, the other for outer tracks. A switch is needed to move between these two modes.

If you own more than one drive, there are switches which allow the computer to select any one of these drives.

In Table 1, under CONTROL LINES, you will see that the computer has a bit to control each of the switches just described. Other than a serial port through which data flows and its associated control location, these are all the control lines used by OS65D to run the disk.

As already mentioned, there are also STATUS LINES to the computer which indicate current conditions at the disk. There is an indicator to tell

Table 1.

DISK STATUS LINES

PA0	DRIVE 1 READY
PA1	HEAD AT TRACK 0
PA2	FAULT INDICATOR
PA3	SECTOR HOLE
PA4	DRIVE 2 READY
PA5	DISK WRITE PROTECTED
PA7	INDEX HOLE

DISK CONTROL LINES

PB0	ENABLE WRITE FUNCTION
PB1	ENABLE ERASE FUNCTION
PB2	STEP MOTOR DIRECTION (IN)
PB3	STEP MOTOR ON (OFF)
PB4	FAULT RESET
PA6	DRIVE ½ SELECT
PB5	DRIVE ½ SELECT
PB6	SET HEAD RECORD CURRENT TO LOW
PB7	PUT HEAD ONTO DISKETTE

The disk PIA has two ports 'A' and 'B'.

PORT A	$C000 (with bits PA0-PA7)
PORT A CONTROL REGISTER	$C001
PORT B	$C002 (with bits PB0-PB7)
PORT B CONTROL REGISTER	$C003

The disk has an ACIA

SERIAL PORT	$C011
STATUS/CONTROL REGISTER	$C010

OS65D configures this port for 8 bit bytes with even parity and 1 stop bit ($58).

Table 2.

8 INCH FLOPPY TIMING

#Sectors	Total Pages	Pages Last Sector	Time	DT
1	13	13	162768	3900
2	13	10	166203	464
3	13	10	166638	29
4	12	1	163209	3458
5	12	1	163144	3023
6	12	1	164079	2588
7	12	1	164514	2153
8	12	1	164949	1718
9	12	1	165384	1283
10	12	1	165819	848
11	12	1	166254	413
12	12	1	166689	-22

t(us.) = 8101 + 12864xp - 1000xr + 435xn

p = number of pages in track
r = number of pages in last track
n = number of sectors
166667 us. = time on one track
DT = time left on track

MINIFLOPPY TIMING

# Sectors	Total Pages	Pages Last Sector	Time	DT
1	8	8	193986	6014
2	8	3	199641	359
3	8	4	199296	704
4	8	4	199951	49
5	7	1	179478	20522
6	7	1	180133	19867
7	7	1	180788	19212
8	8	1	205571	-5571

t(us.) = 8307 + 24128xp - 1000xr + 435xn

200000 ux. = time on one track

if a drive is ready (i.e. if the drive door is closed indicating a diskette is mounted and ready). There may be an indicator to tell if a diskette is write protected. Finally there is an index hole detector. These indicators are all listed in Table 1 under STATUS LINES. You will see a few more than mentioned here. These are not used by OS65D.

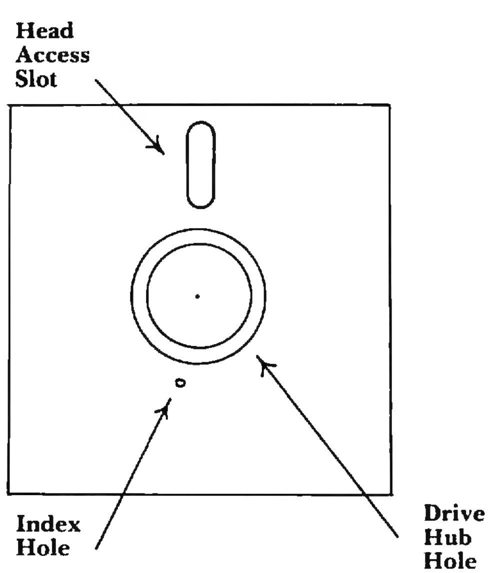
Let's examine the function of the index hole a little more closely. In Figure 1 you will see a small, off-center hole punched in the diskette. (It is off center to prevent functioning if the diskette is inserted into the drive wrong side up.) As the diskette spins, the drive detects when this hole passes over a special indicator. This passage marks the beginning of a track. To find the beginning of a track, the computer moves the head to a track, puts the head on the diskette, and waits for the index hole to flash by.

Once the index hole has passed, the data format on the diskette becomes important. The few methods for encoding data magnetically on the diskette are standardized and adhered to by almost all drive manufacturers. Thus one bit sent to BRAND X drive will be recorded in just about the same way as one bit sent to BRAND Y (i.e. the bit will be recorded in one of about three standard ways). There are a few exceptions to this rule.

This standardization allows computer manufacturers to use drives from different disk manufacturers on the same computer. OSI supplies computers with Shugart, Siemans, and other drives without explicitly telling the buyer which drives he is getting.

Most computer manufacturers send bits from their computers to disk drives as a steady stream of bits, eight bits per byte, and a fixed number of bytes per stream. At the end of a stream are two more bytes called a checksum of cyclic reduncancy

Figure 1.



check (CRC). These two bytes are usually the sum of all the previous bytes in the stream. On reading the stream, the checksum can be recomputed from the stream and compared with the checksum recorded on the diskette at the end of the stream. If there is a mismatch, an error has occurred somewhere in the stream.

OSI does not follow this format. They treat disk communication as an asynchronous communication line. In other words, except for the speed of the bits, the computer sends bits to the disk drive in the same way it sends bits to a modem: through a special serial port called an Asynchronous Communication Interface Adapter (for short, an ACIA, UART, or just serial port). OS65D requires 11 bits to be recorded on the diskette for each eight-bit byte. The first bit is a start bit indicating that the byte is beginning. The next eight bits (bits 2-9) are the actual data byte. The tenth bit is a parity bit indicating whether the byte contains an even or an odd number of value one bits. The last bit is a stop bit indicating the end of the byte.

The disadvantages of this method are twofold. First, it is nonstandard. OSI owners cannot interchange disks made by computers of other manufacturers. Second, OSI can store only 8/11 as much on a disk as other manufacturer's computers.

The advantages are reliability and simplicity. An inexpensive ACIA performs many chores simplifying software and hardware. No cyclic redundancy checks are needed. Each byte can be individually checked for an error by the ACIA. If there is a disk error, usually all but a few bytes can be recovered correctly using the EXAMINE command of OS65D. Other systems make recovery much more difficult. A bit error can cause all bits in a stream to shift by one. In other words, bit two of a byte may be read as bit one, and bit zero of a byte may be read as bit seven in the previous byte. OS65D does an excellent job of error detection. It is a shame that, in a system with such excellent opportunities for error recovery, OS65D has absolutely none. If BASIC encounters a disk error, a program stops with a terse error message.

Track Format

Figure 2 gives the actual data format for an OS65D diskette track. Note that the Track 0 format differs from all other tracks. In particular, Track 0 can only be used by the bootstrap ROM. Track 0 contains the major portion of OS65D and is given added protection by this scheme, but I believe OSI blundered in choosing this format. All tracks should be recorded the same way to maximize flexibility.

The data on a track commences 1 ms. past the index hole (about 23 bytes in time at 44 us./byte). Two bytes are written to indicate the beginning of

Figure 2.

FORMAT FOR TRACKS (>0)

Index Hole	1ms.	$43	$57	Track #	$58	6615 us.

... | $76 | Sector # | # Pages in Sector | That many Pages of Data | ...

... | $47 | $53 | Intersector Wait Time | ... | Repeat for each Sector

FORMAT FOR TRACK 0

Index Hole	1ms.	Load vector high	Load vector low	#Pages

... That many Pages of Data ...

a track. The bytes should be carefully chosen so as to be an unusual combination. OS65D always writes $43 then $57. When the track is read, reading does not commence until the $43 and $57 have been found. A simple encryption method would be to change these bytes. Since the EXAMINE command will even read such a track, this encryption is not terribly secure. OS65U uses different bytes, so OS65U tracks cannot be read by OS65D without minor changes to the operating system.

Next the track number is written in binary coded decimal (BCD). This recorded value is always compared with the stored track number in memory to make certain the head is positioned on the correct track. Then a stop byte ($58) is recorded on the disk (this byte is never checked on a read).

This data constitutes the Track Header. On Initialization, a track is erased then the Track Header is written on the Track. This Track Header is not rerecorded at any future read or write.

There is a lull after the Track Header of just under 6.6 ms. (about 149 bytes). This time differs greatly from the time given in the OS65D GUIDE. You will see why in the following discussion.

During a sector seek operation, a "previous sector" length number p is saved. This value is set to four if we seek Sector 1 (otherwise the "previous sector" length number would be zero, which is not allowable). Then a subroutine waits px800 µs. The OS65D GUIDE says that between Sector N and Sector n + 1 there is a gap of px800 µs. This is not quite correct. After the end of a sector, OS65D waits quietly for px800 µs. The write function is then switched on. A further 185 µs. is allowed to pass. Then the erase function is switched on. We now wait an additional px800 µs. before starting to write data. In other words, the time from the last byte of sector n to the first byte of sector n + 1 is about px1600 + 185 µs. For Sector 1, p is taken to be four. In all other cases, p is the length (in pages, i.e. multiples of 256 bytes) of the "previous sector."

This description requires modification. It applies to systems with a 1 MHZ clock. On cold start, OS65D measures the timing on a serial port to calculate the clock speed. (Remember, a 300 baud port must remain 300 baud no matter what the clock speed.) Then a timing constant in the 1 ms. subroutine is set. However, this calculation does not affect the 100 us. routine used in sector spacing. (I assume this clock versatility is the reason for the NMHZ in the title of this version of OS65D.) In other words, the 100 us. routine is really a 100/T us. routine where T is the clock speed in MHZ.

This calculation accounts only for the wait loops in intersector timing. In addition, there is quite a bit of inline code which adds to intersector timing. This timing can be calculated. A crude estimate would be to add an additional 30 µs. after each sector. In other words, the sector spacing is

(px1600 + 215)/T

where T is the clock speed in MHZ, and p is the number of pages in the preceding sector. Your disk does not necessarily write diskettes identically with mine, though either computer should read the other's diskettes.

All of this says there is some kind of empty space between the end of the Track Header and the start of Sector 1. Each sector is completely rewritten each time it is addressed in a write operation. A sector is written as follows.

We put a sector start code ($76) on the disk. Next comes the sector number s, then the sector length p in pages (each page is 256 bytes). The smallest unit of disk storage in OS65D is one page. The sector number s is verified on a read operation with the value in memory. The sector length is used on read to calculate the number of bytes to load from the disk.

Now comes the actual data. The amount of data is px256 bytes where p is the number of pages in the sector. After this data comes two end check bytes ($47, $53) marking the end of a sector. Thus the sector if 5 + px256 bytes long. The gap between sectors has already been described. Each succeeding sector follows the same format. This format is pictured in Figure 2. This discussion does not apply to Track 0.

Before discussing Track 0, let's make a few calculations. We assume we have 8" floppies and a 1 MHZ clock (this latter enters in only for the timing between sectors). We discuss how many and what kind of sectors may be put on a track. The discussion is important for the following reason: on a write operation, OS65D checks for the index hole when seeking a Track Header. This keeps the computer from "hanging" on uninitialized tracks (i.e. tracks without a Track Header). In writing sector n, the computer must read the preceding sectors 1,2,...,n-1. For each of these, while the computer is searching for the sector start code, it also watches for the index hole to come around again (also to avoid "hanging" on a sector seek). After the start of the preceding sector, the computer no longer checks to see if we pass the index hole. The reason for this is simple. At 1 MHZ with 8" floppies there is just not enough time between input or output bytes from the disk to check for the index hole and to do all the other operations required during a read or write operation.

If the index hole passes, we are back to the beginning of the track. If 1 ms. passes, we're over the Track Header again. Obliterating the Track Header destroys the readibility of the Track. Experienced programmers may salvage matters using the EXAMINE command, but this is not a task you want to face. *Moral:* Don't pass the index hole a second time on a write operation.

If you're not a whiz at algebra, skim over this part until we start drawing conclusions.

We wish to derive a formula for the time from the index hole to the time the head stops writing on the diskette after sector n. If this time occurs before a second appearance of the index hole, then n sectors will fit on a track. We must account for all the time from the first appearance of the index hole until the write function is switched off after the last sector.

The disk spins at 360 rpm. Thus one revolution takes 166,667 µs. The disk data clock runs at 250 KHZ. In particular, each bit takes four µs. Since an OSI byte uses 11 bits, 44 µs. are required per byte. If we could pack a track, this means we could fit 3,787 bytes on a track. But a track is not packed. It is formatted, and we must calculate the formatting time.

We use 1000 us. from the index hole to the Track Header. The Header is four bytes long using 176 us. more. As we have seen, from the Track Header to the start of Sector one, we use 4x1600 + 215 or 6615 us. In particular, 7791 µs. are spent between the index hole and the start of Sector one.

Each sector contains an integral number of pages. Thus, all sectors contain, as an aggregate, *p* pages. Each byte takes 44 us. and there are 256 bytes per page. Thus all these pages account for 11264xp us.

Each sector has five extra bytes. Thus, for *n* sectors, we have 220xn µs.

Next we must account for all the wait time after each of the *n* sectors. Recall that the wait from one sector to the next is qx1600+215 µs. where *q* is the number of pages in the preceding sector. Since we assume *n* sectors are on a track, there are only *n*-1 spaces between *n* sectors. If the last sector has *r* pages, then the preceding *n*-1 contain *p-r* pages altogether. Thus, the total inter-sector wait time is 1600x(p-4)+215x(n-1) µs.

Finally, we must account for the time after the last sector is written until the write and erase functions are switched off. Write and erase continue for 600xr us. after the last byte is written. Then write is switched off and erase continues for 525 µs. more before it too is switched off. This total trailing time is 525+600xr µs.

By adding all our derivations, we can make the following statement. For 8" floppies with a one MHZ clock, the total recording time for *n* sectors is

t(µs.)=8101+12864xp-1000xr+435xn

where *p* is the total number of pages of data in the sectors and *r* is the number of pages in the last sector.

Remember, OS65D must run on all OSI machines, so this formula gives the "worst case" which must always be satisfied. In Table 2 you will see a few 'upper limit' values tabulated (dt gives the 'time remaining' in the track).

Recalculate *t* for your system. A minifloppy spins at 300 rpm. and the data clock is 125 KHZ. Experiment with a few values for *n* and *p* in the formula. Try actually recording this amount on a disk. Be sure to use an empty diskette track. What is wrong with filling the blank space between the index hole and the Track Header with data? (Think about $43, $57.)The maximum allowable number (plus one) of pages per sector in OS65D is stored in $27ED. You may wish to change this for your experiments.

Notice that OSI recommends a maximum of 13 sectors when only one sector is written on a track, and eight sectors (12 sectors in early GUIDES) if more than one sector is written on a track. The early GUIDE value is "just barely wrong." The later value is obviously a shot in the dark meant to be conservative. It is probably the case that many drives would accept 12 single page sectors in a track. But even 11 sectors, including 12 pages, leaves very little room for errors. ©

THE PET® GAZETTE

A Yuletide Tale

Editor's Note: We recently received the following query letter from a Mr. C. Pickins. While we will not be able to accept C's fictional endeavors, we felt his timely outline might be of interest to our PET/CBM readers. We have put C (not his real name surely) in touch with Dr. Chip. Perhaps we'll see the fruit of later joint collaboration. — RCL

Dear Mr. Lock,

I have this great story outline that I thought **COMPUTE!** might like to follow up. It's just that I don't know if you publish fiction submissions.

It's this heartwarming story about a fellow called Scrooge Tramiel, who runs a pet shop in old London, or California, or Philadelphia or somewhere (funny, the location seems to shift every time I think of it). Anyway, he exhorts his lowly clerk, a fellow by the name of Cratchit Finke, to work through the holidays on a new computer system to be called the Humbug III, and leaves to go home.

Well, what should happen but this guy runs across the ghost of his former partner, Jacob Peddle, who rattles chains and chips and emits fearful moans in all directions. The upshot of this visitation is that Scrooge is going to be visited by three more spectres: the barrister of Christmas past, the solicitor of Christmas present, and the lawyer of Christmas future.

Faster than you can say, "restraining order", the Christmas past fellow pops up and reminds Scrooge of how helpful he used to be to others. "Spirit, why do you torture me so?" wails Scrooge. "Yes, I gave that young fellow a chip to play with ... and he promptly founded Apple Computers with it. Indeed, Radio Shack got its start in the time period between when I announced my computer and started delivering it."

Just then, the digital clock beeps and along comes Christmas present. Not a Christmas present, you understand; just the Ghost of a Christmas present, who shows retailers warming themselves over the glow of their CRT screens. "Everybody else's model three has failed," they seem to be saying, "will Scrooge come through for us?"

An announcement of the digital cuckoo clock heralds visitor number three, the spirit of Christmas future. The screen swirls uncertainly ... coughing and gasping, Scrooge peers through the orange smoke ... is that IBM gaining credibility? What's going on here? The Spirit intones, "I see unused joysticks by the fireplace..." But begging and pleading

and threats of countersuits reveal that it ain't necessarily so ... there's still time to reform.

Dawn is breaking. Maybe the light is dawning, too, for Scrooge rushes over to the window and shouts at a passing boy, "Bring me the biggest goose you can find! My competitors have all the turkeys!"

And the story ends with a traditional Christmas scene ... as Tiny Tim says, "God help us, every one!".

Whaddya say, Mr. Lock? Do you think you can use the story? ©

Renumbering An Appended Routine Only

Elizabeth Deal
Malvern, PA

There is a way to append a program to another in the PET even if the line numbers are out of order. It will be shown here for the upgrade ROM tape system. It should work on other PETs. The scheme uses the Toolkit™ or its equivalents.

Suppose that program A exists in the PET and that its line numbers range from 100 to 2000. Suppose, further, that we would like to append a program B with line numbers which are lower than (or overlap) those of program A: for instance 15 to 340. As long as program B contains no GOTOs and no GOSUBs, one renumbering of the entire A-plus-B package will set the line numbers in order. Consequently, target addresses in program A will remain meaningful.

When, however, program B contains GOTOs and GOSUBs, we are in violation of the "appended program must have higher line numbers" rule. And that means save one piece, put the other one in, renumber it, save again, load again ... ad infinitum. Disk people can do it in a jiffy. It's tough for tape owners though.

Some rules just beg to be broken and this is one of them. A simple solution consists of temporarily hiding program A from PET's view by swapping some pointers around. Just before loading program B we tell the PET that the BASIC area begins at the end of program A or *exactly* two (2) bytes back from the start of variables pointer (42-43). We do this carefully by use of the Machine Language Monitor where we replace contents of $28-29 with contents of $2A-2B minus 2. Or by these direct BASIC commands:

```
AD=(PEEK(42)+PEEK(43))-2: AH%=AD/256
POKE40,AD-AH%*256:POKE41,AH%
```

This has to be entered correctly the first time or things get somewhat messy.

At this point we can append program B. It will be placed, in the usual manner, at address AD. We can list this program. And we can RENUMBER it, for instance, with 3000,10 parameters sent to the TOOLKIT.

To finish the process we reset the start of BASIC pointer to its original value, decimal 1025, hex $0401, or whatever other number we have jotted down in case of being in a partition. In BASIC, the reset can be done by:

```
POKE40,1:POKE41,4
```

Using the Monitor, the reset to 1025 decimal is done by putting $01 into $0028 and $04 into $0029.

Program A reappears on the scene and the entire package is ready for use.

Pointer addresses for various releases:

	Original		Upgrade and 4.0	
Start of Basic low byte	122	$7A	40	$28
high byte	123	$7B	41	$29
Start of variables low byte	124	$7C	42	$2A
high byte	125	$7D	43	$2B

©

chips . . . chips . . . chips . . . chips . . . chips . . . chips . . .

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

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.)

"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 (Please indicate PC2, PC4, PC8)	$60.00
Complete with Socket-2-ME (Please indicate PCB2, PCA4, PCA8)	80.00

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 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 MK 4/8 for 4000 or 8000 Series PETs and CBMs	$80.00
Complete with **Socket-2-ME**	95.00

Skyles Catalogue Page

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 South Whisman Road
Mountain View, CA 94041
(415) 965-1735

BRANCH NEVER And QUIF Assembling On SuperPET

Richard Mansfield
Assistant Editor

Ever hear of QUIF? Or HI, ISUPPER, STOI, FSEEK, TABLELOO, COMA, ORB, PULB, SEX, COMB, or BRA? These are some of the 6809 mnemonics, utilities library macros, and "structured programming" statements available to you when you assemble on the SuperPET. The Waterloo 6809 Assembler permits machine language programming which is somewhat like programming in higher level languages. Along with the Assembler is an Editor, a Linker (to connect modules), and a monitor.

Making the transition to this assembler involves two major adjustments: you are now working with a 6809 and you are using a complicated assembler. If you are accustomed to working with simple assemblers (Supermon, Extramon, Micromon, or others), you will be baffled at first by the requirements of this assembler. Before looking into the significant differences between 6809 and our familiar 6502, let's see what is required if you decide you want to place the letter "a" in the upper left corner of your screen.

Simple 6502 Version:

```
0360  LDA #$41
0362  STA $8000
0365  BRK
```

Waterloo 6809 Assembler Version:

```
lda #'a
sta $8000
swi
end
```

SWI means software interrupt and resembles BRK on the 6502. (There are three software interrupts available: SWI, SWI2, and SWI3.) The apostrophe allows you to enter the actual letter which will be translated into the correct value for you. Otherwise, it's fairly simple at this point. You are in the Editor here (no need for addresses yet — they will be created later). The creation of your final, "object" code takes several steps: you must save this "file" to disk by typing p (for PUT) *name.asm*. Then, when the ASM file is on disk, you type BYE to get into the menu and select *a* (assemble) and you are asked for the filename, so you type: *name*. (It adds the ".asm" for you.)

The assembler makes two new files on the disk: *name.4 st* and *name.b09*. The first is a fairly straightforward listing of the source code with line numbers, object codes, mnemonics, and any comments separated into appropriate fields on screen. *Name.b09* is a file containing the object code to be used later by the Linker.

Your next step is to return to the Editor and make a fourth file:

```
"name"
org $1000
"name.b09"
```

and PUT it to disk under the title "name.cmd." The first line here names the "load module," the second line defines the starting address of the object code, and the third line names the object code file to be used in the linking process.

Then you type BYE again, select Linker from the menu, and type: *name*. (The linker will add ".cmd" to the name.) The linker creates two more files (for a total of six): *name.mod* (executable load module) and *name.map* (tells how *name.b09* was mapped into *name.mod*).

Now you are ready to run your program. You enter the monitor by typing "M" from the menu and then type: l name.mod (to load the "module"). You can then type : g 1000 and, voila!, an "a" appears on your screen.

The Monitor And Linker

Like TIM (the resident monitor on PET/CBM computers) the SuperPET monitor has several commands which are useful for debugging (Bank, Clear, Dump, Fill, Go, Modify, Passthrough, Quit, Registers, Stop, and Translate). "Bank" allows you to access any of the 16 banks of upper RAM for reading or writing. "Stop" sets breakpoints and "Clear" clears them. "Dump" is equivalent to "M" on TIM. "Modify" permits the same changes as "Dump," but in the form M ff 12 33 (where the byte at $00ff now becomes $12, $0100 becomes $33). "Quit" is like TIM's "x." "Passthrough" sends all input to a host computer and permits all output from the host to appear on screen.

"Translate" is a disassembly. Curiously, there is no provision for single-stepping or for SAVEing from the monitor. A single-step program exists (it was used at Waterloo to create the SuperPET languages), but it was not included in the monitor. As for SAVE, it was planned, evidently, that modules should be only created from the upper levels of the development system, following the steps

outlined above which result in six files per module.

The linker knits the relocatable object modules (*name.b09*) into longer executable load modules. The linker is invoked by creating the *name.cmd* file mentioned above and including various commands in this file. "Org" specifies the desired starting address for the code. "Banksize" defaults to $1000 if not specified and "Bankorg" defaults to $9000. Programs or modules may be loaded into specified banks with the "Bank" command. To merge external routines from the system library (or from your personal library of modules), use the "Include" command. Finally, "Export" sets aside some memory (Export bytespace = $7b00) which is named "bytespace" and reserved for tables, etc. Following its definition, "bytespace" can be referenced by any routine using the statement: xref bytespace.

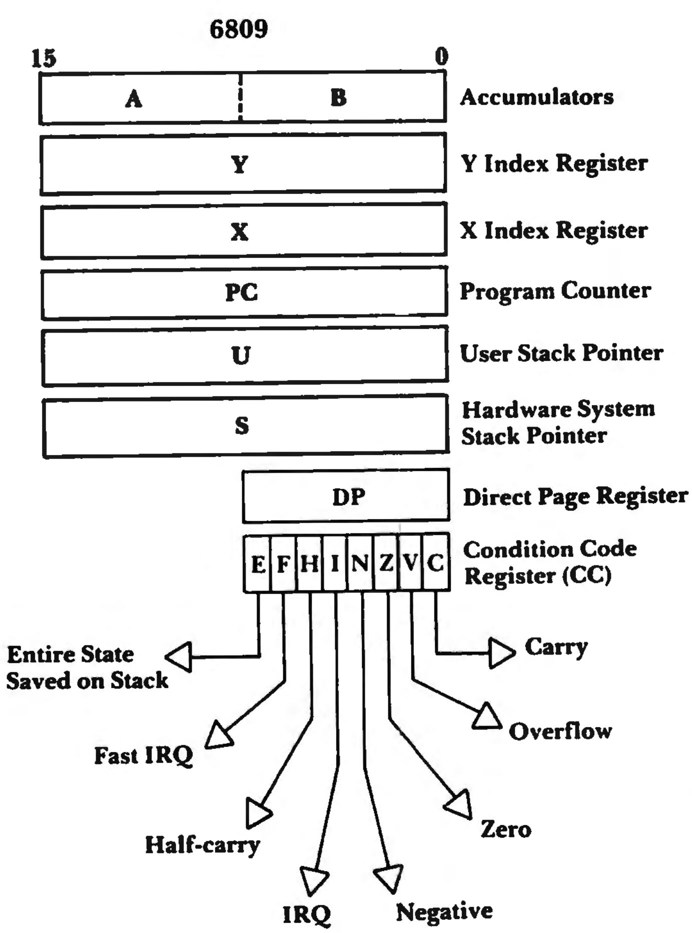
The 6809

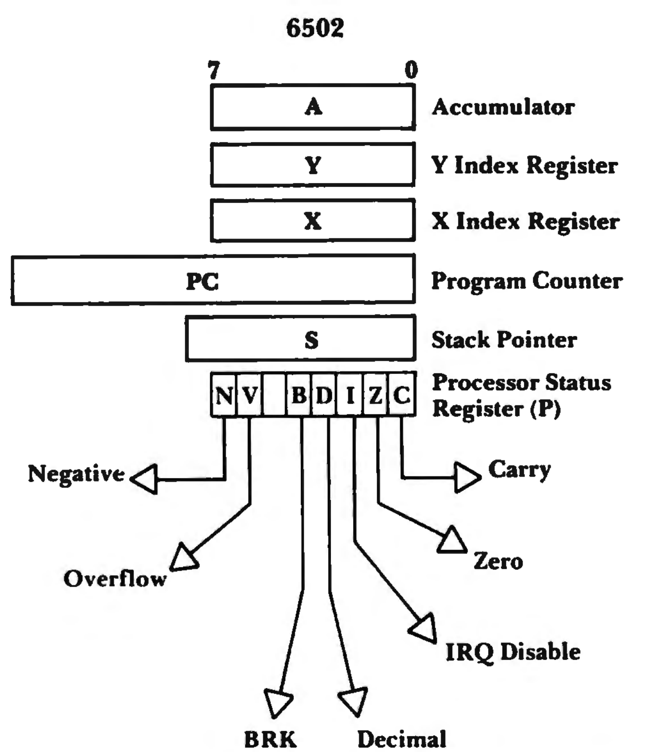
As Figure 1 illustrates, the most obvious novelties in the 6809 are the addition of Accumulator B, the second (User) Stack, a Direct Page register, and half-carry, fast IRQ, and Entire State Saved condition flags. In addition, of course, the Y, X, and Accumulator registers and the stack pointers are expanded to 16 bits. Some of these improvements facilitate simplified addressing since a 16 bit register can address an entire 64K. Likewise, a stack can now be located anywhere in memory and be of any size desired. The A and B Accumulators can be concatenated to form Accumulator D (A is the MSB). This allows 16 bit addition, subtraction, compare, and so forth, via a single mnemonic.

The S stack pointer is used for JSRs and interrupts as expected, but the U stack pointer is controlled completely by the user and is unaffected by hardware status. This permits variables to be passed between routines.

The direct page register (normally 0) is used to form the MSB of an effective address during "direct addressing." The offset is the byte following the direct addressing mode opcode. This is like the familiar zero page addressing, but with the added ability to set "zero" at any page. A half-carry is a carry from bit three during eight-bit addition. There is a fast interrupt request line which can be masked with the fast IRQ flag. The entire-state-saved flag signals that all registers (not simply the program counter and CC) have been saved on the stack.

Figure 1.





Branch Never: Addressing and New Instructions

In addition to the familiar 6502 modes, the 6809 includes "Direct" addressing, "Long Relative" (16 bit relative, position independent), and various indexed and indirect modes including auto-increment and decrement by one or two bytes at a time. The efficiency inherent in 16 bit manipulations, new addressing modes, and new instructions permits greater programming freedom than is possible on the 6502. For example, the 6502 has approximately 56 mnemonics where the 6809 has nearly twice as many. (Mnemonic counts will vary depending on whether such instructions as ROL and ROL Accumulator are counted as distinct instructions.)

Among the more interesting new instructions is SWI (the entire machine state is saved and control is transferred through the vector at $FFFA-B. SWI2 is the same except that the IRQ masks flags are not set and the vector is $FFFA4-5). SEX means sign extended. BRA is branch always. Perhaps the most enigmatic new instruction is BRN, Branch Never. Though hundreds of uses for this spring to mind immediately, the assembler manual suggests that it can be used if you should become tired of NOP.

MUL multiplies accumulators A and B (unsigned) and stores the result in the D (A + B) accumulator. COMA and COMB complement these accumulators. ORB P inclusive ORs the value addressed by P, with B.

Assembler Expressions

The assembler provides for extensive programming options through lables, external references, libraries, macros, operators, conditional assembly, etc. QUIF? It's Quit IF, one of the structured programming statements. HI is a condition which follows QUIF and is true if the carry and zero flags are both clear. Other statements are: IF, ENDIF, ELSE, GUESS, ADMIT, ENDGUESS, LOOP, ENDLOOP, and UNTIL. Like their counterparts in other languages, these statements can be used in the assembler, if that is your preference.

Also, a library of common routines is included and can be called into a program by typing the reference name followed by an "underbar" character, an underline which is created by hitting the back-arrow key. ISDELIM checks to see if the character in question is a delimiter (not alphabetic or numeric). STOI converts a decimal string to an integer. ISUPPER sees if you have an uppercase alphabetic character. FSEEK finds a record in a random file. In all, there are 67 library modules. The first parameter is passed on D, the rest on the stack. Results come back in D.

The "structured programming" statements, 100 mnemonics, 67 library names, 17 addressing modes, 96K, two stacks, 16 memory banks. It's a bit of a transition. Nevertheless, 16 bit addressing, the freedom to MUL at will, and numerous other advantages all combine to make the 6809 option on the SuperPET exciting and promising for machine language programming. ©

PET Repairs For The Amateur

Louis F Sander
Pittsburgh, PA

My small keyboard PET has had several awful-looking symptoms over the past year, and each time I dreaded the size of the possible repair bill. But each time I was able to cure the problem myself, with no need for knowledge of digital electronics. Based on first-hand experience, and on many notes compared with others, here is what to look for when your PET is acting strange: loose connections, period.

Loose connections are probably the most frequent source of trouble in PET-like electronic equipment, and they are often the easiest to find and fix. You'll learn how I found mine, after a few words on safety. First, never look for trouble with your PET plugged in. Under normal circumstances, all lethal voltages are kept away from PET's main circuit board and other exposed parts, but when trouble comes, circumstances aren't normal. So always pull the plug when you're troubleshooting. Also, always take pains to avoid static electricity when you're poking around inside your PET. Tiny sparks that you can't see or feel can ruin some of the IC's in there, so don't take any chances. The best precaution is to ground yourself by touching bare metal on the cabinet whenever you touch an IC or the circuit board; it may look silly, but it's safe. Now for my war stories:

My first trouble was erratic operation. From time to time, I'd get a screen full of garbage, and my cassette motor would run and run. It looked like my reset button was locked down, but I knew it wasn't. On the advice of somebody who knew, I looked for an IC that was loose in its socket. When I found it, the trouble went away. With time and the flexing caused by neat, IC's all tend to walk out of their sockets. If you have symptoms of trouble, check this first. Open your PET and, with one hand touching the cabinet, firmly press down on both ends of every socketed IC, and walk them back into place. You'll be surprised how many are loose. Don't worry too much about flexing the printed circuit board itself — it can withstand a bit of bending.

My second problem came from a bad power connector. I'd lose everything on my screen, right in the middle of something important. At other times, I'd power up and not be able to get anything on the screen at all. When I found a hot power connector, I knew the cause was found. The power connector attaches your main circuit board to the wires coming from the large transformer and electrolytic capacitor at the left rear of PET's base. If you are having problems, especially ones that crop up after some length of 'on' time, run your machine for an hour or so, then feel the power connector. If it's noticeably hot, it is a candidate for replacement. I replaced both ends of my connector with Radio Shack 274-226 and 274-236, for under $3.00 total. If you're not an experienced electronics person, turn this job over to an expert — it's easy, but the new connectors are far from exact replacements.

My biggest and most mysterious problem was caused by a dirty contact on the connector between the main board and tape drive #1. For several months, I'd get strange screen messages and frequent system crashes whenever I tried to load a program that was other than the first one on a cassette. I'd say LOAD "RINKYDINK," the tape would start to move, and then I'd get some horribly misspelled version of ?ILLEGAL QUANTITY ERROR, sometimes before and sometimes after the PET had FOUND the programs preceding RINKYDINK. It got so bad that I gave up on ever being able to put more than one program on a tape. I could tell that the problem was associated with the unrecorded gaps between programs, but that's as far as it went.

I found the problem one day as I connected an audio amp to the tape READ line. The recorder was running a totally blank tape, and the noise on the READ line was tremendous. I accidentally jiggled the wire going from the recorder to the main board, and the noise stopped completely. Later I found that a poor ground contact on the PC board connector was allowing motor noise to get into the signal circuits, and that PET was trying to read the noise as data. No wonder it got an ILEGAL QUANIY ERRR! Two minutes with superfine sandpaper cured the problem, and now I can read through a whole C-60 with no system lockups. Keep your connectors clean.

By the way, I've had one minor problem unrelated to bad connections: My PET likes to read tapes a lot better without any amplifier connected to CB2. I don't know just why, but the machine definitely works better with nothing connected back there. So now I disconnect the amplifier whenever I'm through with a program that uses sound. I guess this really *is* another loose connection problem, but one of a different sort — in this one, loosest is best. But take it from one who knows more about it than he wants to — loose connections are common in your PET, and you can usually fix them yourself. ©

Realtime Clock On Your Pet Screen

Mark L. Robinson

Editor's Note: In the version of Mr. Robinson's clock routine for 4.0 BASIC, the code has been moved up 38 (decimal) from the Upgrade version of Program 1. Add a value of 38 to his POKEs and references for the 4.0 version. — RTM

How many times have you sat down at your computer to fiddle around for a few minutes, returning to the real world hours (or days) later. This is not always a problem and I don't mind being splattered with cold suppers, missing parties, or aggravating my wife. But one night, I had some free moments to ponder the problem of losing track of time. Wouldn't it be great, I thought, if I could always have the correct time on the screen.

I knew that my PET had a 1/60 second counter which is updated during the internal interrupt cycle and some routines to print out the time. I started to study the memory map in Osborne's PET/CBM Personal Computer Guide and found the following items: jiffy memory, clock correction routine and the location of the interrupt addresses.

I figured if I could revector the interrupt through a small machine language program, I could capitalize on all three items - the jiffy clock to keep track of time, the interrupt addresses to return to the correct location and the clock correction routine to make up for the lost time of my program, if necessary.

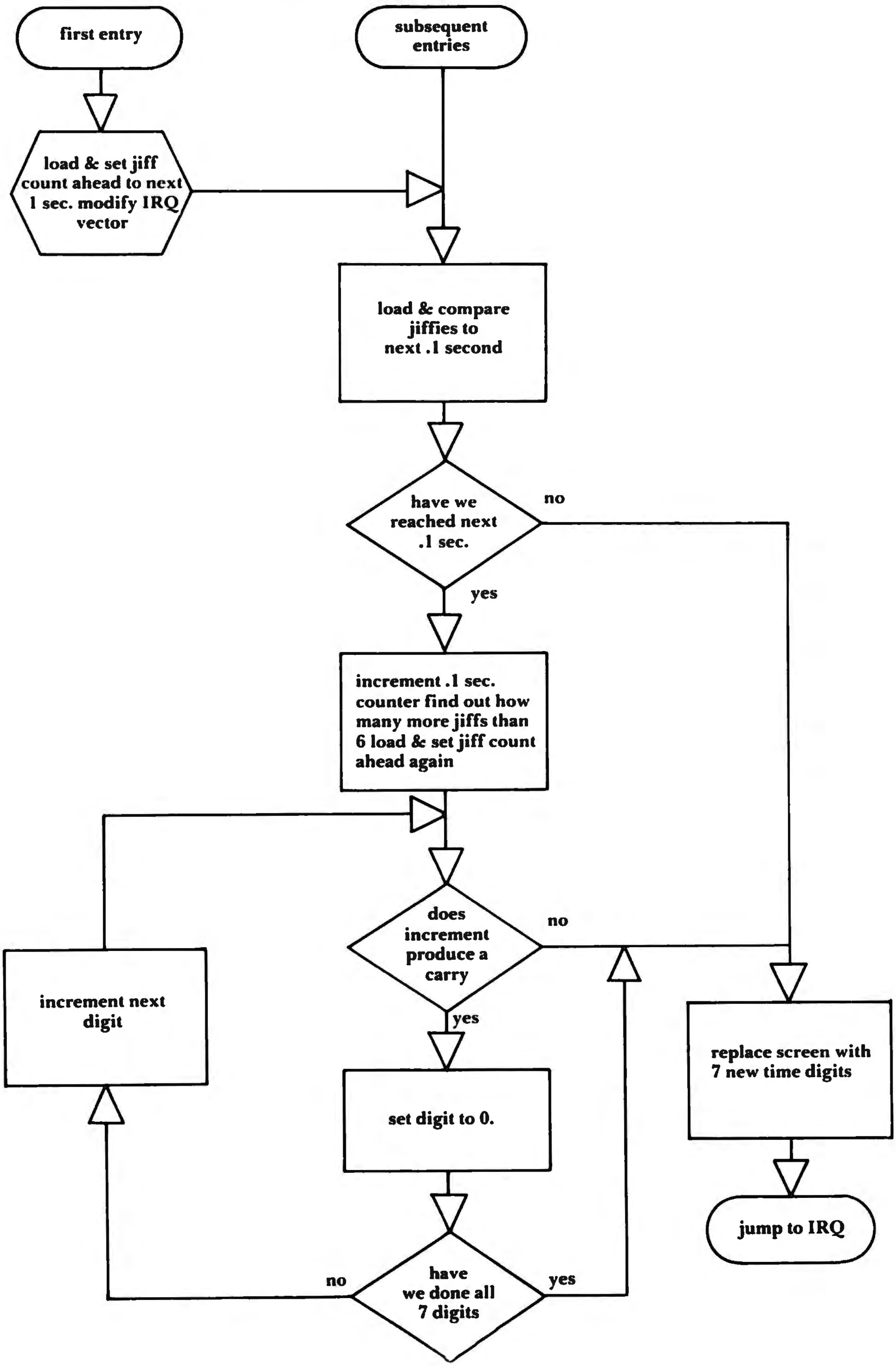
Some other investigation showed that the routines that print the TIME$ use a lot of processing time and interact with memory locations that Basic uses. I figured it would be best to handle it completely as a separate little program. Then, the more I thought about it, I realized that once the time was set, I could follow it with a simple series of little counters rather than keep having to do long divisions. This also has the advantage of being able to jump back to the normal program whenever there is no carry up to the next most significant clock digit. This saves over 50 percent of the time penalty of the screen clock.

To initialize the clock and load the machine language program, I wrote a small BASIC program. You can follow the explanations in the flowchart and the symbolic listing of the machine language program along with the listing of the BASIC program. The machine language program is short enough to load with pokes rather than entering it using the machine language monitor. You can enter and run this as a normal basic program and, while the clock is running, you can use most BASIC programs. There is a small time penalty to use this while running BASIC, but if you are programming or game playing, it is not critical.

Incidentally, since the program is synchronized with the jiffy counter, you are automatically using the PET's internal correction routine. On a three hour run against a stop watch, the PET gained two seconds (so much for my stop watch). Two words of caution when you are writing programs: first, if you hit return on the line the time is on it will be entered in the listing and, second, if you have to load a program from the cassette, turn the clock off (POKE 144,46:POKE 145,230), load the program, and start the clock again (POKE 144,74:POKE 145,3). To reset the clock poke the correct time digits to locations 833-838.

Symbolic Listing
Real Time on Screen

DEFINITIONS	
	LOTB = Least significant time bit — (Jiffy Counter)
	TL0C1 Temporary holding location
	TL0C2 ' ' ' of prior jiffy count
	BASE 1-7 Base of count, 10 or 6
	IMAGE 1-7 Location of time in memory
	SCT 1-7 Screen locations of time
INITIALIZE	
LDA LOTB	Initialize prior count set it ahead
ADC #05	to next .1 second. Note 1
STA TL0C2	
LDA #Start	Revector interrupt to start
STA IRQ Low	
START	
LDA #LOTB	Check jiffies see if we've reached
STA TL0C1	next .1 sec
CMP TL0C2	
ADC #05	Yes-set TL0C2 for next .1 sec, make sure
SBC TL0C2	that if more than 6 jiffs
ADC TL0C1	occurred we do not add too much
STA TL0C2	
INC IMAGE,7	Increase .1 sec memory location by 1
LDX #07	Initialize counter routine
COUNTER	
LDA IMAGE,X	Check to see if we've reached limit
CMP BASE,X	of base which produces a carry
BNE UPDATE INIT	No — then go to Update Init
LDA #00	Yes — place 0 in digit position
STA IMAGE,X	
DEX	Go to next number in sequence
BEQ UPDATE INIT	If we have done all 7 digits go to screen update
INC IMAGE,X	Increment next digit by 1 (result of carry)
JMP COUNTER	Go back and check this digit for carry


first entry
subsequent entries
load & set jiff count ahead to next 1 sec. modify IRQ vector
load & compare jiffies to next .1 second
have we reached next .1 sec.
no
yes
increment .1 sec. counter find out how many more jiffs than 6 load & set jiff count ahead again
does increment produce a carry
no
yes
increment next digit
set digit to 0.
have we done all 7 digits
no
yes
replace screen with 7 new time digits
jump to IRQ

UPDATE INIT	
LDX #07	Initialize the screen update routine
UPDATE	
LDA IMAGE,X	Load time digit
ADC #$30	Convert to PET number code
STA SCT,X	Store on screen
DEX	
BNE UPDATE	Have we done 7 digits? — no go back to update
LDA #3A	Yes — load and store colon on screen
STA SCT,0	
JMP IRQ	Return to PET IRQ routine

Note 1: The reason that five is added to the jiffy count and not six (to get the next 6/60 or .1 sec) is that we are incrementing when the prior count location is less than the jiffy count. If the increment occurred on equality then you would add six. The reason for this is that I do not know if the jiffy count can count two sometimes in which case the equality would not occur for up to 12.8 seconds — when the same binary digit again occurred. This is also the reason the program checks for more than six counts.

Program 1: Upgrade Version

```
5 REM REAL TIME ON PET SCREEN
6 REM C M. ROBINSON 1980
7 REM OK FOR PERSONAL USE
10 FORA = 1 TO 100
20 READ B
30 POKE 825+A,B
40 NEXT
100 PRINT"{CLEAR}             HHMMSS"
110 INPUT"TIME";A$
120 TI$=A$
130 FORA=1 TO 6
140 D=VAL(MID$(A$,A,1))
150 POKE832+A,D
160 NEXT
200 POKE144,74:POKE145,3
250 NEW
1000 DATA10,10,6,10,6,10
1001 DATA10,0,0,0,0,0
1002 DATA0,0,87,90,165,143
1003 DATA105,5,141,73,3,169
1004 DATA85,133,144,165,143,141
1005 DATA72,3,205,73,3,48
1006 DATA38,105,5,237,73,3
1007 DATA109,72,3,141,73,3
1008 DATA238,71,3,162,7,189
1009 DATA64,3,221,57,3,48
1010 DATA14,169,0,157,64,3
1011 DATA202,240,6,254,64,3
1012 DATA76,111,3,162,7,189
1013 DATA64,3,105,48,157,31
1014 DATA128,202,208,245,169,58
1015 DATA141,31,128,76,46,230
1016 DATA0,0,0,0,0,0
```

Program 2: 4.0 Version

```
10 FORA=1TO100
20 READB
30 POKE863+A,B
40 NEXT
100 PRINT"{CLEAR}                HHMMSS"
110 INPUT"TIME";A$
120 TI$=A$
130 FORA=1TO6
140 D=VAL(MID$(A$,A,1))
150 POKE870+A,D
160 NEXT
170 POKE144,112:POKE145,3
180 NEW
864 DATA 10, 10, 6, 10, 6, 10
870 DATA 10, 0, 0, 0, 0, 0
876 DATA 0, 0, 87, 90, 165, 143
882 DATA 105, 5, 141, 111, 3, 169
888 DATA 123, 133, 144, 165, 143, 1
    41
894 DATA 110, 3, 205, 111, 3, 48
900 DATA 38, 105, 5, 237, 111, 3
906 DATA 109, 110, 3, 141, 111, 3
912 DATA 238, 109, 3, 162, 7, 189
918 DATA 102, 3, 221, 95, 3, 48
924 DATA 14, 169, 0, 157, 102, 3
930 DATA 202, 240, 6, 254, 102, 3
936 DATA 76, 149, 3, 162, 7, 189
942 DATA 102, 3, 105, 48, 157, 31
948 DATA 128, 202, 208, 245, 169, 5
    8
954 DATA 141, 31, 128, 76, 85, 228
960 DATA 0, 0, 0, 0, 0, 0
```


Statement of Ownership, Management And Circulation
As required by 39 U.S.C. 3685

1. COMPUTE! The Journal For Progressive Computing
1a. 337250
2. 12-1-81
3. Monthly 3A. 12 3B. $20.00
4. 625 Fulton Street, P.O. Box 5406, Greensboro, NC 27403
5. Same
6. Robert C. Lock, Publisher and Editor, 625 Fulton Street, Greensboro, NC 27403
Kathleen Martinek, Managing Editor, 625 Fulton Street, Greensboro, NC 27403
7. Small System Services, Inc.: Robert C. Lock, William E. Knox, Joretta F. Klepler, Marlene R. Pratto; P.O. Box 5406, Greensboro, NC 27403
8. None
9. NA

10. Extent and nature of circulation	Average no. copies each issue during preceding 12 months	Actual no. copies of single issue published nearest to filing date
A. Total No. Copies (Net Press Run)	31,961	40,040
B. Paid Circulation		
1. Sales through dealers and carriers, street vendors and counter sales	15,232	19,491
2. Mail Subscription	10,516	17,773
C. Total Paid Circulation	25,748	37,264
D. Free distribution by mail, carrier or other means: samples, complimentary and other free copies	1,128	350
E. Total Distribution	26,876	37,614
F. Copies Not Distributed	3,397	2,426
1. Office use, left over, unaccounted, spoiled after printing		
2. Return from news agents	1,488	None to date
G. Total	31,961	40,040

I certify that the statements made by me above are correct and complete. Robert C. Lock, President, Small System Services, Inc.

Tape Load Test And Head Alignment

Louis F. Sander
Pittsburgh, PA

This article shows how to prepare and use a special test tape for the cassette recorder of any PET or CBM. When the tape is LOADed, its contents appear on the screen, allowing the user to see any tape errors *as they occur*. The tape error display is a sensitive indicator of the overall quality of the tape reading process, and one which can be used in curing such mysterious and aggravating problems as defective tapes and dirty or magnetized heads. The test tape can also be used as a working standard for head alignment.

Making The Load Test Tape

The first step in creating your tape is to enter and SAVE the "Test Tape Maker" program that appears later in the article. Then RUN it and follow the instructions on the screen, but be sure you understand the material in this section first.

The instructions ask you to use your Machine Language Monitor. Don't worry if you've never used it before — it's easy. If you have an older PET with Original ROMs, LOAD your monitor from tape and RUN it, being careful not to lose the "Test Tape Maker" instructions from the screen. With any other ROMs, you have a built-in monitor. Activate it by entering SYS 1024.

Once the monitor is running, it will prompt you with a dot. Mount a fully rewound tape, and save the 1st pass program by entering the indicated line exactly as it appears in the "Test Tape Maker" instructions. Then rewind the tape again, and prepare to do something unusual — you are going to record a new header on top of the one already on the tape, but you're going to leave the rest of the tape unchanged! You will do it by initiating another machine language SAVE, this time hitting STOP as soon as the header has been recorded on the tape. Knowing when to hit STOP is the tricky part, but the following paragraphs will teach you the trick.

If you can hear your tapes as they save, your task is easy. Some CB2 amplifiers amplify tape sounds, too, and you're in luck if yours works this way. If it doesn't, just connect your amplifier temporarily to pin eight of the user port connector, which is a convenient pickup point for the Tape Write signal. When you initiate your save, you'll hear about ten seconds of leader tone, followed by three seconds of buzz, followed by two more seconds of leader and a lot more buzz. The three seconds of buzz is the tape header, so you'll want to hit STOP the instant you start hearing the *second* section of leader tone.

Even if you have no way of listening to your SAVEs, you can tell when to hit STOP in making this tape. First, SAVE any program into a fully rewound tape. Then fully rewind it again and LOAD it, using a stopwatch to time the interval between pressing PLAY and seeing the FOUND message on the screen. Then, when recording LOAD TEST, wait exactly this length of time between pressing PLAY & RECORD and hitting STOP. On my PET, this is just over 13 seconds, and it should be the same on yours, but you should use a stopwatch to be sure.

Now that you know when to hit STOP, let's go back to "Test Tape Maker." Use the Monitor to save LOAD TEST onto the rewound 1st pass tape, making the exact entries appearing on your screen. Press PLAY, and as soon as the header has been recorded (the right number of seconds, or the appearance of the second leader tone), hit STOP. The STOP key on the computer is preferable to the one on the recorder, but either one will work. The timing of this move is critical to a fraction of a second, so use your fastest finger.

As soon as you hit STOP, your tape is finished. To be sure you have a good one, rewind it and LOAD it. If all is well, you will see the FOUND LOAD TEST and LOADING messages; then your screen will begin to fill with solid green (or white) squares. Once the screen is full, these will be replaced one-by-one with a full screen of colons, then a screen of shaded squares, then one of minus signs. Finally, an OK will print at the bottom of your screen, and after about 30 seconds, a READY message will appear somewhere on screen. No other characters should appear at any time. The newer machines with dynamic RAMs will not show the last two screens, and 80 column machines will combine the first two on one screen. If you cannot get the perfect "LOAD" described above, either you have made a defective tape, or you have a problem with your recorder. Clean and demagnetize your heads[1], and try a few more loads. If you still don't achieve perfection, try making a new LOAD TEST tape — you may have hit STOP too soon or too late, or you may be working with a defective cassette.

When you have a tape that loads perfectly at least once, load it several more times in succession.

You should get perfect or near-perfect results every time. Anything other than smooth screen filling, with no unusual characters, is an indication of an imperfect load. If you fail to achieve perfection, refer to the material in the next section. Otherwise, consider your test tape ready for use. Mark it with the date it was made, and set it aside in a safe place. If you want a second copy, use "Test Tape Maker" to create one, since there is no way to copy a completed tape. It's a good idea to put a copy of "Test Tape Maker" immediately after LOAD TEST on your tape, so you will have both of them whenever you need them.

Using The Test Tape

Now, whenever you have trouble LOADing a tape, you can evaluate the situation by loading LOAD TEST. If the screen fills properly, you know that your PET worked perfectly during the LOAD. The trouble is probably with your tape — it may be defective, or it may have been made on a recorder whose head is not aligned with yours. Read the Head Alignment section below.

If your screen *doesn't* fill properly, there may be a problem with your machine, and you can use the screen display to evaluate it. Every improper or misplaced character on the screen represents a mishandled byte. By using the second program copy recorded on every tape, PET can automatically correct up to 31 of these. LOAD TEST, by the way, lets you see this as it happens, when "proper" characters appear on the screen in place of the "bad" ones during the 30 seconds just before the READY message. Normally, you should have very few, if any, mishandled bytes. The more you have, the greater your problem. If you have more than a very few, even though PET can correct them, something is awry with your machine's LOAD process, and corrective action is called for.

The first corrective action, of course, is to clean and demagnetize your tape heads[1]. The second is to clean the contacts on the connector and the circuit board where your recorder plugs into your computer. If these steps fail to improve your situation, try a head alignment. If that also fails, see your serviceman.

Head Alignment

For a tape to load properly, your PETs read/write head must be precisely aligned with the magnetic field on the tape. The tape's field is, of course, perfectly aligned with the head of the recorder that made it. A small amount of misalignment between tape and read head often shows up as mishandled bytes, a moderate amount as a ?LOAD ERROR, and a large amount as a complete failure to read the tape.

Misalignment can occur with one of your own tapes if your machine's alignment has changed since you made the tape. It also occurs if a tape you are trying to read was recorded on a machine whose head is out of line with yours. Imperfect alignment between two PETs is quite common, and is often the cause of inability to load other people's tapes.

You can use your LOAD TEST tape to bring *any* recorder's head into alignment with the head that made LOAD TEST. Adjustment procedures have been published elsewhere[2]. Once you know how to make the adjustment, just load your test tape into the appropriate machine and adjust its head for perfect screen patterns. There is no need for any PEEKs to confirm the success of the LOAD, since you can see every mishandled byte right on the screen itself. You can even use LOAD TEST to adjust the head *while the tape is loading*, since it gives you 20-40 seconds of real-time feedback on the quality of your LOAD.

Always remember that you are adjusting the read head to the tape that it is reading. If the recorder which made it was misaligned from "standard," your test tape will be misaligned as well. Nevertheless, you should be able to get *any* recorder to read it. Now that you know how to make and use a "Load Test" tape, *you* need read no further. If you're interested in how and why it works, read on.

Theory Of Operation: Screen Images

Let us consider what is recorded on the Load Test tape. By a series of POKEs, "Test Tape Maker" created a machine language "program" of 1024 "square," 1024 colons, 1024 shaded squared, 997 minus signs, a space, an 'O' and a 'K', all in memory locations 2768 to 6839, (0AD0 – 1AB8 hex). When you saved that material as 1st pass, you made a tape whose header instructed PET to load it into those locations[3]. When you rewound the tape and did the second "computus interruptus" SAVE, you recorded a new header over the old one, but left the remaining material intact. The new header asks PET to load that material into memory locations 32768 – 36839, (8000 – 8FE8 hex), which are very interesting locations.

Experienced PET owners know that "screen memory" occupies the 1000 locations between 32768 and 33767. POKEs to those locations, (such as POKE 33000,42), cause characters to appear instantaneously on the screen. "Load Test" uses a less-well-known fact about screen memory: that POKEs to the screen memory locations *plus* 1024, (and on some machines 2048 or 3072), will *also* put characters on the screen. Clear your screen and POKE (33000 + 1024), 42 to see it for yourself. This multiple POKEability exists because of a peculiarity in PET's address decoding scheme;

there really *isn't* any memory up there. These second, third, and fourth addresses for each screen position are sometimes called "images" of screen memory.

A little reflection on the above paragraphs will reveal that locations 32768 through 36839 include the screen memory plus its images, and that LOADing a program there will actually put the program material onto the screen up to four times in succession. There we can see the LOAD, and any errors, with our own two eagle eyes.

References

1. "Getting the Most From Your PET Cassette Deck," **COMPUTE!**, #10, March, 1981, page 42.
2. "Detecting Loading Problems and Correcting Alignment on Your PET," **COMPUTE!** #8, January 1981, page 114.
3. "All About LOADing PET Cassettes," **COMPUTE!** #16, September, 1981, page 129.


```
100 PRINT"{02 DOWN}TEST TAPE MAKER ~
    - WORKING - (25 SECONDS)"
110 FORI=2768TO3791:POKEI,160:NEXT
120 FORI=3792TO4815:POKEI,58:NEXT
130 FORI=4816TO5839:POKEI,102:NEXT
140 FORI=5840TO6836:POKEI,45:NEXT
150 POKEI,32:POKEI+1,15:POKEI+2,11
160 PRINT"{CLEAR}ACTIVATE THE ML MO
    NITOR, THEN MOUNT"
170 PRINT"A FULLY REWOUND TAPE AND ~
    ENTER:"
180 IFPEEK(50003)=0THEN310
190 PRINT". S"CHR$(34)"1ST PASS"CHR
    $(34)",01,0AD0,1AB8"
200 PRINT"THEN REWIND AND ENTER:"
210 PRINT". S"CHR$(34)"LOAD TEST"CH
    R$(34)",01,8000,8FE8"
220 PRINT"{REV}HIT STOP AS SOON AS ~
    THE HEADER HAS BEEN"
230 PRINT"{REV}RECORDED. (SEE ARTIC
    LE FOR DETAILS).{UP}":END
300 REM ** INSTR FOR ORIGINAL ROMS
310 PRINT".   S 01,1ST PASS,0AD0,1A
    B8"
320 PRINT"THEN REWIND AND ENTER:"
330 PRINT".S 01,LOAD TEST,8000,8FE8
    "
340 GOTO220
```

©

MICROMON

An Enhanced Machine Language Monitor

R. Arthur Cochrane
Beech Island, SC

Editor's note: Micromon is for Upgrade and 4.0 BASICs, all memory sizes, all keyboards and is in the public domain. We present it here because many readers live where there are no computer clubs to permit the exchange of public domain programs. If you have enough memory, you can add the additional commands of "Micromon Plus" as well. "Plus" is from $5B00 to $5F48 and you will want to move Micromon from $1000 up to $6000.

There is quite a bit of typing here so we've provided two checksum programs which will find and flag any errors. If you are unfamiliar with machine language programming, see the instructions for typing in "Supermon" in last month's **COMPUTE!**, *page 134. — RTM*

Background

For those who may not know what Micromon is, I will start with a little background. Micromon started as Extramon which is an extended machine language monitor for the TIM monitor in the PET. Extramon was originally written by Bill Seiler. It is for Upgrade BASIC and has the following commands;

A — A simple one line assembler.

B — Set a break point.

C — Compare two ranges of memory and print the addresses of any differences.

D — Disassemble a range of memory.

F — Fill a range of memory with a byte.

H — Hunt a range of memory for a certain HEX or ASCII pattern and print the addresses where they occur.

I — Do a memory dump or a range of memory by printing the HEX and ASCII values.

N — New Locate a machine language program by adding an offset to the three byte instructions.

Q — Start execution of a machine language program and stop execution when the break point is reached.

T — Transfer a range of memory to another part of memory.

W — Single step execution of a machine language program.

Extramon loads into the address range $1000 to $17FF, but the T and N commands can be used to relocate Extramon to another part of memory.

Micromon is an improved version of Extramon and is also by Bill Seiler. Micromon has the same commands as Extramon plus those of the TIM monitor and works on Upgrade BASIC and BASIC 4.0. It works on both BASICs because only 4 ROM routines are used, two of these routines are in the jump table at the top of memory and the other two used by Micromon are found by checking a location to determine the BASIC. The ability to use the up and down cursor control keys to scroll the memory dump and disassembler is added.

Improvements

Now Micromon has been improved by the addition of more instructions to make it a full 4K program. The following instructions have been added:

E — Kill Micromon by restoring the TIM break vector and IRQ vector and return to BASIC.

K — Kill Micromon by restoring the TIM break vector and IRQ vector and do a BRK to the TIM monitor.

O — Calculate a branch instruction offset given a starting and target address.

Z — Change to the opposite character set from the one currently in use.

$ — Print the decimal value, the ASCII values for the two bytes, and the binary value for an input HEX value.

— Print the HEX value, the ASCII values for the two bytes, and the binary value for an input decimal value.

% — Print the HEX value, the decimal value, and the ASCII values for the two bytes for an input binary value.

" — Print the HEX value, the decimal value, and the binary value for an input ASCII value.

+ — Add two HEX numbers.

− — Subtract two HEX numbers.

& — Print the checksum for a range of memory.

An additional module (Micromon Plus) to work with Micromon is also available. This module is about an additional 1K of program and it has the following commands:

I — Set form feeds and a heading for disassemblies and memory dump printouts.

P — Switch output to a printer for hard copy disassemblies and memory dumps.

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.

PIE-C

PET/CBM* IEEE-488 TO PARALLEL PRINTERS By LemData Products

P.I.E.-C MEANS—Professional design, Indispensible features, Excellent quality and Cost effectiveness. You can't buy a better parallel interface for your PET/CBM.

Our P.I.E.-C will interface your PET/CBM through the IEEE-488 bus to . . . . . .

the NEC Spinwriter, the C. Itoh Starwriter, printers by Centronics, Epson, Anadex, Escon Products, the Paper Tigers by IDS, the MIPLOT by Watanabe, the DIP printers, the AJ-841, the OKIDATA printers, plus ALL OTHER parallel ASCII printers.

Assembled with custom case, CBM-TO-ASCII code converter and appropriate cable, the **P.I.E.-C** is only $129.95 (+ $5 S&H). Md. Res. +5% tax. Specify printer and CBM models.

LemData Products, P.O. Box 1080, Columbia, Md. 21044 Phone (301) 730-3257
*PET/CBM are trademarks of Commodore Business Machines

(Dealer Inquiries Invited)

J — Print the address at which a file loads.

Y — Load a file starting at a specific address and not the load address in the file.

› or @ — The DOS commands for reading the disk error channel, sending commands to the disk, or displaying the disk directory.

Micromon is very useful for debugging machine language programs. The disassembler allows the code to be examined and the single step command allows following the execution of code to spot bugs. The Transfer and New Locate commands allow code to be relocated to another part of memory without the need for reassembly. Micromon is a must for any PET machine language programmer.

There are several extended monitors available for the PET. Supermon is one example. Most of the other monitors have some of the same commands as Micromon and maybe a few others. One of the problems with these monitors is that there are different versions for Upgrade BASIC and BASIC 4.0. Micromon will work, as is, on either BASIC. It does not work on Original BASIC though it might be possible to modify it. There is a version of Supermon for each of the three BASICs if a super monitor is needed for Original BASIC.

Because the VIC-20 has Upgrade Basic it will be possible to modify Micromon for VIC use, giving it a powerful machine language monitor. The modification will involve checking the subroutine calls and modifying the scroll for the screen size of the VIC. If anyone is successful in this modification they should be sure to publish the results for others. Because the full Micromon is a 4K program, it would be a good program for programming into a VIC plug-in program cartridge.

Micromon is *free* (so is Supermon), but where do you get it? A PET user group is one source. For those who would like source code, Micromon source code in Carl Moser's MAE assembler format is available. Micromon can be assembled and burned into an EPROM and plugged into an empty socket in the PET so Micromon is available with a SYS and does not have to be loaded each time the PET is reset or powered up.

I hope that you will pass Micromon on to your friends. This program is in the public domain and should be passed around freely. If anyone finds bugs or has comments please contact me about them.

I would like to thank James Strasma for all the information which he provided me for this work on Micromon.

Note To Other 6502 Users

Because Micromon uses only four ROM routines (input a character, output a character, load a program, and save a program) and a few zero page locations (IRQ vector, BRK vector, and screen line pointers) it may be possible for Apple, Atari, or other 6502 users to modify Micromon for their machine. If someone is successful at this be sure to pass the information on to others.

Micromon Instructions

SIMPLE ASSEMBLER

```
.A 2000 A9 12    LDA #$12
.A 2002 9D 00 80 STA $8000,X
.A 2005 DEX:GARBAGE
```

In the above example, the user started assembly at 2000 HEX. The first instruction was load a register with immediate 12 HEX. In the second line the user did not need to type the A and address. The simple assembler retypes the last entered line and prompts with the next address. To exit the assembler, type a return after the address prompt. Syntax is the same as the Disassembler output. A colon (:) can be used to terminate a line.

BREAK SET

```
.B 1000 00FF
```

The example sets a break at 1000 HEX on the FF HEX occurrence of the instruction at 1000. Break set is used with the QUICK TRACE command. A BREAK SET with count blank stops at the first occurrence of the break address.

COMPARE MEMORY

```
.C 1000 2000 C000
```

Compares memory from HEX 1000 to HEX 2000 to memory beginning at HEX C000. Compare will print the locations of the unequal bytes.

DISASSEMBLER

```
.D 2000 3000
., 2000 A9 12    LDA #$12
., 2002 9D 00 80 STA $8000,X
., 2005 AA       TAX
```

Disassembles from 2000 to 3000. The three bytes following the address may be modified. Use the CRSR KEYS to move to and modify the bytes. Hit return and the bytes in memory will be changed. MICROMON will then disassemble that line again.

Disassembly can be done under the control of the cursor. To disassemble one at a time from $1000.

```
.D 1000
```

If the cursor is on the last line, one instruction can be disassembled for each pressing of the cursor down key. If it is held down, the key will repeat and continuous disassembly will occur. Disassembly can even be in reverse! If the screen is full of a disassembly listing, place the cursor at the top line of the screen and press the cursor up key.

EXIT MICROMON

.E

Combine the killing of MICROMON and exit to BASIC.

FILL MEMORY

.F 1000 1100 FF

Fills the memory from 1000 HEX to 1100 HEX with the byte FF HEX.

GO RUN

.G

Go to the address in the PC Register display and begin run code. All the registers will be replaced with the displayed values.

.G 1000

Go to address 1000 HEX and begin running code.

HUNT MEMORY

.H C000 D000 'READ

Hunt thru memory from C000 HEX to D000 HEX for the ASCII string "read" and print the address where it is found. Maximum of 32 characters may be used.

.H C000 D000 20 D2 FF

Hunt memory from C000 HEX to D000 HEX for the sequence of bytes 20 D2 FF and print the address. A maximum of 32 bytes may be used. Hunt can be stopped with the STOP key.

KILL MICROMON

.K

Restore the Break vector and IRQ that was saved before MICROMON was called and break into the TIM monitor. A return to MICROMON can be done with a Go to the value in the PC register.

LOAD

.L "RAM TEST",08

Load the program named RAM TEST from the disk. *Note for cassette users:* To load or save to cassette. Kill MICROMON with the K command to return to the TIM monitor. Then use the TIM monitor L and S commands to load and save to the cassettes. This has to be done because of the repeat keys of MICROMON. BASIC 4.0 users then can return to MICROMON with a Go command to the PC value but BASIC 2.0 users should return to BASIC then SYS to Micromon because the TIM overwrites the IRQ value for loads and saves with a filename.

MEMORY DISPLAY

.M 0000 0008

```
.: 0000 30 31 32 33 34 35 36 37 1234567
.: 0008 38 41 42 43 44 45 46 47 89ABCDE
```

Display memory from 0000 HEX to 0008 in HEX and ASCII. The bytes following the address may be modified by editing and then typing a RETURN.

Memory display can also be done with the cursor control keys.

NEW LOCATER

```
.N 1000 17FF 6000 1000 1FFF
.N 1FB0 1FFF 6000 1000 1FFF W
```

The first line fixes all three byte instructions in the range 1000 HEX to 1FFF HEX by adding 6000 HEX offset to the bytes following the instruction. New Locater will not adjust any instruction outside of the 1000 HEX to 1FFF HEX range. The second line adjusts Word values in the same range as the first line. New Locater stops and disassembles on any bad op code.

CALCULATE BRANCH OFFSET

.O 033A 033A FE

Calculate the offset for branch instructions. The first address is the starting address and the second address is the target address. The offset is then displayed.

QUICK TRACE

```
.Q
.Q 1000
```

The first example begins trace at the address in the PC of the register display. The second begins at 1000 HEX. Each instruction is executed as in the WALK command, but no disassembly is shown. The Break Address is checked for the break on Nth occurrence. The execution may be stopped by pressing the STOP and = (left arrow on business) keys at the same time.

REGISTER DISPLAY

```
.R
   PC IRQ SR AC XR YR SP
.: 0000 E455 01 02 03 04 05
```

Displays the register values saved when MICROMON was entered. The values may be changed with the edit followed by a RETURN.

SAVE

.S "1:PROGRAM NAME",08,0800,0C80

Save to disk drive #1 memory from 0800 HEX up to, *but not including*, 0C80 HEX and name it PROGRAM NAME. See note in LOAD command for cassette users.

TRANSFER MEMORY

.T 1000 1100 5000

Transfer memory in the range 1000 HEX to 1100 HEX and start storing it at address 5000 HEX.

WALK CODE

.W

Single step starting at address in register PC.

.W 1000

Single step starting at address 1000 HEX. Walk will cause a single step to execute and will disassemble the next instruction. Stop key stops walking. The J key finishes a subroutine that is walking then continues with the walk.

EXIT TO BASIC

.X

Return to BASIC READY mode. The stack value saved when entered will be restored. Care should be taken that this value is the same as when the MONITOR was entered. A CLR in BASIC will fix any stack problems. Do not X to BASIC then return to MICROMON via a SYS to the cold start address. Return via a SYS to a BRK (SYS 1024) or SYS to the Warm start of MICROMON (Warm start = Cold start + 3) An X and cold start will write over the TIM break vector that was saved.

CHANGE CHARACTER SETS

.Z

Change from uppercase/graphics to lower/ uppercase mode or vice versa.

HEX CONVERSION

.$4142 16706 A B 0100 0001 0100 0010

A HEX number is input and the decimal value, the ASCII for the two bytes, and the binary values are returned. The ASCII control values are returned in reverse.

HEX conversion can also be scrolled with the cursor control keys.

DECIMAL CONVERSION

.#16706 7142 A B 0100 0001 0100 0010

A decimal number is input and the HEX value, the ASCII for the two bytes, and the binary values are returned.

BINARY CONVERSION

.%0100000101000010 4142 16706 A B

A binary number is input and the HEX value, the decimal number, and the ASCII values are returned.

ASCII CONVERSION

."A 41 65 0100 0001

An ASCII character is input and the HEX value, decimal value, and binary values are returned. Because of the quote, *the control characters can be determined also.*

ADDITION

.+ 1111 2222 3333

The two HEX numbers input are added, and the sum displayed.

SUBTRACTION

.-3333 1111 2222

The second number is subtracted from the first number and the difference displayed.

CHECKSUM

.& A000 AFFF 67E2

The checksum between the two addresses is calculated and displayed.

MICROMON INSTRUCTIONS:

A SIMPLE ASSEMBLE
B BREAK SET
C COMPARE MEMORY
D DISASSEMBLER
E EXIT MICROMON
F FILL MEMORY
G GO RUN
H HUNT MEMORY
K KILL MICROMON
L LOAD
M MEMORY DISPLAY
N NEW LOCATER
O CALCULATE BRANCH
Q QUICK TRACE
R REGISTER DISPLAY
S SAVE
T TRANSFER MEMORY
W WALK CODE
X EXIT TO BASIC
Z CHANGE CHARACTER SETS
$ HEX CONVERSION
# DECIMAL CONVERSION
% BINARY CONVERSION
" ASCII CONVERSION
\+ ADDITION
\- SUBTRACTION
& CHECKSUM

MICROMON also has repeat for all keys.

MICROMON is executed by the following: SYS 4096 as listed in Program 2 where it resides in $1000 to $1FFF.

For 8032, make the following changes for MICROMON operation. In location the X stands for the start of MICROMON. Values in HEX.

Location	Old Value	New Value	
X3E7	08	10	To display 16 instead
X3EC	08	10	of 8 bytes.
X3F6	08	10	
X427	08	10	
XDA3	08	10	
XCFC	28	50	To fix scroll.
XD7B	28	50	
XE16	83	87	
XE20	28	50	
XE24	C0	80	

XE26	04	08	
XE37	27	4F	
XE46	28	50	
X681	24	00	To print all characters in Walk command.

Micromon Plus Instructions

PRINTING DISASSEMBLER

.(Shift) D 1000 1FFF

The same as the Disassembler but no ., printed before each line. Also the ASCII values for the bytes are output at the end of the line.

FORM FEED SET

.I

Sets a form feed for printout. Gives 57 printed lines per page. Works with the Shift D and Shift M commands.

.I "Heading"

Sets form feed with a message to be printed at the top of each page.

.I X

Cancels form feed.

PRINT LOAD ADDRESS

.J "File name"

Read the load address of the file and print it in hex. Device number 8 is used.

KILL MICROMON ADDITIONS

.(Shift) K

Kill MICROMON and its additions and BRK to the TIM monitor. This is the same as the unshifted K command except now a G command will reinitialize MICROMON and the additions.

LOAD FROM DISK

.(Shift) L "filename"

This is the same as the normal load command except that the disk (device #8) is used as the default, not the cassette.

PRINTING MEMORY DUMP

.(Shift) M F000 F100

The same as the normal Memory dump, but does not print the .: and prints out 16 hex bytes and the ASCII for them.

PRINT SWITCHER

.P

If the output is to the CRT then switch the output to the printer (device #4). If the output is not the CRT then clear the output device and restore the output to the CRT.

.P 06

Make device #6 the output device if the current output is the CRT.

SEND TO PROM PROGRAMMER

.U 06 7000 7FFF

This command will send out bytes to a PROM programmer on the IEEE bus. The first byte is the device number and the two addresses are the range of memory to output. A CHR$(2) is sent first to start the programmer. This is followed by the memory bytes as ASCII characters separated by spaces. After all bytes have been sent, a CHR$(3) is sent to stop the programmer. MICROMON then does a checksum on the range to compare against the programmer checksum. Although this is for a particular programmer, it could be modified for others.

SPECIFY LOAD ADDRESS

.Y 7000 "Filename"

This command allows a file to be loaded starting at the address you specify and not the load address it would normally load into. The disk (device #8) is used for loading.

TEXT FLIP FOR 8032 & FAT 40's

.(Shift) Z

This is for 8032 and Fat 40's to go from Text to Graphics mode or vice versa.

DOS SUPPORT

.@ or .>

This reads the error channel from disk device number 8.

.@ disk command or .> disk command

This sends the disk command to disk device number 8.

.@ $0 or .>$0

This reads the directory from disk device number 8. The SPACE BAR will hold the display and any other key will start it again and the STOP key will return to command mode.

CONTROL CHARACTERS

.(Up arrow)G

This command will print the control character of the ASCII character input.

Examples of controls:

g	Ring bell
i	Tab set and clear
M	Insert line
n	Text mode
N	Graphics mode
q	Cursor down
Q	Cursor up
s	Home cursor
S	Clear screen
u	Delete line
v	Erase end
V	Erase begin

MICROMON ADDITIONAL INSTRUCTIONS

(Shift) D PRINTING DISASSEMBLER
I HEADING AND FORM FEED CONTROL
J PRINT LOAD ADDRESS
(Shift) K KILL MICROMON ADDITIONS
(Shift) L LOAD FROM DISK
(Shift) M PRINT MEMORY DISPLAY
P PRINTER SWITCHING
U SEND TO PROM PROGRAMMER
Y SPECIFY LOAD ADDRESS
(Shift) Z TEXT/GRAPHICS FLIP
› DOS SUPPORT COMMANDS
@ DOS SUPPORT COMMANDS
(Up arrow) CONTROL CHARACTERS

Program 1.

```
10 DATA 15463,14894,14290,11897,12
     453,13919,14116,11715,1257
     5,14571
20 DATA 13693,11853,12903,14513,12
     137,15006,12654,13291,1243
     6,13899
30 DATA 15366,9999,11834,13512,128
     92,14475,15149,14896,15782
     ,9511
40 DATA 12171,8985
100 Q=4096
110 FOR BLOCK=1TO32
120 FOR BYTE=0TO127
130 X=PEEK(Q+BYTE):CK=CK+X
140 NEXT BYTE
150 READ SUM
160 IF SUM <> CK THEN PRINT" ERROR
    IN BLOCK #"BLOCK:GOTO170
165 PRINT"                    BLOCK"
    BLOCK" IS CORRECT
170 CK=0:Q=Q+128
180 NEXT BLOCK
```

Program 2.

```
1000 4C 0C 10 4C 6F 10 4C CF
1008 FF 4C D2 FF 78 A5 92 A6
1010 93 8D E5 02 8E E6 02 AD
1018 F6 1F AE F7 1F 8D E3 02
1020 8E E4 02 AD F0 1F AE F1
1028 1F 85 92 86 93 A5 90 A6
1030 91 CD EE 1F D0 05 EC EF
1038 1F F0 10 8D 9E 02 8E 9F
1040 02 AD EE 1F AE EF 1F 85
1048 90 86 91 AD EC 1F AE ED
1050 1F E0 80 B0 08 85 34 86
1058 35 85 30 86 31 A9 10 8D
1060 84 02 8D 85 02 A9 00 8D
1068 86 02 8D A2 02 58 00 38
1070 AD 7B 02 E9 01 8D 7B 02
1078 AD 7A 02 E9 00 8D 7A 02

1080 20 55 19 A2 42 A9 2A 20
1088 29 18 A9 52 D0 23 A9 3F
1090 20 09 10 20 55 19 A9 2E
1098 20 09 10 A9 00 8D 94 02
10A0 8D A2 02 A2 FF 9A 20 A4
10A8 18 C9 2E F0 F9 C9 20 F0
10B0 F5 A2 1D DD 92 1F D0 13
10B8 8D 87 02 8A 0A AA BD B0
10C0 1F 85 FB BD B1 1F 85 FC
10C8 6C FB 00 CA 10 E5 6C E3
10D0 02 A2 02 D0 02 A2 00 B4
10D8 FB D0 09 B4 FC D0 03 EE
10E0 94 02 D6 FC D6 FB 60 A9
10E8 00 8D 8C 02 20 4F 12 A2
10F0 09 20 52 19 CA D0 FA 60
10F8 A2 02 B5 FA 48 BD 91 02

1100 95 FA 68 9D 91 02 CA D0
1108 F1 60 AD 92 02 AC 93 02
1110 4C 17 11 A5 FD A4 FE 38
1118 E5 FB 8D 91 02 98 E5 FC
1120 A8 0D 91 02 60 A9 00 F0
1128 02 A9 01 8D 95 02 20 E6
1130 17 20 55 19 20 13 11 20
1138 3C 18 90 1B 20 0A 11 B0
1140 03 4C C5 11 20 7F 11 E6
1148 FD D0 02 E6 FE 20 3B 19
1150 AC 94 02 D0 45 F0 E5 20
1158 0A 11 18 AD 91 02 65 FD
1160 85 FD 98 65 FE 85 FE 20
1168 F8 10 20 7F 11 20 0A 11
1170 B0 53 20 D1 10 20 D5 10
1178 AC 94 02 D0 1D F0 EB A2

1180 00 A1 FB AC 95 02 F0 02
1188 81 FD C1 FD F0 0B 20 13
1190 18 20 52 19 20 AE 18 F0
1198 01 60 4C 93 10 20 01 18
11A0 20 0B 18 20 A4 18 20 6F
11A8 18 90 17 8D 89 02 AE 94
11B0 02 D0 12 20 13 11 90 0D
11B8 AD 89 02 81 FB 20 3B 19
11C0 D0 EC 4C 8E 10 4C 93 10
11C8 20 01 18 20 0B 18 20 A4
11D0 18 A2 00 20 A4 18 C9 27
11D8 D0 14 20 A4 18 9D A3 02
11E0 E8 20 06 10 C9 0D F0 22
11E8 E0 20 D0 F1 F0 1C 8E 97
11F0 02 20 77 18 90 CC 9D A3
11F8 02 E8 20 06 10 C9 0D F0
```

```
1200 09 20 6F 18 90 BC E0 20
1208 D0 EC 8E 88 02 20 55 19
1210 A2 00 A0 00 B1 FB DD A3
1218 02 D0 0A C8 E8 EC 88 02
1220 D0 F2 20 8E 11 20 3B 19
1228 AC 94 02 D0 05 20 13 11
1230 B0 DE 4C 93 10 20 39 14
1238 20 13 11 90 0D A0 2C 20
1240 E7 10 20 AB 12 20 AE 18
1248 D0 EE 20 B3 15 D0 E3 20
1250 47 19 20 13 18 20 52 19
1258 20 0E 1E 48 20 0B 13 68
1260 20 22 13 A2 06 E0 03 D0
1268 14 AC 8B 02 F0 0F AD 96
1270 02 C9 E8 B1 FB B0 1D 20
1278 A1 12 88 D0 F1 0E 96 02

1280 90 0E BD E9 1E 20 AD 15
1288 BD EF 1E F0 03 20 AD 15
1290 CA D0 D2 60 20 B7 12 AA
1298 E8 D0 01 C8 98 20 A1 12
12A0 8A 8E 88 02 20 1A 18 AE
12A8 88 02 60 AD 8B 02 20 B6
12B0 12 85 FB 84 FC 60 38 A4
12B8 FC AA 10 01 88 65 FB 90
12C0 01 C8 60 A8 4A 90 0B 4A
12C8 B0 17 C9 22 F0 13 29 07
12D0 09 80 4A AA BD 98 1E B0
12D8 04 4A 4A 4A 4A 29 0F D0
12E0 04 A0 80 A9 00 AA BD DC
12E8 1E 8D 96 02 29 03 8D 8B
12F0 02 98 29 8F AA 98 A0 03
12F8 E0 8A F0 0B 4A 90 08 4A

1300 4A 09 20 88 D0 FA C8 88
1308 D0 F2 60 B1 FB 20 A1 12
1310 A2 01 20 F1 10 CC 8B 02
1318 C8 90 F0 A2 03 C0 03 90
1320 F1 60 A8 B9 F6 1E 8D 92
1328 02 B9 36 1F 8D 93 02 A9
1330 00 A0 05 0E 93 02 2E 92
1338 02 2A 88 D0 F6 69 3F 20
1340 09 10 CA D0 EA 4C 52 19
1348 20 01 18 A9 03 20 AC 13
1350 A0 2C 4C 50 15 BD 05 01
1358 CD F8 1F D0 0B BD 06 01
1360 CD F9 1F D0 03 20 D7 18
1368 A5 97 CD 83 02 F0 0A 8D
1370 83 02 A9 10 8D 84 02 D0
1378 24 C9 FF F0 20 AD 84 02

1380 F0-05 CE 84 02 D0 16 CE
1388 85 02 D0 11 A9 02 8D 85
1390 02 A5 9E D0 08 A9 00 85
```

```
1398 97 A9 02 85 A8 AD F3 1F
13A0 48 AD F2 1F 48 08 48 48
13A8 48 6C 9E 02 8D 89 02 48
13B0 20 A4 18 20 19 19 D0 F8
13B8 68 49 FF 4C AE 12 20 39
13C0 14 AE 94 02 D0 0D 20 13
13C8 11 90 08 20 D6 13 20 AE
13D0 18 D0 EE 4C 4A 12 20 55
13D8 19 A2 2E A9 3A 20 29 18
13E0 20 52 19 20 13 18 A9 08
13E8 20 03 19 A9 08 20 B9 13
13F0 A9 12 20 09 10 A0 08 A2
13F8 00 A1 FB 29 7F C9 20 B0

1400 02 A9 2E 20 09 10 C9 22
1408 F0 04 C9 62 D0 0A A9 14
1410 20 09 10 A9 22 20 09 10
1418 20 3B 19 88 D0 DB A9 92
1420 4C 09 10 20 01 18 A9 08
1428 20 AC 13 20 B3 15 20 D6
1430 13 A9 3A 8D 6F 02 4C 5C
1438 15 20 01 18 85 FD 86 FE
1440 20 06 10 C9 0D F0 03 20
1448 06 18 4C 55 19 20 4C 18
1450 85 FD 86 FE A2 00 8E A4
1458 02 20 A4 18 C9 20 F0 F4
1460 9D 8D 02 E8 E0 03 D0 F1
1468 CA 30 14 BD 8D 02 38 E9
1470 3F A0 05 4A 6E A4 02 6E
1478 A3 02 88 D0 F6 F0 E9 A2

1480 02 20 06 10 C9 0D F0 22
1488 C9 3A F0 1E C9 20 F0 F1
1490 20 A4 15 B0 0F 20 84 18
1498 A4 FB 84 FC 85 FB A9 30
14A0 9D A3 02 E8 9D A3 02 E8
14A8 D0 D7 8E 92 02 A2 00 8E
14B0 94 02 A2 00 8E 89 02 AD
14B8 94 02 20 C3 12 AE 96 02
14C0 8E 93 02 AA BD 36 1F 20
14C8 84 15 BD F6 1E 20 84 15
14D0 A2 06 E0 03 D0 14 AC 8B
14D8 02 F0 0F AD 96 02 C9 E8
14E0 A9 30 B0 1E 20 81 15 88
14E8 D0 F1 0E 96 02 90 0E BD
14F0 E9 1E 20 84 15 BD EF 1E
14F8 F0 03 20 84 15 CA D0 D2

1500 F0 06 20 81 15 20 81 15
1508 AD 92 02 CD 89 02 F0 03
1510 4C 91 15 20 3C 18 AC 8B
1518 02 F0 2E AD 93 02 C9 9D
1520 D0 1F 20 13 11 90 0A 98
1528 D0 6F AE 91 02 30 6A 10
```

```
153Ø Ø8 C8 DØ 65 AE 91 Ø2 1Ø
1538 6Ø CA CA 8A AC 8B Ø2 DØ
154Ø Ø3 B9 FC ØØ 91 FB 88 DØ
1548 F8 AD 94 Ø2 91 FB AØ 41
155Ø 8C 6F Ø2 2Ø B3 15 2Ø E7
1558 1Ø 2Ø AB 12 A9 2Ø 8D 7Ø
156Ø Ø2 8D 75 Ø2 A5 FC 2Ø B8
1568 15 8E 71 Ø2 8D 72 Ø2 A5
157Ø FB 2Ø B8 15 8E 73 Ø2 8D
1578 74 Ø2 A9 Ø7 85 9E 4C 93
```

```
158Ø 1Ø 2Ø 84 15 8E 88 Ø2 AE
1588 89 Ø2 DD A3 Ø2 FØ ØD 68
159Ø 68 EE 94 Ø2 FØ Ø3 4C B2
1598 14 4C 8E 1Ø E8 8E 89 Ø2
15AØ AE 88 Ø2 6Ø C9 3Ø 9Ø Ø3
15A8 C9 47 6Ø 38 6Ø CD 8C Ø2
15BØ DØ Ø3 6Ø A9 91 4C Ø9 1Ø
15B8 48 4A 4A 4A 4A 2Ø 32 18
15CØ AA 68 29 ØF 4C 32 18 8D
15C8 7D Ø2 Ø8 68 29 EF 8D 7C
15DØ Ø2 8E 7E Ø2 8C 7F Ø2 68
15D8 18 69 Ø1 8D 7B Ø2 68 69
15EØ ØØ 8D 7A Ø2 A9 8Ø 8D 86
15E8 Ø2 DØ 21 AD 13 E8 1Ø Ø3
15FØ 4C 55 13 D8 68 8D 7F Ø2
15F8 68 8D 7E Ø2 68 8D 7D Ø2
```

```
16ØØ 68 8D 7C Ø2 68 8D 7B Ø2
16Ø8 68 8D 7A Ø2 A5 9Ø 8D 82
161Ø Ø2 A5 91 8D 81 Ø2 BA 8E
1618 8Ø Ø2 2Ø D7 18 AD 12 E8
162Ø 58 AD 7C Ø2 29 1Ø FØ Ø3
1628 4C 6F 1Ø 2C 86 Ø2 5Ø 1F
163Ø AD 7A Ø2 CD 99 Ø2 DØ 6D
1638 AD 7B Ø2 CD 98 Ø2 DØ 65
164Ø AD 9C Ø2 DØ 5D AD 9D Ø2
1648 DØ 55 A9 8Ø 8D 86 Ø2 3Ø
165Ø 14 4E 86 Ø2 9Ø D2 AE 8Ø
1658 Ø2 9A AD F5 1F 48 AD F4
166Ø 1F 48 4C 1F 17 2Ø 55 19
1668 2Ø 3Ø 19 8D 89 Ø2 AØ ØØ
167Ø 2Ø ØB 19 AD 7B Ø2 AE 7A
1678 Ø2 85 FB 86 FC 2Ø 52 19
```

```
168Ø A9 24 8D 8C Ø2 2Ø 52 12
1688 2Ø E4 FF FØ FB C9 Ø3 DØ
169Ø Ø3 4C 93 1Ø C9 4A DØ 56
1698 A9 Ø1 8D 86 Ø2 DØ 4F CE
16AØ 9D Ø2 CE 9C Ø2 AD 12 E8
16A8 C9 EE FØ Ø4 C9 6F DØ 3E
16BØ A2 53 4C 85 1Ø A9 ØØ FØ
16B8 12 AD 9A Ø2 AE 9B Ø2 8D
16CØ 9C Ø2 8E 9D Ø2 A9 4Ø DØ
16C8 Ø2 A9 8Ø 8D 86 Ø2 2Ø Ø6
16DØ 1Ø C9 ØD FØ 11 C9 2Ø DØ
16D8 5C 2Ø 6Ø 18 2Ø FC 18 2Ø
16EØ Ø6 1Ø C9 ØD DØ 4F 2Ø 55
16E8 19 AD 86 Ø2 FØ 22 78 A9
16FØ AØ 8D 4E E8 CE 13 E8 2C
16F8 12 E8 AD FØ 1F AE F1 1F
```

```
17ØØ 8D 82 Ø2 8E 81 Ø2 A9 3B
17Ø8 A2 ØØ 8D 48 E8 8E 49 E8
171Ø AE 8Ø Ø2 9A 78 AD 81 Ø2
1718 85 91 AD 82 Ø2 85 9Ø AD
172Ø 7A Ø2 48 AD 7B Ø2 48 AD
1728 7C Ø2 48 AD 7D Ø2 AE 7E
173Ø Ø2 AC 7F Ø2 4Ø 4C 8E 1Ø
1738 2Ø 4C 18 8D 98 Ø2 8E 99
174Ø Ø2 A9 ØØ 8D 9A Ø2 8D 9B
1748 Ø2 2Ø 5D 18 8D 9A Ø2 8E
175Ø 9B Ø2 4C 93 1Ø 2Ø E6 17
1758 8D AØ Ø2 8E A1 Ø2 2Ø 5D
176Ø 18 8D 8D Ø2 8E 8E Ø2 2Ø
1768 5D 18 8D 8F Ø2 8E 9Ø Ø2
177Ø 2Ø Ø6 1Ø C9 ØD FØ ØA 2Ø
1778 Ø6 1Ø C9 57 DØ Ø3 EE 8C
```

```
178Ø Ø2 2Ø 3C 18 AE 94 Ø2 DØ
1788 18 2Ø ØA 11 9Ø 13 AC 8C
179Ø Ø2 DØ 1A B1 FB 2Ø C3 12
1798 AA BD F6 1E DØ Ø6 2Ø E7
17AØ 1Ø 4C 93 1Ø AC 8B Ø2 CØ
17A8 Ø2 DØ 33 FØ Ø3 8C 8B Ø2
17BØ 88 38 B1 FB AA ED 8D Ø2
17B8 C8 B1 FB ED 8E Ø2 9Ø 1E
17CØ 88 AD 8F Ø2 F1 FB C8 AD
17C8 9Ø Ø2 F1 FB 9Ø 1Ø 88 18
17DØ 8A 6D AØ Ø2 91 FB C8 B1
17D8 FB 6D A1 Ø2 91 FB 2Ø 3B
17EØ 19 88 1Ø FA 3Ø 9E 2Ø 4C
17E8 18 85 FD 86 FE 2Ø 5D 18
17FØ 8D 92 Ø2 8E 93 Ø2 2Ø A4
17F8 18 2Ø 6Ø 18 85 FB 86 FC
```

```
18ØØ 6Ø 2Ø 4C 18 BØ F6 2Ø 6Ø
18Ø8 18 BØ Ø3 2Ø 5D 18 85 FD
181Ø 86 FE 6Ø A5 FC 2Ø 1A 18
1818 A5 FB 48 4A 4A 4A 4A 2Ø
182Ø 32 18 AA 68 29 ØF 2Ø 32
1828 18 48 8A 2Ø Ø9 1Ø 68 4C
183Ø Ø9 1Ø 18 69 F6 9Ø Ø2 69
1838 Ø6 69 3A 6Ø A2 Ø2 B5 FA
184Ø 48 B5 FC 95 FA 68 95 FC
1848 CA DØ F3 6Ø A9 ØØ 8D 97
185Ø Ø2 2Ø A4 18 C9 2Ø FØ F9
1858 2Ø 84 18 BØ Ø8 2Ø A4 18
```

```
1860 20 6F 18 90 07 AA 20 6F
1868 18 90 01 60 4C 8E 10 A9
1870 00 8D 97 02 20 A4 18 C9
1878 20 D0 09 20 A4 18 C9 20

1880 D0 0F 18 60 20 99 18 0A
1888 0A 0A 0A 8D 97 02 20 A4
1890 18 20 99 18 0D 97 02 38
1898 60 C9 3A 08 29 0F 28 90
18A0 02 69 08 60 20 06 10 C9
18A8 0D D0 F8 4C 93 10 A5 9B
18B0 C9 EF D0 07 08 20 CC FF
18B8 85 9E 28 60 20 C6 18 AD
18C0 13 E8 6A 90 F7 60 20 AE
18C8 18 D0 0B 20 D7 18 A9 03
18D0 85 B0 A9 00 85 AF 60 08
18D8 78 AD 40 E8 09 10 8D 40
18E0 E8 A9 7F 8D 4E E8 A9 3C
18E8 8D 11 E8 A9 3D 8D 13 E8
18F0 AD EE 1F 85 90 AD EF 1F
18F8 85 91 28 60 8D 7B 02 8E

1900 7A 02 60 8D 89 02 A0 00
1908 20 52 19 B1 FB 20 1A 18
1910 20 3B 19 CE 89 02 D0 F0
1918 60 20 6F 18 90 0B A2 00
1920 81 FB C1 FB F0 03 4C 8E
1928 10 20 3B 19 CE 89 02 60
1930 A9 7C 85 FB A9 02 85 FC
1938 A9 05 60 E6 FB D0 07 E6
1940 FC D0 03 EE 94 02 60 98
1948 48 20 55 19 68 A2 2E 20
1950 29 18 A9 20 2C A9 0D 4C
1958 09 10 A2 00 BD 76 1F 20
1960 09 10 E8 E0 1C D0 F5 A0
1968 3B 20 47 19 AD 7A 02 20
1970 1A 18 AD 7B 02 20 1A 18
1978 20 52 19 AD 81 02 20 1A

1980 18 AD 82 02 20 1A 18 20
1988 30 19 20 03 19 4C 93 10
1990 4C 8E 10 20 4C 18 20 FC
1998 18 20 5D 18 8D 82 02 8E
19A0 81 02 20 30 19 8D 89 02
19A8 20 A4 18 20 19 19 D0 F8
19B0 F0 DB 20 60 1C AE 80 02
19B8 9A 6C 94 00 4C 8E 10 A0
19C0 01 84 D4 88 84 D1 84 96
19C8 84 9D A9 02 85 DB A9 A3
19D0 85 DA 20 06 10 C9 20 F0
19D8 F9 C9 0D F0 1A C9 22 D0
19E0 DB 20 06 10 C9 22 F0 36
19E8 C9 0D F0 0B 91 DA E6 D1
19F0 C8 C0 10 F0 C7 D0 EA AD
19F8 87 02 C9 4C D0 E1 AD 00

1A00 C0 C9 40 D0 06 20 22 F3
1A08 4C 12 1A C9 4C D0 AD 20
1A10 56 F3 20 BC 18 A5 96 29
1A18 10 D0 E1 4C 93 10 20 06
1A20 10 C9 0D F0 D2 C9 2C D0
1A28 F0 20 6F 18 29 0F F0 C3
1A30 C9 03 F0 FA 85 D4 20 06
1A38 10 C9 0D F0 BA C9 2C D0
1A40 E6 20 F9 17 20 06 10 C9
1A48 2C D0 F4 20 60 18 85 C9
1A50 86 CA 20 06 10 C9 20 F0
1A58 F9 C9 0D D0 EC AD 87 02
1A60 C9 53 D0 F7 AD 00 C0 C9
1A68 40 D0 06 20 A4 F6 4C 93
1A70 10 C9 4C D0 D4 20 E3 F6
1A78 4C 93 10 20 01 18 20 3B

1A80 19 20 3B 19 20 0B 18 20
1A88 52 19 20 13 11 90 0A 98
1A90 D0 15 AD 91 02 30 10 10
1A98 08 C8 D0 0B AD 91 02 10
1AA0 06 20 1A 18 4C 93 10 4C
1AA8 8E 10 20 01 18 20 C0 1A
1AB0 4C 93 10 20 55 19 A2 2E
1AB8 A9 24 20 29 18 20 13 18
1AC0 20 2F 1B 20 E6 1A 20 52
1AC8 19 20 CC 1A 20 CF 1A 20
1AD0 52 19 A2 04 A9 30 18 0E
1AD8 92 02 2E 93 02 69 00 20
1AE0 09 10 CA D0 EF 60 A5 FC
1AE8 A6 FB 8D 93 02 8E 92 02
1AF0 20 52 19 A5 FC 20 FA 1A
1AF8 A5 FB AA 20 52 19 8A 29

1B00 7F C9 20 08 B0 0A A9 12
1B08 20 09 10 8A 18 69 40 AA
1B10 8A 20 09 10 C9 22 F0 04
1B18 C9 62 D0 0A A9 14 20 09
1B20 10 A9 22 20 09 10 28 B0
1B28 05 A9 92 20 09 10 60 20
1B30 52 19 A6 FB A5 FC AC 00
1B38 C0 C0 40 D0 03 4C D9 DC
1B40 C0 4C D0 03 4C 83 CF 4C
1B48 8E 10 20 5B 1B B0 F8 20
1B50 52 19 20 13 18 20 C3 1A
1B58 4C 93 10 A2 04 A9 00 85
1B60 FC 20 17 1C 20 83 1B 85
1B68 FB 20 78 1B 20 92 1B CA
1B70 D0 F7 08 20 52 19 28 60
1B78 20 06 10 C9 0D F0 0F C9

1B80 20 F0 0B C9 30 90 C0 C9
```

```
1B88 3A B0 BC 29 0F 60 68 68
1B90 18 60 85 FE A5 FC 48 A5
1B98 FB 48 06 FB 26 FC 06 FB
1BA0 26 FC 68 65 FB 85 FB 68
1BA8 65 FC 85 FC 06 FB 26 FC
1BB0 A5 FE 65 FB 85 FB A9 00
1BB8 65 FC 85 FC 60 20 17 1C
1BC0 8D 93 02 48 48 20 52 19
1BC8 20 52 19 68 20 1A 18 20
1BD0 52 19 68 AA A9 00 20 36
1BD8 1B 20 52 19 20 CC 1A 4C
1BE0 93 10 20 F4 1B 20 52 19
1BE8 20 13 18 20 2F 1B 20 E6
1BF0 1A 4C 93 10 A2 0F A9 00
1BF8 85 FB 85 FC 20 17 1C 20
```

```
1C00 83 1B 20 11 1C 20 78 1B
1C08 20 11 1C CA D0 F7 4C 52
1C10 19 4A 26 FB 26 FC 60 20
1C18 A4 18 C9 20 F0 F9 60 A9
1C20 02 4D 4C E8 8D 4C E8 4C
1C28 93 10 20 0B 18 4C F6 17
1C30 20 2A 1C 18 A5 FB 65 FD
1C38 85 FB A5 FC 65 FE 85 FC
1C40 4C 50 1C 20 2A 1C 20 13
1C48 11 84 FC AD 91 02 85 FB
1C50 20 52 19 20 13 18 4C 93
1C58 10 20 60 1C 00 6C EC 1F
1C60 78 AD E5 02 AE E6 02 85
1C68 92 86 93 AD 9E 02 AE 9F
1C70 02 85 90 86 91 58 60 20
1C78 2A 1C 20 3C 18 20 52 19
```

```
1C80 A0 00 8C 92 02 8C 93 02
1C88 20 13 11 90 1D AD 94 02
1C90 D0 18 A0 00 18 B1 FB 6D
1C98 92 02 8D 92 02 98 6D 93
1CA0 02 8D 93 02 20 3B 19 4C
1CA8 88 1C AD 93 02 20 1A 18
1CB0 AD 92 02 20 1A 18 4C 93
1CB8 10 AD A2 02 D0 04 A5 9E
1CC0 D0 06 68 A8 68 AA 68 40
1CC8 AD 6F 02 C9 11 D0 7D A5
1CD0 D8 C9 18 D0 ED A5 C4 85
1CD8 FD A5 C5 85 FE A9 19 8D
1CE0 9C 02 A0 01 20 8C 1E C9
1CE8 3A F0 1A C9 2C F0 16 C9
1CF0 24 F0 12 CE 9C 02 F0 CA
1CF8 38 A5 FD E9 28 85 FD B0
```

```
1D00 E1 C6 FE D0 DD 8D 87 02
1D08 20 45 1E B0 B5 AD 87 02
1D10 C9 3A D0 11 18 A5 FB 69
1D18 08 85 FB 90 02 E6 FC 20
```

```
1D20 D6 13 4C 39 1D C9 24 F0
1D28 1A 20 0E 1E 20 AB 12 A9
1D30 00 8D 8C 02 A0 2C 20 4F
1D38 12 A9 00 85 9E 4C 4A 12
1D40 4C C2 1C 20 3B 19 20 B3
1D48 1A 4C 39 1D C9 91 D0 F0
1D50 A5 D8 D0 EC A5 C4 85 FD
1D58 A5 C5 85 FE A9 19 8D 9C
1D60 02 A0 01 20 8C 1E C9 3A
1D68 F0 1A C9 2C F0 16 C9 24
1D70 F0 12 CE 9C 02 F0 15 18
1D78 A5 FD 69 28 85 FD 90 E1
```


1D00 E1 C6 FE D0 DD 8D 87 02
1D08 20 45 1E B0 B5 AD 87 02
1D10 C9 3A D0 11 18 A5 FB 69
1D18 08 85 FB 90 02 E6 FC 20
1D20 D6 13 4C 39 1D C9 24 F0
1D28 1A 20 0E 1E 20 AB 12 A9
1D30 00 8D 8C 02 A0 2C 20 4F
1D38 12 A9 00 85 9E 4C 4A 12
1D40 4C C2 1C 20 3B 19 20 B3
1D48 1A 4C 39 1D C9 91 D0 F0
1D50 A5 D8 D0 EC A5 C4 85 FD
1D58 A5 C5 85 FE A9 19 8D 9C
1D60 02 A0 01 20 8C 1E C9 3A
1D68 F0 1A C9 2C F0 16 C9 24
1D70 F0 12 CE 9C 02 F0 15 18
1D78 A5 FD 69 28 85 FD 90 E1


```
1D80 E6 FE D0 DD 8D 87 02 20
1D88 45 1E 90 03 4C C2 1C AD
1D90 87 02 C9 3A F0 06 C9 24
1D98 F0 1D D0 27 20 15 1E 38
1DA0 A5 FB E9 08 85 FB B0 02
1DA8 C6 FC 20 D9 13 A9 00 85
1DB0 9E 20 40 1E 4C 96 10 20
1DB8 15 1E 20 D5 10 20 B6 1A
1DC0 4C AD 1D 20 15 1E A5 FB
1DC8 A6 FC 85 FD 86 FE A9 10
1DD0 8D 9C 02 38 A5 FD ED 9C
1DD8 02 85 FB A5 FE E9 00 85
1DE0 FC 20 0E 1E 20 AB 12 20
1DE8 13 11 F0 07 B0 F3 CE 9C
1DF0 02 D0 E0 EE 8B 02 AD 8B
1DF8 02 20 B9 13 A2 00 A1 FB
```

```
1E00 8E 8C 02 A9 2C 20 4D 19
1E08 20 52 12 4C AD 1D A2 00
1E10 A1 FB 4C C3 12 A9 83 85
1E18 C8 85 FE A9 00 85 C7 A9
1E20 28 85 FD A0 C0 A2 04 88
1E28 B1 C7 91 FD 98 D0 F8 C6
1E30 C8 C6 FE CA D0 F1 A2 27
```

```
1E38 A9 20 9D 00 80 CA 10 FA
1E40 A9 13 4C 09 10 C0 28 D0
1E48 02 38 60 20 8C 1E C9 20
1E50 F0 F3 88 20 75 1E AA 20
1E58 75 1E 85 FB 86 FC A9 FF
1E60 8D A2 02 85 A7 A5 AA F0
1E68 0A A5 A9 A4 C6 91 C4 A9
1E70 00 85 AA 18 60 20 8C 1E
1E78 20 99 18 0A 0A 0A 0A 8D

1E80 97 02 20 8C 1E 20 99 18
1E88 0D 97 02 60 B1 FD C8 29
1E90 7F C9 20 B0 02 09 40 60
1E98 40 02 45 03 D0 08 40 09
1EA0 30 22 45 33 D0 08 40 09
1EA8 40 02 45 33 D0 08 40 09
1EB0 40 02 45 B3 D0 08 40 09
1EB8 00 22 44 33 D0 8C 44 00
1EC0 11 22 44 33 D0 8C 44 9A
1EC8 10 22 44 33 D0 08 40 09
1ED0 10 22 44 33 D0 08 40 09
1ED8 62 13 78 A9 00 21 81 82
1EE0 00 00 59 4D 91 92 86 4A
1EE8 85 9D 2C 29 2C 23 28 24
1EF0 59 00 58 24 24 00 1C 8A
1EF8 1C 23 5D 8B 1B A1 9D 8A

1F00 1D 23 9D 8B 1D A1 00 29
1F08 19 AE 69 A8 19 23 24 53
1F10 1B 23 24 53 19 A1 00 1A
1F18 5B 5B A5 69 24 24 AE AE
1F20 A8 AD 29 00 7C 00 15 9C
1F28 6D 9C A5 69 29 53 84 13
1F30 34 11 A5 69 23 A0 D8 62
1F38 5A 48 26 62 94 88 54 44
1F40 C8 54 68 44 E8 94 00 B4
1F48 08 84 74 B4 28 6E 74 F4
1F50 CC 4A 72 F2 A4 8A 00 AA
1F58 A2 A2 74 74 74 72 44 68
1F60 B2 32 B2 00 22 00 1A 1A
1F68 26 26 72 72 88 C8 C4 CA
1F70 26 48 44 44 A2 C8 0D 20
1F78 20 20 20 50 43 20 20 49

1F80 52 51 20 20 53 52 20 41
1F88 43 20 58 52 20 59 52 20
1F90 53 50 41 42 43 44 46 47
1F98 48 4C 4D 4E 51 52 53 54
1FA0 57 58 2C 3A 3B 24 23 22
1FA8 2B 2D 4F 5A 4B 25 26 45
1FB0 4D 14 38 17 25 11 35 12
1FB8 9D 11 B5 16 C8 11 BF 19
1FC0 BE 13 55 17 B9 16 5A 19
1FC8 BF 19 29 11 C9 16 B5 19
1FD0 48 13 23 14 93 19 AA 1A
1FD8 4A 1B BD 1B 30 1C 43 1C
1FE0 7B 1A 1F 1C 59 1C E2 1B
1FE8 77 1C B2 19 00 10 55 13
1FF0 EB 15 B9 1C C6 15 8E 10
1FF8 BC 18 30 35 32 37 38 31
```

Program 3.

```
10 DATA 15965,14778,13059,14282,14
    416,17693,12979,12903,1767
    6,21760
20 DATA 14416,17693,12979,12903
100 Q=23296
110 FOR BLOCK=1TO8
120 FOR BYTE=0TO127
130 X=PEEK(Q+BYTE):CK=CK+X
140 NEXT BYTE
150 READ SUM
160 IF SUM <> CK THEN PRINT" ERROR
    IN BLOCK #"BLOCK:GOTO170
165 PRINT"                    BLOCK"
    BLOCK" IS CORRECT
170 CK=0:Q=Q+128
180 NEXT BLOCK
190 PRINT"ANY REMAINING PROBLEMS AR
    E EITHER WITHIN THE FINAL"

200 PRINT"SHORT BLOCK OR WITHIN DAT
    A STATEMENTS IN THIS PROGR
    AM."
```

Program 4.

```
5B00 78 A5 90 A6 91 CD EE 6F
5B08 D0 05 EC EF 6F F0 30 8D
5B10 9E 02 8E 9F 02 AD EE 6F
5B18 AE EF 6F 85 90 86 91 A5
5B20 92 A6 93 8D E5 02 8E E6
5B28 02 AD 3C 5F AE 3D 5F 8D
5B30 E3 02 8E E4 02 AD F0 6F
5B38 AE F1 6F 85 92 86 93 AD
5B40 3E 5F AE 3F 5F E0 80 B0
5B48 08 85 34 86 35 85 30 86
5B50 31 A9 10 8D 84 02 8D 85
5B58 02 A9 00 8D 86 02 8D A2
5B60 02 8D E7 02 8D E8 02 58
5B68 00 A2 0C DD 15 5F D0 13
5B70 8D 87 02 8A 0A AA BD 22
5B78 5F 85 FB BD 23 5F 85 FC

5B80 6C FB 00 CA 10 E5 4C 8E
5B88 60 20 39 64 20 13 61 90
5B90 17 20 EF 60 8E 8C 02 20
```

```
5B98 52 62 20 AB 5B 20 AB 62
5BA0 20 93 5C 20 AE 68 D0 E4
5BA8 4C 9B 60 A2 1E 20 F1 60
5BB0 A0 00 B1 FB 20 60 5C CC
5BB8 8B 02 C8 90 F5 60 A5 B0
5BC0 C9 03 D0 19 20 06 60 AA
5BC8 A9 04 E0 0D F0 09 20 6F
5BD0 68 29 1F C9 04 90 AF 20
5BD8 E3 5B 4C 9B 60 20 CC FF
5BE0 4C 93 60 85 B0 85 D4 20
5BE8 09 5C AE 00 C0 E0 40 D0
5BF0 0B 20 BA F0 20 2D F1 A5
5BF8 96 D0 E2 60 E0 4C D0 5D

5C00 20 D5 F0 20 48 F1 4C F7
5C08 5B A9 00 85 96 8D FC 03
5C10 85 0D 8D E8 02 60 20 39
5C18 64 AE 94 02 D0 10 20 13
5C20 61 90 0B 20 31 5C 20 93
5C28 5C 20 AE 68 D0 EB 4C A8
5C30 5B A2 05 20 F1 60 20 13
5C38 68 A2 02 20 F1 60 A9 10
5C40 20 03 69 A9 10 20 B9 63
5C48 A2 04 20 F1 60 A0 10 A2
5C50 00 A1 FB 20 60 5C 20 3B
5C58 69 88 D0 F5 60 4C 8E 60
5C60 29 7F C9 20 B0 02 A9 20
5C68 4C 09 60 20 06 60 C9 0D
5C70 F0 19 C9 20 D0 03 20 17
5C78 6C C9 58 F0 50 20 71 5D

5C80 8E E8 02 A2 02 20 A7 5C
5C88 4C 9B 60 A2 04 20 C1 5C
5C90 4C 9B 60 20 55 69 AE E7
5C98 02 F0 31 CE E7 02 D0 2C
5CA0 AE E8 02 F0 1A A2 06 20
5CA8 C1 5C A2 14 20 F1 60 BD
5CB0 A3 02 20 09 60 E8 EC E8
5CB8 02 D0 F4 A2 03 D0 02 A2
5CC0 09 20 55 69 CA D0 FA A9
5CC8 39 8D E7 02 60 A9 00 8D
5CD0 E7 02 8D E8 02 4C 9B 60
5CD8 20 09 5C 20 CC FF 20 06
5CE0 60 C9 0D F0 16 C9 24 F0
5CE8 24 48 20 9E 5D 68 20 09
5CF0 60 20 06 60 C9 0D D0 F6
5CF8 4C DD 5B 20 52 69 20 C5

5D00 5D 20 06 60 C9 0D F0 F0
5D08 20 09 60 D0 F4 A2 00 20
5D10 82 5D 20 8B 5D 20 55 69
5D18 20 55 69 A0 03 D0 02 A0
5D20 02 84 D1 A9 08 85 AF 20
5D28 06 60 AA A4 96 D0 36 20
5D30 06 60 A4 96 D0 2F C6 D1
5D38 D0 ED 20 36 6B 20 52 69
5D40 20 06 60 F0 05 20 09 60
5D48 D0 F6 20 55 69 A9 00 85
5D50 AF 20 E4 FF F0 C9 D0 05
5D58 20 E4 FF F0 FB C9 20 F0
5D60 F7 C9 03 D0 BA 20 12 5E
5D68 20 55 69 4C 93 60 20 17
5D70 6C C9 22 D0 7B A2 00 20
5D78 06 60 C9 0D F0 0C C9 22

5D80 F0 08 9D A3 02 E8 E0 40
5D88 90 ED 60 86 D1 A9 A3 85
5D90 DA A9 02 85 DB 20 CC FF
5D98 20 F3 5D 4C C9 5D A9 08
5DA0 85 D4 85 B0 AC 00 C0 C0
5DA8 40 D0 0B 20 BA F0 A9 6F
5DB0 20 28 F1 4C F7 5B C0 4C
5DB8 D0 36 20 D5 F0 A9 6F 20
5DC0 43 F1 4C F7 5B A9 6F 85
5DC8 D3 A9 08 85 D4 85 AF AC
5DD0 00 C0 C0 40 D0 0B 20 B6
5DD8 F0 A5 D3 20 64 F1 4C F7
5DE0 5B C0 4C D0 0B 20 D2 F0
5DE8 A5 D3 20 93 F1 4C F7 5B
5DF0 4C 8E 60 A9 08 85 D4 A9
5DF8 60 85 D3 AD 00 C0 C9 40

5E00 D0 06 20 66 F4 4C F7 5B
5E08 C9 4C D0 E4 20 A5 F4 4C
5E10 F7 5B A9 00 85 AF AD 00
5E18 C0 C9 40 D0 03 4C 8F F3
5E20 C9 4C D0 CC 4C CE F3 A9
5E28 02 2C 4C E8 08 A9 0E 28
5E30 F0 02 09 80 20 09 60 4C
5E38 93 60 20 09 5C 20 6E 5D
5E40 20 8B 5D 20 06 60 8D FB
5E48 00 20 06 60 8D FC 00 20
5E50 12 5E 20 52 69 A9 24 A2
5E58 20 20 29 68 20 13 68 4C
5E60 93 60 20 60 6C 00 6C 3E
5E68 5F A0 08 84 D4 A0 4C 8C
5E70 87 02 A0 00 4C C4 69 20
5E78 17 6C 29 9F 4C 34 5E 4C

5E80 8E 60 20 A4 68 20 6F 68
5E88 29 1F C9 04 90 F1 85 D4
5E90 20 2A 6C A5 FD A6 FE 8D
5E98 92 02 8E 93 02 20 3C 68
5EA0 A5 D4 20 E3 5B A9 02 20
5EA8 09 60 20 52 69 20 13 61
5EB0 90 0F AE 94 02 D0 0A A1
5EB8 FB 20 1A 68 20 3B 69 D0
5EC0 E9 A9 03 20 09 60 20 EF
5EC8 60 20 CC FF 20 F8 60 4C
```

```
5ED0 7D 6C 20 09 5C 20 01 68
5ED8 20 6E 5D 86 D1 20 04 5F
5EE0 20 8D 5D 20 06 60 20 06
5EE8 60 A9 00 85 AF AD 00 C0
5EF0 C9 40 D0 06 20 52 F3 4C
5EF8 01 5F C9 4C D0 81 20 8C

5F00 F3 4C 12 6A AD 00 C0 C9
5F08 40 D0 03 4C 0A F4 C9 4C
5F10 D0 EA 4C 49 F4 50 C4 49
5F18 CD 40 3E DA 4A CB CC 5E
5F20 55 59 BE 5B 89 5B 6B 5C
5F28 16 5C D8 5C D8 5C 27 5E
5F30 3A 5E 62 5E 69 5E 77 5E
5F38 82 5E D2 5E 69 5B 00 5B
5F40 31 30 32 31 38 31 AA AA
```

©

Self-Modifying Programs In BASIC

David Williams
Toronto, Canada

The notion of a program which alters itself as it runs raises feelings of doubt and mistrust in many novice computer users. It seems that such a program would be doomed to failure through some kind of logical paradox. In fact this is not the case. Providing that the part of the program which guides the modification process is separate from that which is being changed, and that no attempt is made to execute program lines which are in the process of being modified, no problems need arise.

As a demonstration, try keying in the following program. As you do so, be careful *not* to include any spaces in lines 10 or 20, or between the quote marks in line 120. Line 20 should consist of a string of exactly twenty π's.

```
10 GOTO100
20 ^^^^^^^^^^^^^^^^^^^^
30 RETURN
100 FORI=826TO838:POKEI,32:NEXT
110 INPUTS$
120 S$="GOTO200:"+S$+CHR$(13)
130 FORI=1TOLEN(S$):POKE838+I,ASC(MID$(S
    ¬$,I)):NEXT
140 POKE175,2:POKE212,2:POKE59408,
    ¬PEEK(59408)ANDNOT32:POKE188,0:
    ¬POKE176,2
150 END
200 POKE175,0:POKE176,3
210 I=0
220 PK=PEEK(517+I)
230 IFPK=0THEN300
240 POKE1038+I,PK
250 I=I+1
260 GOTO220
300 FORI=ITO19:POKE1038+I,32:NEXT
400 GOSUB20
READY.
```

When you have finished entering the program, SAVE it before you first run it. If you have made any typing mistakes it is possible that the program may destroy itself or crash the PET when it is run. Having a copy on tape could save you a lot of re-typing!

When the program is run, a question mark and flashing cursor should appear on the screen. This is the input line 110. Respond to this by typing in some simple instruction in BASIC, such as PRINT 2+3*5, and hit the return key. Within the next couple of seconds the number 17 (the correct response to our input instruction) should be printed, followed by the word READY and the flashing cursor.

The output from this program is less interesting than another result, which can be seen by LISTing the program after it has run. Line 20 will be found to have changed from a meaningless string of π's to:

20 PRINT 2+3*5

the very same instruction that was entered while the program was running. In fact the π's were there only to reserve a set of twenty addresses into which the new line was POKEd. There are still twenty characters in line 20, but most of them are now blanks, which are not visible in the listing and do not cause any problems when the line is executed. Since the number of characters in the line is unchanged, the program can be run repeatedly, altering the contents of this line each time.

Maybe you now think that the program is far more complicated than it needs to be to achieve the result of poking the desired instruction into line 20. Surely all that needs to be done is to poke the ASCII numbers corresponding to P,R,I,N,T,etc. into the 20 addresses of the line. Write your own program to do this, if you want, but you're in for a disappointment. When your program is working properly, the new line will LIST exactly as it should, but when you try to execute it you will get a SYNTAX ERROR. The problem is that BASIC instruction words are stored in PET's memory as single token characters (the LISTing routine translates them back into English words) and the machine cannot understand them except in token form.

The demonstration program not only enters the new line in correct token form, it also does so without invoking the line editor, which would cause the erasure of any pre-existing variables, strings, etc. in memory. To provide this, enter "X=5" in direct mode, then start the program without erasing memory by entering "GOTO 10". Put in any simple BASIC instruction, such as PRINT "DONE", when line 110 asks for it. When the program has finished, enter PRINT X in direct mode. The value 5 will be returned, showing that it is still in memory.

Let's now look at the program to see how it works. The first few lines are arranged so that the changeable line is as near the start of the program as possible. This makes its addresses easy to find (e.g. by using the machine-language monitor), and also protects them from being messed around with by any editing of the rest of the program. Lines 100 to 130 take the input instruction, in string form, prefix it with "GOTO 200", and then POKE it, letter by letter, into the second cassette buffer in the PET starting several characters from the start of the buffer. This buffer is used by the program for one of its originally intended purposes, as an input/output device. Line 140 contains a set of POKEs which "persuade" the PET that a second cassette unit is present, that its "Play" key is pressed, and that this is the device from which it should take its next input and to which it should make its next output, starting at the beginning of the buffer.

At line 150, an END instruction is encountered. This makes the PET print READY into the start of the second cassette buffer and then to take the instructions which are waiting for it in the later locations in the buffer. These are first translated into token form (just what we wanted!) and entered into another buffer, from which they are later read by the routines which execute BASIC instructions. However, the first instruction to be executed is GOTO 200, which re-starts the program and leaves the instructions which we want to put into line 20, in token form, in the basic input buffer.

Line 200 restores the keyboard as the PET's input device and the screen as its output device. Lines 210 to 260 copy the desired text from the basic input buffer into the addresses occupied by line 20, then line 300 fills the remainder of these addresses with blanks. Finally, line 400 demonstrates that the new line actually works, and the machine prints the word READY on the screen as the program ends.

There is an obvious criticism which can be made of this program as it stands. Why go to the trouble of copying the instructions into line 20 when they could have been executed directly from the basic input buffer? This is a valid criticism, provided the instructions are to be executed only once, and that they can legally be performed in direct mode. In practical applications of this technique, however, one or the other of these conditions is often not true.

So much for the mechanics of simple self-modifying programs. Their potential usefulness is great. They represent a class of interactive programs which allow the user not only to supply the values of variables and to make simple choices, but also to give precise logical instructions to the program as it operates.

Probably the simplest applications are in general mathematical programs. These can easily be written to draw the graph of any function, to use an iterative method to solve any equation, or any similar task. The program asks the user to enter the equation he is interested in, and then writes this into one of its own lines. This line can later be executed as many times as necessary for the program to complete its job.

I have recently written a self-modifying program with a very different purpose: to teach students how to set up computer programs in the form of flow-charts. The program allows a student to draw a flow-chart on the PET screen, with BASIC instructions placed on the diagram in the appropriate places. When the diagram is complete, its instructions can be executed without the student having to write a conventional program. The PET simply follows the logic lines on the diagram. When an instruction is encountered, it is written into one of several modifiable lines in the main program and executed appropriately.

I am sure there are thousands of other applications, but I'll leave them for you to discover... ©

VIC-20 Update

TINYMON1: A Simple Monitor For The VIC

Jim Butterfield
Toronto, Canada

One of the things you may miss on the VIC is a Machine Language Monitor: it's not there.

Commodore will be releasing a very powerful MLM on a plug-in cartridge, and serious programmers will certainly want to use it. But for occasional use, a tape-loadable MLM might be very handy.

Here's an early version that may be of use. It should fit on any VIC, with or without extra memory added; and it honors all the commands from the built-in Monitors we know from PET/CBM usage. One minor syntax change: the two addresses of the Memory display command (.M) should be separated by a space rather than a comma.

It's not really practical to type TINYMON directly into a VIC. DATA statements in decimal would take up more room than is available in small VICs; and hex entry would need an MLM to be in place already. So I've prepared the program so that it can be entered on a PET and saved on tape. After it's been created once, the VIC can make its own copies. You'll need a PET with Upgrade ROM or 4.0 ROM to do the job, since the Original ROM PETs don't have a Machine Language Monitor and things would get too complicated.

TINYMON loads like a BASIC Program, and copies can be made with a simple LOAD and SAVE sequence as you would do with BASIC. When you load TINYMON and say RUN, however, some interesting things happen ... the monitor system is repacked into the top of memory, and it will stay there until you turn the power off. You can say .X to return to BASIC and load and run BASIC programs, providing they are not too big. TINYMON grabs about 760 bytes of memory, so you lose a little space.

Find A Zero

Once you're back in BASIC, the question arises: how can you invoke TINYMON when desired? Not an easy trick, since memory is more mobile in the VIC than in the PET/CBM. The thing to do is to find a zero value in memory and SYS to that location. If you have a basic (5K) VIC, SYS 4096 is safe. The sure way is to PEEK first and ensure that there's a zero there (location 10 is often zero).

TINYMON1 must be considered preliminary. It was designed with two major considerations: to use minimum space, and to automatically load into any VIC regardless of the memory fitted. The space consideration is fairly obvious: with 3500-odd bytes available on a small VIC, you want to use up as little as posible. The automatic load feature was tricky to implement; VIC may relocate programs as it loads. What's more, the screen area tends to move around as you add memory.

I scratched my head over the .S (Save) command. If VIC automatically relocates programs during loading, will a SAVEd Machine Language program be safe? As it turns out, VIC has a new tape format available – when a tape is written, it may be defined as "absolute" and will not relocate when it loads. This seems the best compromise, but it has one drawback – the PET/CBM won't load this type of tape. Perhaps that's a design decision that will need to be revised...

Finding Space In Zero Page

VIC is desperately short of zero page space: machine language programmers will have to cope with the shortage as best they can. I have used the same locations that the big Commodore MLM is expected to use. There's a difference, however, the Commodore job will swap out selected parts of zero page and put them back later; I didn't want to give up the space for that kind of luxury. As a result, you may be annoyed by some locations that are disturbed by TINYMON1.

For those unfamiliar with the PET/CBM Machine Language Monitor, the commands are:

.R – display 6502 registers;
Users can use screen editing to type over a display and change the registers;

.M FFFF TTTT – display memory (from .. to);

Users can use screen editing to type over a display and change memory;

.X – exit to BASIC;
It may be wise to type CLR in BASIC after exiting;

.G AAAA – GOTO (execute) address;

.S "PPPP",01,FFFF,TTTT – Save (program-name, device, from, to);

.L "PPPP" – Load (program-name)

There's a delicate tradeoff between features and memory space. There will undoubtedly be other small monitors with a different balance. In any case, I wrote one because I had nothing ... and others in the same position will undoubtedly greet TINYMON1 with glad cries.

Program 1: TINYMON1

Enter on a PET/CBM, using the Machine Language Monitor. Do not try to RUN, but follow your entry with the checksum program, Program 2.

First, make the following change:

```
.:  0028 01 04 18 08 18 08 18 08
```

Now, enter TINYMON1:

```
.:  0400 00 18 04 64 00 99 22 93
.:  0408 11 11 12 1d 1d 1d 20 54
.:  0410 49 4e 59 4d 4f 4e 20 00
.:  0418 31 04 6e 00 99 22 11 20
.:  0420 4a 49 4d 20 42 55 54 54
.:  0428 45 52 46 49 45 4c 44 22
.:  0430 00 4c 04 78 00 9e 28 c2
.:  0438 28 34 33 29 aa 32 35 36
.:  0440 ac c2 28 34 34 29 aa 30
.:  0448 37 38 29 00 00 00 ea ea
.:  0450 a5 2d 85 22 a5 2e 85 23
.:  0458 a5 37 85 24 a5 38 85 25
.:  0460 a0 00 a5 22 d0 02 c6 23
.:  0468 c6 22 b1 22 d0 3c a5 22
.:  0470 d0 02 c6 23 c6 22 b1 22
.:  0478 f0 21 85 26 a5 22 d0 02
.:  0480 c6 23 c6 22 b1 22 18 65
.:  0488 24 aa a5 26 65 25 48 a5
.:  0490 37 d0 02 c6 38 c6 37 68
.:  0498 91 37 8a 48 a5 37 d0 02
.:  04a0 c6 38 c6 37 68 91 37 18
.:  04a8 90 b6 c9 bf d0 ed a5 37
.:  04b0 85 33 a5 38 85 34 6c 37
.:  04b8 00 00 00 00 00 00 00 00
.:  04c0 bf 78 ad fe ff 00 ae ff
.:  04c8 ff 00 8d 16 03 8e 17 03
.:  04d0 a9 80 20 90 ff 58 00 00
.:  04d8 68 85 05 68 85 04 68 85
.:  04e0 03 68 85 02 68 85 01 68
.:  04e8 85 00 00 ba 86 06 38 a5
.:  04f0 01 e9 02 85 01 a5 00 00
.:  04f8 e9 00 00 85 00 00 20 b2
.:  0500 fe 00 a2 42 a9 2a 20 db
.:  0508 fd 00 a9 52 d0 1c a9 3f
.:  0510 20 d2 ff 20 b2 fe 00 a9
.:  0518 2e 20 d2 ff a9 00 00 85
.:  0520 27 20 40 fe 00 c9 2e f0
.:  0528 f9 c9 20 f0 f5 a2 07 dd
.:  0530 e6 ff 00 d0 12 85 1c 8a
.:  0538 0a aa bd ee ff 00 85 c1
.:  0540 bd ef ff 00 85 c2 6c c1
.:  0548 00 00 ca 10 e6 4c 4b fd
.:  0550 00 20 bd fd 00 90 f8 20
.:  0558 ee fd 00 20 bd fd 00 90
.:  0560 f0 20 ee fd 00 20 4c fe
.:  0568 00 f0 1f 20 b2 fe 00 a2
.:  0570 2e a9 3a 20 db fd 00 20
.:  0578 c5 fd 00 a9 05 20 6f fe
.:  0580 00 a5 c3 c5 c1 a5 c4 e5
.:  0588 c2 b0 df 4c 50 fd 00 4c
.:  0590 50 fd 00 20 fe fd 00 85
.:  0598 c1 86 c2 60 a5 c2 20 cc
.:  05a0 fd 00 a5 c1 48 4a 4a 4a
.:  05a8 4a 20 e4 fd 00 aa 68 29
.:  05b0 0f 20 e4 fd 00 48 8a 20
.:  05b8 d2 ff 68 4c d2 ff 18 69
.:  05c0 f6 90 02 69 06 69 3a 60
.:  05c8 a2 02 b5 c0 48 b5 c2 95
.:  05d0 c0 68 95 c2 ca d0 f3 60
.:  05d8 20 0d fe 00 90 07 aa 20
.:  05e0 0d fe 00 90 01 60 4c 4b
.:  05e8 fd 00 a9 00 00 85 2a 20
.:  05f0 40 fe 00 c9 20 f0 f9 20
.:  05f8 20 fe 00 90 17 20 40 fe
.:  0600 00 c9 30 90 10 20 35 fe
.:  0608 00 06 2a 06 2a 06 2a 06
.:  0610 2a 05 2a 85 2a 38 63 c9
.:  0618 3a 08 29 0f 28 90 02 69
.:  0620 08 60 20 cf ff c9 0d d0
.:  0628 f8 68 68 4c 50 fd 00 a5
.:  0630 91 c9 fe d0 05 08 20 cc
.:  0638 ff 28 60 20 61 fe 00 2c
.:  0640 2d 91 30 f8 60 20 4c fe
.:  0648 00 d0 08 a9 03 85 9a a9
.:  0650 00 00 85 99 60 85 1e a0
.:  0658 00 00 20 af fe 00 b1 c1
.:  0660 20 cc fd 00 20 a4 fe 00
.:  0668 c6 1e d0 f1 60 20 0d fe
.:  0670 00 90 0b a2 00 00 81 c1
.:  0678 c1 c1 f0 03 4c 4b fd 00
.:  0680 20 a4 fe 00 c6 1e 60 a9
.:  0688 02 85 c1 a9 00 00 95 c2
.:  0690 a9 05 60 e6 c1 d0 06 e6
.:  0698 c2 d0 02 e6 27 60 a9 20
```

```
.:  06a0 2c a9 0d 4c d2 ff a2 00
.:  06a8 00 bd d0 ff 00 20 d2 ff
.:  06b0 e8 e0 16 d0 f5 20 b2 fe
.:  06b8 00 a2 2e a0 3b 20 db fd
.:  06c0 00 a5 00 00 20 cc fd 00
.:  06c8 a5 01 20 cc fd 00 20 99
.:  06d0 fe 00 20 6f fe 00 4c 50
.:  06d8 fd 00 20 fe fd 00 85 01
.:  06e0 86 00 00 20 99 fe 00 85
.:  06e8 1e 20 83 fe 00 d0 fb f0
.:  06f0 ea 20 bd fd 00 a9 05 85
.:  06f8 1e 20 83 fe 00 d0 fb f0
.:  0700 dc 20 cf ff c9 0d f0 07
.:  0708 20 bd fd 00 85 01 86 00
.:  0710 00 a6 06 9a a5 00 00 48
.:  0718 a5 01 48 a5 02 48 a5 03
.:  0720 a6 04 a4 05 40 78 a6 06
.:  0728 9a 6c 02 c0 4c 4b fd 00
.:  0730 a0 01 84 ba 84 b9 88 84
.:  0738 b7 84 90 84 93 a9 02 85
.:  0740 bc a9 40 85 bb 20 cf ff
.:  0748 c9 20 f0 f9 c9 0d f0 1a
.:  0750 c9 22 d0 d9 20 cf ff c9
.:  0758 22 f0 26 c9 0d f0 0b 91
.:  0760 bb e6 b7 c8 c0 10 f0 c5
.:  0768 d0 ea a5 1c c9 4c d0 e2
.:  0770 a9 00 00 20 d5 ff 20 58
.:  0778 fe 00 a5 90 29 10 d0 f0
.:  0780 4c 50 fd 00 20 cf ff c9
.:  0788 0d f0 e2 c9 2c d0 f0 20
.:  0790 0d fe 00 29 0f f0 d3 c9
.:  0798 03 f0 fa 85 ba 20 cf ff
.:  07a0 c9 0d f0 ca c9 2c d0 e6
.:  07a8 20 bd fd 00 20 cf ff c9
.:  07b0 2c d0 f4 20 fe fd 00 85
.:  07b8 ae 86 af 20 cf ff c9 20
.:  07c0 f0 f9 c9 0d d0 ec a5 1c
.:  07c8 c9 53 d0 f8 20 b2 fe 00
.:  07d0 a9 01 85 b9 20 82 f6 4c
.:  07d8 50 fd 00 0d 20 20 20 50
.:  07e0 43 20 20 53 52 20 41 43
.:  07e8 20 58 52 20 59 52 20 53
.:  07f0 50 4d 52 58 47 3a 3b 4c
.:  07f8 53 86 fd 00 b7 fe 00 23
.:  0800 ff 00 02 ff 00 f4 fe 00
.:  0808 e1 fe 00 2d ff 00 2d ff
.:  0810 00 1b fd 00 00 00 00 00
```

Whew! TINYMON1 for the VIC is now entered. Check it with the following program:

Program 2: A Checking Program

Type the following direct line on the screen of your PET/CBM:

```
forj=1024to2071step8:t=0:fork=jtoj+7:t=t+peek(k):next:?t;:next
```

You should see the following numbers appear on the screen of your PET. Check them carefully. Each one represents one line of entry, starting at 0400 hexadecimal. If any of these totals is wrong, you've entered the line incorrectly.

The numbers in brackets appearing to the right won't appear on your screen; they are there to help you locate an incorrect line.

When you are satisfied that the program is entered correctly, SAVE it to cassette tape. It may now be loaded into your VIC.

462	255	506	399	575	541	592	511	(0400)
769	620	756	780	802	910	886	853	
801	784	876	840	835	1383	753	0	
1422	589	816	720	584	680	535	576	
944	972	1130	845	876	1357	1010	1188	(0500)
1311	852	898	1109	1125	897	809	1021	
1340	1078	1005	1212	905	902	770	1239	
762	1133	1388	652	659	629	1072	803	
748	150	617	413	1020	1030	1057	818	(0600)
944	844	705	831	939	1072	639	1033	
943	824	1137	970	929	1149	1395	940	
654	840	807	926	706	1146	1015	1146	
1175	742	563	645	695	860	1064	1042	(0700)
1235	1202	1355	922	1445	1346	789	1068	
1104	1204	975	1306	1339	1169	1168	1210	
1340	1204	972	522	460	520	591	942	
1010	1079	280						(0800)

VIC Color Tips

Charles Brannon
Editorial Assistant

Users of other computers, such as the ATARI or Apple, will find the VIC harder to use for color graphics because there are no dedicated statements for controlling these features. First time users will not know the difference, but this article should make things easier. Before we begin, it should be noted that there will soon be available a VIC Super Expander Cartridge that will add special sound and graphics commands to BASIC, as well as adding 3K of memory.

"Poking" Graphics

The only command that can be used for graphics besides PRINT is POKE. POKE places a number into a memory location. Its format is POKE A,B. A is the memory location, and B is the value to be placed there, zero to 255. Some spots in memory can control Input/Output chips, such as the Video Interface Chip inside of the VIC. Location 36879 is the control register for background and border colors. To get each combination, you place a number from zero to 255 into 36879, as previously mentioned. For any particular combination, you can look up the colors in the table at the end of this article (Table 2). There is an easier way, however, at least from a programming standpoint.

An Easier Way

The DEF FN command allows the programmer to design his own function. The VIC has, for example, the standard INT function. INT(X) will give you the whole-number value of the argument X by dropping the fractional portion. It does not round X. To provide a rounding-up function, we can use the DEF FN command. To round dollar and cents amounts, the statement DEF FNR(V)=INT (V*100+.5)/100 is executed at the start of the program. After that, FNR(X) will give you the rounded version of X, or any value in parentheses. PRINT FNR(3.1415927) will return 3.14, while PRINT FNR(500.076) will give 500.08 The R after the FN is a label to remind you what the function does. Here R stands for Round. These labels have the same format as numerical variable names.

What we want to do is to devise a formula which will give us the right number from the table for each color, one to sixteen. We will give the background color from one to sixteen through the FN routine, and it will give us the number ready for POKEing. To get any background color from any of the sixteen possible colors, just multiply the color number by 16 and then subtract eight. We can code this as DEF FNC(V)=V*16-8. Remember, V is just a dummy variable used to define the relationship of the argument (what we give the routine) in the formula. Next we use a little shorthand. The number 36879 (the color control) is a little hard to remember, and it does not look much different than any other memory location. We will make it easier to remember (make it *mnemonic*) by making it a variable, SCREEN = 36879. Now we can call forth any of our sixteen colors with the statement: POKE SCREEN, FNC (*color*), where *color* is the number from one to sixteen. This almost looks like a real graphics command.

Adding Border Colors

What about the border colors? In addition to the background, you can have eight border colors, numbered from zero to seven. This is one less than the corresponding number on the color keys (CTRL-6 would be 5). Now just take this number and add it to the number that you POKE into SCREEN. Now we just use: POKE SCREEN, FNC(*color*) + *border*, where *border* is the border color, zero to seven. If you don't use border colors, or don't add anything to FNC(color), then the border will be black.

Remember that if the background is the same color as the text, the cursor will become invisible. If you need to, set things straight with POKE 36879,27 or hold down RUN/STOP and press RESTORE to reset.

The little program at the end of this article demonstrates what I've been talking about by displaying all the combinations of screen and border colors. It's simple to figure out so look it over, and get to work on your VICtorious applications!

Table 1. Screen/Border Colors

Screen		Border	
1	Black	0	Black
2	White	1	White
3	Red	2	Red
4	Cyan	3	Cyan
5	Purple	4	Purple
6	Green	5	Green
7	Blue	6	Blue
8	Yellow	7	Yellow
9	Orange		
10	Light Orange		
11	Pink		
12	Light Cyan		
13	Light Purple		
14	Light Green		
15	Light Blue		
16	Light Yellow		

```
100 REM * ANOTHER RAINBOW *
110 DEF FNC(V)=V*16-8
120 SCREEN=36879
130 FOR BK=1 TO 16
140 PRINT "{CLEAR}{WHT}";
150 IF BK>1 THEN PRINT "{BLK}";
160   PRINT "SCREEN";BK
170   FOR BD=0 TO 7
180   POKE SCREEN,FNC(BK)+BD
190   PRINT,"BORDER";BD
200   FOR W=1 TO 500:NEXT W
210  NEXT BD
220 NEXT BK
230 POKE SCREEN,27
240 END
```

Table 2. POKE Values

BACKGROUND #	BORDER 0	1	2	3	4	5	6	7
1:	8	9	10	11	12	13	14	15
2:	24	25	26	27	28	29	30	31
3:	40	41	42	43	44	45	46	47
4:	56	57	58	59	60	61	62	63
5:	72	73	74	75	76	77	78	79
6:	88	89	90	91	92	93	94	95
7:	104	105	106	107	108	109	110	111
8:	120	121	122	123	124	125	126	127
9:	136	137	138	139	140	141	142	143
10:	152	153	154	155	156	157	158	159
11:	168	169	170	171	172	173	174	175
12:	184	185	186	187	188	189	190	191
13:	200	201	202	203	204	205	206	207
14:	216	217	218	219	220	221	222	223
15:	232	233	234	235	236	237	238	239
16:	248	249	250	251	252	253	254	255

VIC Memory Map Above Page Zero

Jim Butterfield
Toronto, Canada

Editor's Note: Next month we'll have a VIC zero page map and Jim's comments on the VIC's memory. — RTM

0100-103E	256-318	Tape error log
0100-01FF	256-511	Processor stack area
0200-0258	512-600	Basic input buffer
0259-0262	601-610	Logical file table
0263-026C	611-620	Device # table
026D-0276	621-630	Sec Adds table
0277-0280	631-640	Keybd buffer
0285	645	Serial bus timeout flag
0286	646	Current color code
0287	647	Color under cursor
0288	648	Screen memory page
0289	649	Max size of keybd buffer
028A	650	Repeat all keys
028B	651	Repeat speed counter
028C	652	Repeat delay counter
028D	653	Keyboard Shift/Control flag
028E	654	Last shift pattern
028F-0290	655-656	Keyboard table settup pointer
0291	657	Keymode (Kattacanna)
0292	658	0=scroll enable
0293	659	VIC chip control
0294	660	VIC chip command

0295-0296	661-662	Bit timing	
0297	663	RS-232 status	
0298	664	# bits to send	
0299-029A	665	RS-232 speed/code	
029B	667	RS232 receive pointer	
029C	668	RS232 input pointer	
029D	669	RS232 transmit pointer	
029E	670	RS232 output pointer	
029F-02A0	671-672	IRQ save during tape I/O	
0300-0301	768-769	Error message link	
0302-0303	770-771	Basic warm start link	
0304-0305	772-773	Crunch Basic tokens link	
0306-0307	774-775	Print tokens link	
0308-0309	776-777	Start new Basic code link	
030A-030B	778-779	Get arithmetic element link	
0314-0315	788-789	Hardware interrupt vector	(EABF)
0316-0317	790-791	Break interrupt vector	(FED2)
0318-0319	792-793	NMI interrupt vector	(FEAD)
031A-031B	794-795	OPEN vector	(F40A)
031C-031D	796-797	CLOSE vector	(F34A)
031E-031F	798-799	Set-input vector	(F2C7)
0320-0321	800-801	Set-output vector	(F309)
0322-0323	802-803	Restore I/O vector	(F3F3)
0324-0325	804-805	INPUT vector	(F20E)
0326-0327	806-807	Output vector	(F27A)
0328-0329	808-809	Test-STOP vector	(F770)
032A-032B	810-811	GET vector	(F1F5)
032C-032D	812-813	Abort I/O vector	(F3EF)
032E-032F	814-815	USR vector	(FED2)
0330-0331	816-817	LOAD link	
0332-0333	818-819	SAVE link	
033C-03FB	828-1019	Cassette buffer	
0400-0FFF	1024-4095	3K RAM expansion area	
1000-1FFF	4096-8191	Normal Basic memory	
2000-7FFF	8192-32767	Memory expansion area	
8000-8FFF	32768-36863	Character bit maps	
9000-900F	36864-36879	Video Interface Chip	
9110-912F	37136-37167	6522 Interface Chips	
9400-95FF	37888-38399	Alternate Colour Nybble area	
9600-97FF	38400-38911	Main Colour Nybble area	
A000-BFFF	40960-49151	Plug-in ROM area	
C000-FFFF	49152-65535	ROM: Basic and Operating System	

VIC usage: The 6560 V. I. Chip

$9000	Inter-lace (0)	Left Margin (=5)		36864
$9001	Top Margin (=25)			36865
$9002	Sc. Adds b9	# Columns (=22)		36866
$9003	b0	# Rows (=23)	Double Char	36867
$9004	Raster Value In: 68-b1			36868
$9005	Screen Address b13-b10	Character Address b13-b10		36869
$9006	Light Pen	Horizontal		36870
$9007	(option)	Vertical		36872
$9008	Potentiometer	X		36872
$9009	Sense (option)	Y		36874
$900A	^ Voice 1			36874
$900B	^ Voice 2			36875
$900C	^ Voice 3			36876
$900D	^ Noise			36877
$900E	Multi-Colour Mode (=0)	Sound Amplitude		36878
$900F	Screen Background	Fore/Back -Ground	Frame Color	36879

Values, where shown, are the normal default VIC values.

Light Pen and Potentiometer are implemented in hardware but not used in ROM programs.

VIC Usage: The 6522-A

$9110	RS-232 or Parallel User Port								37136
$9111	unused — see $911F								37137
$9112	DDRB (for $9110)								37138
$9113	DDRA (for $911F)								37139
$9114	T1-L	RS-232 Send speed;							37140
$9115	T1-H	Tape write timing							37141
$9116	T1 Latch								37142
$9117	T1 Latch								37143
$9118	T2-L	RS-232							37144
$9119	T2-H	Input timing							37145
$911A	Shift Register (not used)								37146
$911B	T1 Control	T2C	SR Control				PBLE	PALE	37147
$911C	RS-232 Send			Cb1 cont	Tape motor			CA1 cont	37148
$911D / $911E	NMI:	T1	T2	CB1: RS-232 IN			CA1: RSTR butn		37149 / 37150
$911F	ATN out	Tape sens	Joystick ----- Butn	0	1	2	In: Serl	In: Clok	37151

VIC Usage; The 6522-B

$9120	RS232 in / Joy 3		Tape out						37152
	---- Keyboard Row Select ----								
$9121	Keyboard Row Input								37153
$9122	DDRB (for $9120)								37154
$9123	DDRA (for $9121)								37155
$9124	T1-L	Cassette tape read;							37156
$9125	T1-H	Keyboard (interrupt)							37157
$9126	T1 Latch	timing							37158
$9127	T1 Latch								37159
$9128	T2-L	Serial Bus timeout;							37160
$9129	T2-H	Cassette R/W timing							37161
$912A	Shift Register (unused)								37162
$912B	T1 Control T2C		SR Control				PBLE	PALE	37163
$912C	Serial bus out			Clock line out				CA1 Contl	37164
$912D / $912E	IRQ:	T1	T2				CA1: Tape		37165 / 37166
$912F	unused — see $9121								37167

©

ZAP!!

Dub Scroggin
Ft. Walton Beach, FL

"Zap!!" is an exciting and challenging VIC-20 game program that is designed for up to six players and up to five rounds per player. Each player may select from any of five skill levels and may change levels each round, if desired. Using keyboard controls, players maneuver a block around the screen and through a field of randomly placed and color coded graphic figures. The object is to run over and erase as many of these figures as possible in two minutes, but also to avoid running into asterisks and being zapped. After the player block is moved it cannot be stopped, but the direction of movement may be changed. The higher the skill level, the faster the block moves and the more asterisks there are. The number of scoring figures is increased also so that a higher score is possible too.

The figures on the screen count differently toward the score. If a player is "zapped," he retains his score, but his time is over. Players may run off the screen, but may strike a hidden asterisk if they do so. A vertical wraparound feature prevents players from wandering too far off the screen. A variety of colors, graphics, and sound effects add excitement to the program.

As with most computer games, proficiency at Zap! will take some practice and a lot of concentration.

The player block is moved around the screen by the computer PEEKing at the keyboard to determine the value of the last control key pressed. A direction factor is then assigned to the variable DR (steps 590-620). When moving left, DR is -1, right is 1, up is -22 and down is 22. This factor is then added to the position of the block (B) (step 650). The old block is then erased by POKEing it to 32 (blank) and a new one is placed in position (step 570). This all happens so quickly that the illusion of motion is created.

Scoring and zaps are determined by PEEKs at the block's position to see if any other figure is there (steps 670-720). Depending on the figure found at "PEEK (B)", a score is assigned and the loop continues, or if the figure is an asterisk, a "Zap!!" routine is initiated and the round ends.

In each pass through the loop (steps 550-780), several things happen or are checked for. The elapsed time is printed and there is a check to see if the time is up. If so, the loop is terminated and the round ends. A block is POKEd into position B. Steps 580 and 585 provide the vertical wraparound effect. A check is made for direction change input from the keyboard. A tone is sounded based on the current direction of movement. The old position of the block is erased and a new position is calculated. A check is made to see if any figures have been struck. If so, they are either scored or, in the case of an asterisk, the loop is terminated. After a new total score is calculated and displayed, the loop begins again.

Steps 640 and 760 are time delay steps to slow the block's motion and to increase speed as the skill level increases. If a faster or slower movement is desired, these steps may be altered.

A number of REMarks have been included in the program listing as an aid to understanding it, but I recommend that they not be typed in on your computer. This program uses all but about 250 bytes of standard VIC-20 memory and including all the remarks may result in an "out of memory" error.

Good luck and I hope you enjoy the game.

```
10 PRINT"{CLEAR}"
20 DIM PL(6),R(5)
30 FORY=1TO5:FORX=1TO6:Z(X,Y)=0:NEXTX:NE
   XTY
40 C=30720:TB=0:TS=0
50 POKE36879,239
60 CP=0:GOTO810
70 PRINTTAB(3)"{06 DOWN}BY DUB SCROGGIN"

80 REM-404 WOODROW ST.,FT. WALTON BEACH,
   FL 32548
90 CP=1
100 FORT=1TO2000:NEXTT
110 PRINT"{CLEAR}"
120 PRINTTAB(5)"{DOWN}DIRECTIONS"
130 PRINTTAB(5)"7777777777"
140 PRINT"{DOWN}YOU WILL HAVE 2 MIN.":PRI
    NT"TO GET YOUR BEST SCORE"
150 PRINT"{YEL}MOVEMENT:":PRINT"{DOWN}CRS
    R DN=LEFT":PRINT"CRSR RT=RIGHT":
    PRINT"F5=UP"
160 PRINT"F7=DOWN":PRINT"{HOME}{04 DOWN}"

170 PRINTTAB(14)"{03 DOWN}{WHT}SCORING:"
180 PRINTTAB(14)"{BLK}W=1"
190 PRINTTAB(14)"{CYN}Q=2"
200 PRINTTAB(14)"{YEL}Z=3"
210 PRINTTAB(14)"{RED}S=5"
220 PRINTTAB(14)"{WHT}A=10"
230 PRINT"{DOWN}YOU ARE: {BLU}{REV} {OFF}
    "
240 PRINT"{DOWN}DON'T HIT A {PUR}*{BLU} O
    R":PRINT"YOU WILL GET {PUR}ZAPPE
    D."
250 PRINT"{WHT}{DOWN}PRESS ANY KEY TO STA
    RT"
260 GETA$:IFA$=""THEN260
270 PRINT"{CLEAR}{WHT}HOW MANY ROUNDS (1-
    5)"
280 INPUTRN:IFRN<1ORRN>5THENPRINT"HUH?":G
    OTO270
290 PRINT"{DOWN}HOW MANY PLAYERS":PRINT"(
    1-6)";
300 INPUTPN:IFPN<1ORPN>6THENPRINT"HUH?":G
    OTO290
310 FORR=1TORN
320 FORP=1TOPN:PRINT"{BLU}{DOWN}PLAYER #"
    ;P
330 PRINT"{DOWN}WHAT SKILL LEVEL?"
340 PRINT"PRESS 0,1,2,3 OR 4";
350 INPUT S
360 IFS>4 ORS<0THENPRINT"HUH?":GOTO340
370 PRINT"{CLEAR}{BLU}{REV}SCORE TO BEAT:
    ";TB:PRINT"{REV}SKILL LEVEL:";SL

380 PRINT"{REV}PLAYER #";PB
390 FORT=1TO2000:NEXTT:PRINT"{CLEAR}"
400 DEF FN A(L)=INT(RND(1)* L )+7702
410 FORF=1TO40-2*S:D=FNA(483)
420 POKED,87:POKED+C,0:NEXTF
430 FORF=1TO25:D=FNA(483)
440 POKED,81:POKED+C,3:NEXTF
450 FORF=1TO10+4*S:D=FNA(505)
460 POKED,42:POKED+C,4:NEXTF
470 FORF=1TO19:D=FNA(483)
480 POKED,90:POKED+C,7:NEXTF
490 FORF=1TO14:D=FNA(483)
500 POKED,83:POKED+C,2:NEXTF
510 FORF=1TO9+S:D=FNA(505)
```

```
520 POKED,65:POKED+C,1:NEXTF
530 B=7932
540 TI$="ØØØØØØ"
550 PRINT"{HOME}TIME:";12Ø-INT(TI/6Ø);"{L
    EFT}  "
560 IFTI/6Ø>=12ØTHENGOTO93Ø
570 POKEB,16Ø:POKEB+C,6
580 IFB<7636THENB=8229+B-7635
585 IFB>8229THENB=7636+B-823Ø
590 IFPEEK(197)=31THENH=19Ø:DR=-1:GOTO63Ø

600 IFPEEK(197)=23THENH=2ØØ:DR=1:GOTO63Ø
610 IFPEEK(197)=55THENH=21Ø:DR=-22:GOTO63
    Ø
620 IFPEEK(197)=63THENH=22Ø:DR=22
630 POKE36878,15:POKE36876,H
640 FORT=1TO3Ø-5*S:NEXTT
650 POKEB,32:B=B+DR
660 SC=Ø
670 IFPEEK(B)=42THENGOTO79Ø
680 IFPEEK(B)=87THENSC=1:GOTO74Ø
690 IFPEEK(B)=81THENSC=2:GOTO74Ø
700 IFPEEK(B)=9ØTHENSC=3:GOTO74Ø
710 IFPEEK(B)=83THENSC=5:GOTO74Ø
720 IFPEEK(B)=65THENSC=1Ø:GOTO74Ø
730 GOTO76Ø
740 TS=TS+SC
750 POKE36878,15:POKE36876,16Ø+PEEK(B)
760 FORT=1TO3Ø-5*S:NEXTT
770 PRINT"{HOME}{DOWN}SCORE=";TS
780 GOTO55Ø
790 POKE36878,15
800 FORPI=1TO4Ø:POKE36876,18Ø-PI:NEXTPI
810 PRINT"{PUR}{RIGHT}{Ø6 DOWN}&&&& &&&& ~
    &&&& && &&"
820 PRINT"{RIGHT}&&&& &&&& &&&& && &&"
830 PRINT"{RIGHT}   & &  & &  & && &&"
840 PRINT"{RIGHT}  &  &&&& &&&& && &&"
850 PRINT"{RIGHT} &   &&&& &&&& && &&"
860 PRINT"{RIGHT}&&&& &  & &"
870 PRINT"{RIGHT}&&&& &  & &     && &&{BLU
    }"
880 IFCP=ØTHEN7Ø
890 POKE36878,Ø:POKE36876,Ø
900 FORT=1TO2ØØØ:NEXTT:PRINT"{CLEAR}"
910 PRINT"{WHT}{DOWN}{REV}YOU LASTED";INT
    (TI/6Ø)-3;"{LEFT} SECONDS{OFF}"
920 GOTO97Ø
930 POKE36878,15:FORAC=1TO8Ø:POKE36876,21
    Ø-AC:NEXTAC
940 POKE36876,Ø:POKE36878,Ø
950 PRINT"{CLEAR}"
960 PRINT"{HOME}{Ø4 DOWN} {BLU}{REV}....T
    IME IS UP...."
970 PRINTTAB(6)"{DOWN}{REV}{WHT}SCORE=";T
    S
980 Z(P,R)=Z(P,R-1)+TS
990 IFTS>TBTHENTB=TS:SL=S:PB=P
1000 PRINT"{DOWN}{BLK}ROUND #:";R;"{DOWN}"

1010 FORX=1TOPN
1020 PRINT"{YEL}PLAYER #";X;":";Z(X,R)
1030 NEXTX
1040 TS=Ø:DR=Ø:H=Ø:PRINT"{DOWN}"
1050 NEXTP:NEXTR
1060 END
```

©

CAPUTE!

1. COMPUTE! #12, pg. 94. The authors suggest that the following lines should be changed to:

```
255 UG=GU:PRINT
257 IF GU=0 THEN 270
290 WD=WI: WF=WI: WT=(12-WI)/2
370 IF WI>4 THEN 300
530 IF WT<0 OR WT=0 THEN WT=1: RN=0
610 IF PEEK(KY)=251 THEN ME=ME-1: KK= -1
1060 IF WD<3 THEN PRINT "LITTLE";: GOTO 1200
1120 PRINT "CHEATER";
1200 PRINT "FOO";
```

2. COMPUTE! #17, pg. 112. The following changes to the "Atari Program Library" will lock all cataloged programs. Then, after adding new programs to a disk, only the unlocked (new) programs need to be cataloged:

```
420 IF B<3 THEN ? "DISK IS #";VOL$:
    XIO35,#3,0,0,"D:*.*":?:?"INSERT LIBRARY
    DISK"
370 IF IN$(1,1)="*" OR DSN$="DOS.SYS" OR
    DSN$="DUP.SYS" OR DSN$="MEM.SAV" OR
    DSN$="DISK.CAT" THEN 310
```

3. COMPUTE! #17, pg. 143. Mr. Swaim has suggested the following lines as an alternative way to load the X$ array for business keyboard users and to correct an error in transcription:

```
110 X$(1)=CHR$(164):X$(2)=CHR$(175): X$(3)=
    CHR$(185): X$(4)=CHR$(162)
112 X$(5)=CHR$(18)+CHR$(184)+CHR$(146):
    X$(6)=CHR$(18)+CHR$(183)+CHR$(146)
115 X$(7)=CHR$(18)+CHR$(163)+CHR$(146):
    X$(8)=CHR$(18)+CHR$(32)+CHR$(146)
```

4. COMPUTE! #17, pg. 152. The correct SYS is 7168 in line 120.

5. COMPUTE! #17, pg. 162. Table 1 is missing number 9, Subtraction:

Load FPAC1 with subtrahend

```
LDA AL       source address
LDY AH       for minuend
JSR $C58F
```

(Addressed value is loaded into FPAC2, FPAC1 is subtracted from FPAC2 and result in FPAC1; FPAC2 unchanged.)

Number 6 should include JSR $C0D1.

COMPUTE!'s Listing Conventions

Many programs which are listed in **COMPUTE!** use cursor control keys, color keys, and so forth. We have established a listing convention which we believe eases the task of typing programs in accurately.

Atari Conventions

For the Atari, all the editing and cursor-control characters are spelled out and surrounded by brackets: [CLEAR] for "clear screen." Other characters, such as CTRL-T (the "ball" character) will be listed as the "normal" character, but within brackets: [T]. A series of identical control characters will be indicated by a number within the brackets: [3 DOWN] means type the cursor-down key three times; [12 R] means type CTRL-R twelve times.

Two control characters, [=] and [–] should be shifted. Any reverse field text will be enclosed within vertical lines. (Press the Atari logo key [⅄] for each vertical line you see.)

PET/CBM/VIC Conventions

Generally, PET/CBM/VIC programs will contain bracketed words for any special characters: [DOWN] means the cursor-down key; [3 DOWN] means type the cursor-down key three times.

If a program line runs over onto the next line down, the ~ symbol indicates where the line broke (in case the number of spaces is unclear between quotes). An underline means that that key is shifted.

8032/Fat 40 Conventions

SET WINDOW TOP	[SET TOP]
SET WINDOW BOTTOM	[SET BOT]
SCROLL UP	[SCR UP]
SCROLL DOWN	[SCR DOWN]
INSERT LINE	[INST LINE]
DELETE LINE	[DEL LINE]
ERASE TO BEGINNING	[ERASE BEG]
ERASE TO END	[ERASE END]
TOGGLE TAB	[TGL TAB]
TAB	[TAB]
ESCAPE KEY	[ESC]

All Commodore Machines

CLEAR SCREEN	[CLEAR]
HOME CURSOR	[HOME]
CURSOR UP	[UP]
CURSOR DOWN	[DOWN]
CURSOR RIGHT	[RIGHT]
CURSOR LEFT	[LEFT]
INSERT CHARACTER	[INST]
DELETE CHARACTER	[DEL]
REVERSE FIELD ON	[RVS]
REVERSE FIELD OFF	[OFF]

VIC Conventions

SET COLOR TO BLACK	[BLK]
SET COLOR TO WHITE	[WHT]
SET COLOR TO RED	[RED]
SET COLOR TO CYAN	[CYN]
SET COLOR TO PURPLE	[PUR]
SET COLOR TO GREEN	[GRN]
SET COLOR TO BLUE	[BLU]
SET COLOR TO YELLOW	[YEL]
FUNCTION ONE	[F1]
FUNCTION TWO	[F2]
FUNCTION THREE	[F3]
FUNCTION FOUR	[F4]
FUNCTION FIVE	[F5]
FUNCTION SIX	[F6]
FUNCTION SEVEN	[F7]
FUNCTION EIGHT	[F8]
ANY NON-IMPLEMENTED FUNCTION	[NIM]

the SOFTWARE connection

The Most Exciting Video Games Available Today At The Lowest Prices

apple 5¼" Floppy Disks

David's Midnight Magic Retail $34.95 **Now $29.50** This pinball game by Broderbund is being offered for the first time.

Genetic Drift Retail $29.95 **Now $23.50** One of Apple's most popular games by Broderbund. You turn unfriendly life forms into friendly.

Firebird Retail $29.95 **Now $24.50** All new by Nasir. Piggo, the fearless firefighter, battles the incredible Firebird.

Horizon V Retail $34.95 **Now $29.50** Intercept an alien force before they destroy the planet. Nasir's latest graphic innovation.

Sneakers Retail $29.95 **Now $24.50** Endless excitement by Sirius stomping Sneakers and endless other creatures.

Hadron Retail $34.95 **Now $29.50** Battle in an asteroid field in this brand new game by Sirius.

Red Alert Retail $29.95 **Now $24.50** Newest Broderbund release

Apple Panic Retail $29.95 **Now $23.50** A number one seller by Broderbund.

Russki Duck Retail $34.95 **Now $29.50** First time released for any machine — recover the MX Missile plans hidden by foreign agents

Gorgon Retail $39.95 **Now $33.50** This Number 1 seller just re-released by Sirius. The earth has just entered a time-warp and the battle has just begun.

Space Eggs Retail $29.95 **Now $24.50** This number 1 seller is a real crack-up from Sirius.

Epoch Retail $34.95 **Now $29.50** A Number 1 seller from Sirius where your secret weapon is the fourth dimension.

VISA

ATARI® 800 5¼" Floppy Disks

Pathfinder Retail $39.95 **Now $31.96** Blast your way through the maze filled with nuclear waste and radioactive monsters in this never-before released game by Gebelli Software.

Match Racer Retail $29.95 **Now $23.50** Gebelli Software has you matching your wits and ability against your racecar opponent in a showdown of racing skill.

Andromeda 2 Retail $34.95 **Now $27.96** First time ever for any machine by Gebelli Software. Trapped inside a gigantic living organism you must find your way out before the antibodies get you.

The Software Connection
5133 Vista Del Oro Way
Fair Oaks, CA 95628

MAIL ORDERS: Send check or money order for total purchase price plus $1.50 for postage and handling. California residents add 6% Sales Tax.

Charge card orders may call (916) 989-3174. Subject to stock on hand. Prices subject to change.

master charge

SHADOWHAWK 1

A LONE SPACE PIRATE ATTACKS THE GALACTIC EMPIRE

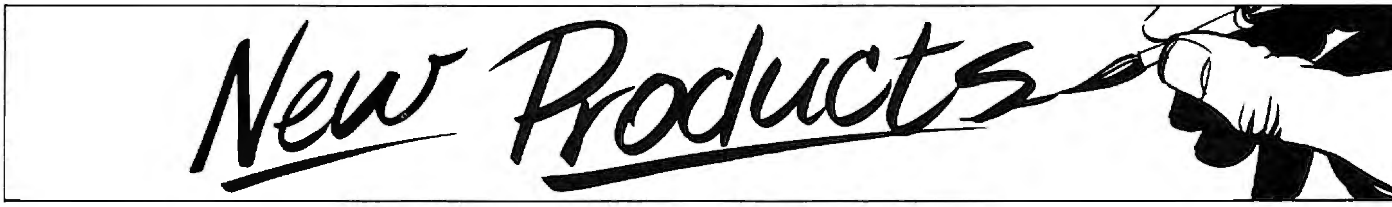
You are the sole surviving Naval Commander of the Free Space Confederation. The Galactic Empire has overrun the entire solar system . . . **except** for your remote outpost on a moon at the outer limits. Exploiting the unmatched ship **SHADOW HAWK I**,™ you prey on the Empire's merchant fleet to capture enemy material, which can be bartered for better weaponry, shielding, missiles, etc., for **SHADOW HAWK I**.™ But the Empire's interceptors, corvettes, lancers, destroyers, and cruisers are probing the galaxy for you. You must evade them and the deadly battle stations throughout. Your skill is measured by nine rankings, up to **STAR LORD**. Warning: You must be very, very good to reach **STAR LORD** rank! Very good indeed!

SHADOW HAWK I™ uses **THREE-AXIS** rotation, high resolution 3-D graphics! On **Apple II***, **DOS 3.3**, or **Atari 800***, 48K with disk drive. Joysticks required. Specify **Apple** or **Atari** on your order. Games are on **Dysan**™ diskettes. Game time: 15-30 min. **$49.95** ppd. VISA & MasterCard accepted. DEALER INQUIRIES ARE WELCOME

7561 Crater Lake Hwy.
White City, OR 97503
(503) 826-4640

*__Apple II__ and **Atari 800** are registered trademarks respectively of Apple Computer, Inc and Atari, Inc

HORIZON SIMULATIONS



New Journal For Math And Science Teachers

A new periodical for educators interested in using computers to teach mathematics and science is *The Journal Of Computers In Mathematics And Science Teaching*. The *Journal* is published by the Association for computers in Mathematics and Science Teaching, a professional non-profit organization. It is published quarterly in September, December, March and June. ACMST membership is $7.00 per year and includes a subscription to the Journal. Single copy price is $2.50. Contact them at P.O. Box 4455, Austin, TX 78765.

Dental Computer Newsletter

The DCN represents an international group of dentists, physicians and office management people who have interests in office computers. Though the emphasis is on micro computers, many members use minis. We cater to all makes and brand names. DCN offers members:

1. A monthly newsletter
2. Software exchange
3. Advice and experience
4. Access to members world wide
5. Computer bulletin board

1982 annual membership dues are $15.00. A little over a dollar per month. Overseas subscriptions are $23.00 (US). Membership runs from January to January. If you join mid year, we will supply you with the year's back issues. Checks can be sent to:

Dental Computer Newsletter
E. J. Neiburger DDS–Editor
1000 North Ave.
Waukegan, IL 60085 USA

Orders: You may order past issues starting with V1:1 (Nov. '78) for $15 per year ... $45 for all past issues. Membership/equipment listings are $5.00. Commercial software and DCN software exchange lists (24 pages+) are $4.00.

Sue Neiburger RN
Managing Editor

E. J. Neiburger DDS
Editor/Publisher

Please feel free to call any Tuesday, Thursday, Saturday or Sunday 9-12 noon (CST) (312)223-5077.

Capital Children's Museum And Reston Publishing Company To Develop Software

The Capital Children's Museum of Washington, DC and Reston Publishing Company (A Prentice-Hall Company) are pleased to announce an agreement to produce a series of educational book/software packages for microcomputers. The series is designed to promote a creative, interactive use of computers by children, parents, and teachers. The product will be implemented initially on the Atari 800 Personal Computer System; design and testing are being managed by Superboots, the software development arm of the Capital Children's Museum.

The first package, entitled *Paint*, will be available in early 1982. *Paint* will be a versatile educational tool that will be suitable for use either in the home or in a classroom setting. The book accompanying the software will be a guide to a wide range of activities which parents or teachers can use to extend a child's interest in computer learning.

Early in 1981, Atari made a significant donation to the Capital Children's Museum in the form of 30 Atari microcomputer systems. This contribution has allowed the Museum to establish a computer learning environment called *Future Center*, to put computer programs in exhibits, and to utilize other computers in Superboots. Superboots is the software development lab where computer programs are created. Software is used both in *Future Center* and in exhibits and is marketed outside the Museum through Reston Publishing Company.

For more information, contact: Bob Evans, Administrator, Superboots, Capital Children's Museum, 800 Third Street, N.E., Washington, DC 20002 (202)543-8600. Nikki Hardin, Editor, Reston Publishing Company, 11480 Sunset Hills Road, Reston, VA 22090 (703)437-8900.

Artworx Announces New Atari Software

Arthur M. Walsh, Manager of Software Products for Artworx Software Company, 150 N. Main St., Fairport, New York 14450, announced the addition of nine

CRYSTALWARE

ATARI 48K — w/disk

TRS-80 48 — w/disk

APPLE 48K — w/disk

THE FINEST IN FANTASY GAME SOFTWARE

At Crystal we are doing our best to provide the finest state-of-the-art graphic adventure software in the world. Our list of credits include the first indoor-outdoor graphic adventure, the first multi-disk graphic adventure, and now for the Atari, the first graphic adventure in the world which includes screen scrolling and animation. The era of the text adventure and games which are simple combinations of static graphics and text is rapidly drawing to a close. We attempt to utilize the full potential of your computer. True, many of our games use up to 48K and we only deal in disk products, but there are a lot of users out there who have worked hard to upgrade their systems to the max and we think they deserve games that will give their computer system a run for its money.

★★★ ADVENTURE GAMES ★★★

1-THE HOUSE OF USHER- Haunted house type adventure game with scrolling in the Atari version. Wander the creepy hallways of the three story castle based on Edgar Allen Poe's short story of the same name. Written in graphics, of course, with animation and sound. We have introduced a new mystery for another $100 prize. $29.95/1 disk

2-FANTASYLAND 2041 A.D.- The largest disk based adventure game in the world (that we know of). Enter the Hall of Heroes and prepare yourself for the greatest fantasy-role-playing game you will see for years to come. To win you must survive Congoland, Arabia, King Arthur, Captain Nemo, Olympus (a sea voyage), and Dante's Inferno (Hell itself). In both the Atari and Apple versions it takes up more than 400,000 bytes of memory and uses more than 400 hires screens. The winner of the contest described in the manual with this game will receive $1000.00 and a bronze trophy. We have pushed the award date forward to February 1982 to allow more people to participate in the contest. $59.95/6 disks

3-GLAMIS CASTLE- Yes, Pat and I are on our way to Britain to stay in the dreaded Glamis Castle. If we survive our real life adventure, we'll be measuring it and will be able to provide you with a 3-D game based on this ancient haunted site where King Duncan met his end at the hands of Macbeth. Our good friend, Mark Benioff, after much research, said there's a mystery room that has never been found in this castle and a half beast, half-man creature that guards a treasure therein. Our stay will be covered by the British media and we hope to share our experience with you through the writing of this game. $49.95/2 disks

4-BENEATH THE PYRAMIDS- You are an archaeologist in 1932 and must find your way through the perilous chambers beneath the pyramids to discover a golden statue of the cat goddess Bast. This game is in hires graphics, includes sound, your little man actually moves through the corridors which you can see on the screen. The monsters are animated and very agressive. There is a new $100 prize for the first to solve the mystery; which is a toughie! $29.95/1 disk

★★★ SPACE GAMES ★★★

5-GALACTIC QUEST- An excellent combination of Star Trek and Space Trader. Battle the animated Vegan fighters as you warp from galaxy to galaxy. At the same time, you may land on and trade with hundreds of planets. Super hires graphics and lots of sound. This has been one of our most popular games. $29.95/1 disk

6-SANDS OF MARS- Take an exciting voyage to the planet Mars via the Starship Herman. This game compared to the rest, is second only to Fantasyland 2041 A.D. It includes scrolling on the Atari and hundreds of full screen graphics. You can move your little man through the terrain of Mars; if, of course, you survive the exciting journey to Mars, which occupies the whole first disk. There is a new mystery and another $100 prize just waiting for some clever adventurer out there. Good luck! $39.95/2 disks

★★★ WAR GAMES ★★★

7-WORLD WAR III- You Atari gamers will have to see this in the Atari version to believe it! If your tired of war games which take 15 minutes a move and have a manual the size of a telephone book; but still want a complex, real-time action war game-this is it! It is designed for two arm-chair generals which may manuever up to 128 seperate type of units at a time. The game displays a map of Iran & Iraq in the first scenario and later on you will find yourself moving nuclear submarines and battleships through two world wars. This is not a boring copy of a board based game but an original war game which takes a lot of skill and may take weeks to play. $29.95/1 disk

8-WATERLOO II- If you had been Napolean would you have done a few things differently? Well as you approach this final battle you are equipped with the same forces, face the same opposition, and survey the same terrain which he did. We have done a great deal of research to make this historically accurate as well as extremely complex. Even the angle of sight, fatigue of the individual soldier, and his psychological profile are included in the calculations. Oh by the way, your opposition is no slouch. You may find it more difficult to change the course of history than you think! $49.95/2 disks

★★★ ARCADIA ★★★

9-LASAR WARS- Hires-3d space war simulation. Protect the earth from alien invaders. $29.95

10-LITTLE CRYSTAL- The first of our line of education software, which will be completed by December. It includes a very fine version of Hangman, Mr. Music; which transforms the computer into a piano, Gunk-a hilarious shoot-em up game, and Storytime- an anthology of bedtime stories featuring Herman, the cat, Oscar, the Hamster, and of course, Little Crystal. $39.95

11-IMPERIAL WALKER- A fine game pack written by our Atari programmer, Michael (graphics) Potter. Includes the Walker animation which is superb, Gunfight, and Lasar Nim, a game of 'how many robots'. $29.95

12-ADVENTURE PACK- (#1-4) $112

13-SPACEOUT- (#5-6) $58

14-THE WARRIOR- (#7-8) $64

15-ARCADE- (#9-11) $60

VISA — master charge

Crystal has many other fine fantasy and space games. For a copy of CRYSTAL VISION which includes a complete catalog please send $3.00 to the address below.

Our order lines are open 24 hrs. a day 7 days a week.

(408) 778-2966 — CRYSTAL COMPUTER — 17120 Monterey Rd., Morgan Hill, CA 95037

CRYSTALWARE

ATARI 48K — w/disk

TRS-80 48 — w/disk

APPLE 48K — w/disk

THE FINEST IN FANTASY GAME SOFTWARE

"Having previewed over fifty of your competitors' games, I can assure you that your use of scrolling far exceeds anything I've seen for the Atari and, of course, for the Apple. I'm very impressed by the dedication and quality that your company exhibits by virtue of this demo." **David Sosna** — Associate Producer, Universal Pictures

Crystal has done its best to become the Porsche of the computer game industry. New scrolling techniques, video disk games, a real-life fantasyland — our mad programmers toil onward with little food or sleep to produce some incredible firsts in the microcomputer world. If you are an unappreciated genius and want to join our staff to help create the world of tomorrow today, give me a call. Our magazine *Crystal Vison* will within the next month have a circulation of 80,000 and we look forward very soon to producing our first full length motion picture. I'd like to thank my friends at Votrax and Axlon for giving us the tools (128K RAM for Atari and a vocal text synthesizer) to truly produce some programming miracles.

★ ★ ★ NEW RELEASES ★ ★ ★

THE CRYPT — One evening you awake at sunset to find yourself in what appears to be an endless cemetery. Although defenseless, you must somehow find your way out or perish from the hideous assaults of flesh-eating zombies, rats, vampires, werewolves, and other repulsive monstrosities. To escape you may have to descend into the catacombs beneath the cemetery. This game is a little different from the others of our series because we use a lot of static graphics to set the mood. It is similar in some respects (without any copying intended) to those of our friends at On-Line who produce excellent static graphic adventures. You must use all your common sense and a great deal of courage to escape from this perilous adventure alive. We have made it so nearly impossible that the first player to do it successfully will receive a $200.00 prize. **$49.95 2 disks**

QUEST FOR POWER by Mark Benioff — An extraordinary game with the adventure and magic of Arthurian legend. Join Galahad as he leaves Camelot in search of the Scroll of Truth. Explore the treacherous depths of the Caves of Somerset, visit the medieval city of Essex. Along the way you will meet powerful wizards and great prophets. The villages of Sunderland and Leeds dot your path. Somewhere in an evil castle called Skenfirth, lurks the devil himself, while the Evil Giant Gogmogo, hungry for human prey, roams the forests. In Fantasyland tradition we include 64 full screens of hires scrolling and some sensational graphic and animation sequences. Well worth the **$39.95 1 disk**, enjoyable to all ages.

★ ★ ★ GALACTIC EXPEDITION ★ ★ ★

The year is 3021, almost 100 years since the expedition to the Sands of Mars has returned. The Starship Herman now rests quietly in the Zikon Museum in New Brisbane. It's nearly 80 years since World War III, the Ames Research Center celebrates its 150th anniversary, and you stand at the unveiling of a truly technological wonder — the first ion-propelled vessel, saucer-shaped Lady Joanne, its viewport of pure diamond, its hull of synthetic emeralds. The Martian glyphs of the Meshim and those of Lemuria have now been deciphered and it appears that a much greater mystery is about to unravel. 7 planes and 7 doors — 7 guardians and 7 candles. 7 strange new worlds await the ultimate adventurer to unlock a timeless secret. The starship may seem strange and unfamiliar to our veteran adventurers, faced with its marvelous new technology; this craft must be flown by constant monitoring of ion stabilizers. During your galactic expedition you are surrounded by the flickering heavens, beset by meteor showers and time-warps. Each unique world holds one of the 7 keys to unlock the Great Mystery. The games all run off the Main Module which also is a game unto itself.

From Earth to Moon — On the Moon's dark side lie entrances to caverns extending to the moon's hollow core which contains a timeless secret. Here live a race of burrowing creatures, who have built vast earthen cities with storehouses full of precious stones. Gravity is extremely critical and you must use all your skills to manually land your craft. This first Master Disk contains the dos needed to run additional scenarios. Its price is **$39.95** and includes 64 screens of Hires graphics.

Mists of Venus — On Venus' ever hot surface are endless jungles and swamps. The air is unbreathable and spacesuits and oxygen must be carried. This world is especially treacherous with all sorts of loathsome creatures and hardly any place dry enough to land your ship. Beneath the green seas our adventurer may find the second key to solving the Mystery. **$29.95** (must have Master Disk to run)

Planet Herman — It is hard to tell where Herman's atmosphere ends and the surface begins. Much of this adventure will have the feeling of a starship submarine. Navigating around Herman is very dangerous but with a computer on board Lady Joanne it may be just possible. This senario costs **$29.95** and needs the Master to run

The Asteroid Belt — Every play something oids. A combination of the best machine language sub-routines of our new Crystaloids with a fast moving adven ure game. Penal colonies, lurking pirates, and some unusual forms of scavenger life exist here. It's difficult to travel in the Asteroid Belt without getting blown up. Perhaps you should find some expert help by rescuing a pilot, who is also a sentenced thief or murderer, from one of the penal colonies. There are places for trading and you may wish to indulge yourself with a visit to the sensual Pleasure Planet. **$29.95** (needs Master Disk)

Uranus - World of Ice — A freezing place with nights of −200° F. Bring along Thermasuits, as well as some Laars with which to battle the Grungik, a 12 foot tall relative of Big Foot, fond of human flesh. Uranus also has a secret inner labyrinth with tropical flora and fauna. However, the King of the Ice Planet, Norion may have his own idea about your trespassing. Without proper clothing, weapons and supplies, your stay here may be very exciting and very short. **$29.95** (needs Master Disk to run)

Jupiter - World of Dwarfs — How would it feel to weigh 300 or so lbs.? A trip to Jupiter should fill you in fast. There is a particularly interesting red spot on Jupiter and a curious set of moons. Picking up some antigravs will help. Landing should really tax your energies. In the Jupiterian atmosphere, you fall fast! Be prepared to use 10 times the normal amount of fuel. Better find the 6th key quickly before your fuel and food are exhausted. **$29.95** (needs Master Disk)

The Crystal Planet - You will have to embark on this final portion of your expedition ignorant of what you may encounter here on this mysterious planet, excepting that the 7th world holds the ultimate key to winning the contest. **$29.95** (needs Master Disk)

The Contest — To the Winner with the highest score, who solves the mystery by November of 1982 will go $5000.00 in cash. Good Luck!

★ ★ ★ ★ ★ ★

GLAMIS CASTLE — According to ancient legend and records this castle is one of the most haunted sites in Great Britain. One Lady Glamis, known to be in league with the devil, liked to send out a destructive demon to harrass the townspeople. She finally was burnt at the stake on Castle Hill, cursing as she died all future generations of the Lyon family. Her demon still seems to haunt that spot, murdering the curious who stray up to Castle Hill after dark. The curse stipulated that each succeeding generation would have at least one child, often female, who would be a vampire. When an heir comes of age, there is a secret ceremony in which the heir, his father, and the steward take crowbars and chip away plaster concealing a hidden chamber, known only to them, that Earl Patie used when he gambled with the devil. Another tradition says that a creature, half-man, half-beast stalks the passages in the walls of Glamis to insure the fulfilling of the curse. The mystery, of course, is to determine the location of this secret chamber. Our game, occupying 2 disks, will have as exact a replica of the castle as possible. It's definitely one of a kind! And we will be offering a $500 prize to the first person daring enough to solve the centuries-old mystery of Glamis Castle. **$49.95 2 disks.**

Crystal has many other fine fantasy and space games. For a copy of CRYSTAL VISION which includes a complete catalog please send $3.00 to the address below.

Our order lines are open 24 hrs. a day 7 days a week.

(408) 778-2966 **CRYSTAL COMPUTER** 17120 Monterey Rd., Morgan Hill, CA 95037

VISA

new programs to its present line of software for home computers. All nine programs are available for the Atari computer. They include space games (ENCOUNTER AT QUESTAR IV, $23.95; ROCKET RAIDERS, $19.95; SPACE TRAP, $14.95), a flight landing simulator (PILOT, $16.95), an excellent blockade game (BLOCKADE, $14.95), two suspenseful adventures (CRANSTON MANOR, $21.95 diskette; THE VAULTS OF ZURICH, $21.95), a text editor (TEXT EDITOR, $39.95 diskette) and a "player missile" editor (PM EDITOR, $29.95).

These and other Artworx programs are available at computer stores or can be ordered directly from Artworx toll free at 800-828-6573 or 716-425-2833.

Cimarron Releases File Handling System For The Commodore Business Computer

Costa Mesa, CA — CIMARRON CORPORATION announced today that it is making its proprietary file handling system called CMAR available to systems houses and retailers who are developing business applications software for the Commodore line of small business computers.

CMAR is a keyed file access method that provides the foundation for CIMARRON'S Legal Time Accountinng and Medical Accounting packages marketed by Commodore nationally. CMAR is compatible with all present Commodore disc subsystems utilizing the existing disc format. It is written in 6502 machine language and interacts directly with Commodore Basic 4.0.

With its post "Binary Tree" technology referred to as *Inverted Key File Method*, CMAR offers a

commodore

Check Our New Lowest Prices!

EQUIPMENT:

	VIC-20 (Includes RF Modulator)	265
	CBM 4016 CPU (40 Col. Screen, 16K RAM)	790
	CBM 4032 CPU (40 Col. Screen, 32K RAM)	990
	CBM 8032 CPU (80 Col. Screen, 32K RAM)	1090
	CBM 8096 CPU (80 Col. Screen, 96K RAM)	1590
NEW	CBM Micro Mainframe (Super PET)	1690
NEW	CBM 2031 Single Disk Drive (170K per 5¼ Diskette)	570
NEW	CBM 4040 Dual Disk Drive (170K per 5¼ Diskette)	990
	CBM 8050 Dual Disk Drive (½ Meg per 5¼ Diskette)	1340
NEW	VIC Tractor Feed Printer (80 Col., 30 CPS)	325
	CBM 4022 Tractor Feed Printer	625
	CBM 8010 Phone Modem	225
	CBM C2N Cassette Deck (New Style)	65
	CBM CPU/IEEE Cable	35
	CBM IEEE/IEEE Cable	40
	Tally 8024-7 (7x7 Matrix) Printer	1590
	Tally 8024-9 (7x9 Matrix) Printer	1690

SOFTWARE:

Wor[illegible]rdprocessor	295
Wordpro[illegible]rdprocessor	325
OZZ Data Ba[illegible] System	295
Visicalc	175
Tax Preparation System	395
IRMA Information Retrieval	395
Dow Jones Portfolio	115
Pascal Development Package	225
Assembler Development Package	89
Personal Tax Calculator	59

PUBLICATIONS:

CBM User Guide	6.49
CBM Basic 4.0 Reference Manual	7.97
CBM Disk Manual	6.49
CBM Printer Manual	6.49
MOS Hardware Manual	5.75
MOS Programming Manual	5.75
The PET Revealed	17.49
Library of PET Subroutines	17.49
Commodore Software Encyclopedia	4.95

VIC SOFTWARE:

Six Pack A	- Includes Car Chase, Blue Meanies, Space Math, Slither/Super Slither, Biorhythm Capability	43.95
VIC Games	- Includes VIC Trap, Seawolf, Bounce Out	24.95
Household Finance	- Includes: Part I - Entering & Update; Part II - Summing & Displaying; Part III - Budgeting & Graphing; Part IV - Deductibles Analysis	14.95
Home Inventory	- Includes: Part I - Entering Inventory; Part II - Evaluating Inventory	14.95
Logic Games	- Includes Code Maker, Code Breaker	14.95
Rec/Ed I	- Includes Hangman, Hangmath	14.95
Rec/Ed II	- Includes Math Hurdler, Monster Maze	14.95
	Inquire about other VIC software!	

COMING SOON:

More [illegible]pher[illegible]d Software
CBM 82[illegible] Dual Disk Drive
CBM 8061/8062 8 inch Disk Drives
Data Acquisition and Control Devices

All items Insured
COD – UPS
Prepaid Orders Shipped Free
In stock items shipped within 48 hours

MASTERCHARGE OR VISA ADD 3%

GA. RESIDENTS ADD 4% SALES TAX

MART

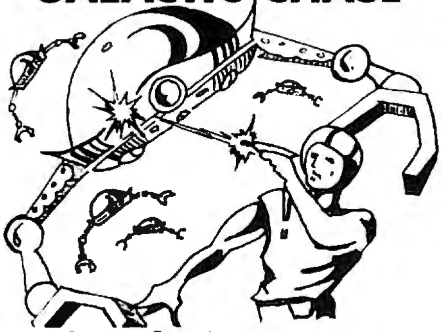
P.O. Box 77286
Atlanta, Ga. 30357

404-981-5939
Call 9 AM-6:30 PM EST

Special Offer		
	8032 CPU	$1090.00
	8096 CPU	$1590.00
	Micro Main Frame	$1690.00

THE PROGRAM STORE CALL TOLL FREE 800 424-2738

GALACTIC CHASE



From Spectrum Computers
An all machine language arcade game that combines challenging play with great graphics and sound. Insect-like alien ships form ranks above you, moving back and forth as they ready their assault. Suddenly they start breaking formation to swoop down on you, the lone defender. Fight them off with swift missiles . . . until their invisible ray slows your missiles down. May be played by 1 or 2 players, with two skill levels.

16K tape...$24.95
32K disk...$29.95

TRICKY TUTORIALS

From Santa Cruz Software
A novel approach to learning about the special programming functions of your Atari. Each TRICKY TUTORIAL combines printed information with several programs (complete with listings) that demonstrate the concept being presented. You are encouraged to modify and incorporate the programs in your own programming!

#1 - DISPLAY LISTS
16K tape or disk...$14.95
#2 - HORIZONTAL & VERTICAL SCROLLING
16K tape or disk...$14.95
#3 - PAGE FLIPPING
16K tape or disk...$14.95
#4 - BASICS OF ANIMATION
16K tape or disk...$14.95

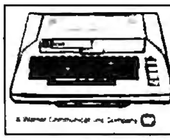
Also from Santa Cruz: BOB'S BUSINESS
A baker's dozen programs for business and personal applications: amortization, mortgage, saviongs, averages, square feet/yards, paycheck, investment, mortgage comparisons, property expense, bar graphs, decimal/hex, U.S./metric, checkbook.

32K tape...14.95

Air Traffic Controller

By David Mannering from Creative
Wonder if the air traffic controllers are really under stress? Want to see what all the fuss is about? This program will give you a taste of what goes on in those towers as you try to guide 26 aircraft safely through your airspace. This advanced version has five separate control areas from which to choose, as well as other enhancements.

16K tape...$11.95



Programs for ATARI 400/800
Unless Otherwise Noted

GIN RUMMY 3.0

By S. Silverman from Manhattan Software
Using Atari sound for input cues, this program presents your hand, the discards and the computers moves. All input is via the joystick, and you can manipulate (reorder) the cards in your hand any time it's your turn. Scoring of both hands is done by the computer, as is the overall game scoring. It makes a good Gin Rummy opponent -- what more can we say?

32K tape...$16.95

STARBASE HYPERION

By Don Ursem from Quality
Fight wave upon wave of Empire warriors as you carry out STARCOM orders and defend Starbase Hyperion. Very different fron the arcade-type space games, STARBASE HYPERION is a complex tactical simulation. You can choose from six levels of play with various scenarios within each level. Comes with full instructions and a Battle Manual.

24K tape...$19.95
24K disk...$22.95

EASTERN FRONT (1941)

By Chris Crawford from APE
A map-based simulation of Operation Barbossa, the german invasion of Russia. A complex and accurate war game, EASTERN FRONT pits you, as German Commander, against the terrain, the weather, and the Russian forces. Good use of colorful graphics and sound, and well written documentation make this game a winner!

16K tape...$26.95
32K disk...$29.95

SHOOTING GALLERY

From Analog
A remarkably realistic shooting gallery, complete with carnival music. Use your joystick to shoot at moving owls, rabbits, ducks, and clay pipes. Hit stars and targets for more shots. If you can shoot them all, you'll get a try at the raging bear; if any ducks fly south, they'll eat your bullets. Great family fun!

16K tape...$18.95

ASSEMBLER BOOK

By Don & Kurt Inman
While the ATARI ASSEMBLER CARTRIDGE comes with an operating manual, it assumes that you already know assembly language. If you're new to the Atari or its 6502 processor, this book is a must.

The Inmans guide you through the rudiments of this fascinating type of programing in clear, easy steps. Includes full listing and description of 6502 mnemonics and addressing modes.

$12.95

BULLETIN BOARD 2.0

By Skip Potter from Showcase Software
A complete package that allows you to set up your own computer bulletin board. Full documentation makes it easy to define the special functions to best serve your needs. Requires Atari 400/800, 810 disk drive, auto-answer modem, and a phone line.

24K disk...$59.95

DISK DETECTIVE

By M. Siegel from Datasoft
Utility programs to unlock the mysteries of your disk system. DETECTIVE lets you examine and modify your disks, sector by sector. DISKMAP provides a graphic display of a disk, noting which tracks and sectors contain data and which do not. A must for disk drive owners.

16K disk...$29.95

SCOTT ADAMS
ADVENTURES

By Scott Adams from Adventure International
Until you've played an Adventure, you can't appreciate the hours of challenge and fun built into each program. Each tests your powers of reason and deduction as you attempt to accomplish your mission using the implements you have, find or devise. Tape for 24K Atari

1. ADVENTURELAND $19.95
2. PIRATE'S ADVENTURE $19.95
3. MISSION IMPOSSIBLE $19.95
4. VOODOO CASTLE $19.95
5. THE COUNT $19.95
6. STRANGE ODYSSEY $19.95
7. MYSTERY FUN HOUSE $19.95
8. PYRAMID OF DOOM $19.95
9. GHOST TOWN $19.95

Visit our other stores: Seven Corners Center · Falls Church, VA & W. Bell Plaza · 6600 Security Blvd. · Baltimore, MD

TO ORDER CALL TOLL FREE 800 424-2738 For information Call (202) 363-9797

MAIL ORDERS: Send check or M.O. for total purchase price, plus $1.00 postage & handling. D.C. residents, add 6% tax. Charge card customers: include all embossed information on card.

THE PROGRAM STORE
4200 Wisconsin Avenue NW, Dept. U01 Box 9609
Washington, D.C. 20016

new level of efficient disc management for programs requiring sophisticated multi-key accessing. CMAR performs all read, write, change and delete functions and supports up to five opened keys concurrently. Future hard disc emulations with an unlimited number of secondary keys will provide a "big system" working environment at low cost.

With a random access speed gain of nearly 60% and a sequential speed gain of over 45%, even programs with heavy posting routines, sorts and fetches can become practical and manageable.

CMAR integrates its dynamic file substructure with the application program and Commodore Basic. Since the file system is dynamic, no file reorganization is ever required as in ISAM and Binary Tree configurations.

CMAR requires less than 10K bytes of memory and can be expanded to any operational file size, limited only by disc storage capacity. Files can consist of 2,000 to 20,000 keys depending on key size. Additionally, two kinds of key files are permitted; a primary key and its associated data record both based on relative file format.

CMAR is available through CIMARRON and is priced at $99.00 dealer net. A multiple quantity discount structure for application developers is also offered. Documentation only can be purchased separately for $10.00 on diskette in WordCraft 80 format.

OSI Screen Editor

Edit All is a full screen editor for OSI computers from DMP Systems, 319 Hampton Blvd., Rochester NY 14612. It replaces the standard I/O routines to allow the user to edit any program line that is on the screen. As editing takes place, the line is dynamically expanded or contracted.

Edit All supports a scroll window screen handler that allows you to define where on the screen you want your output to go. All output to the screen is via a window whose length, height and width are all user changeable. Full cursor control is supported along with an instant screen clear. Edit All works with OS65D Basic and Assembler. Price is $19.95 for 5¼ disk.

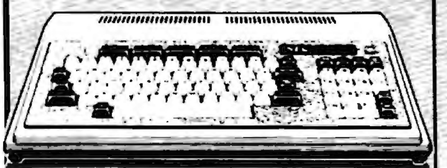
World Book Encyclopedia Now On CompuServe

Columbus, Ohio — The CompuServe Information Service today announced the signing of an agreement with World Book

consumer computers Mail Order DISCOUNTS

ORDER TOLL-FREE

NEC Microcomputer



PC-8001A 32K Computer	CALL
PC-8012A I/O Unit w/32K RAM	CALL
PC-8013A Dual Mini-Disk Drive Unit	CALL
PC-8001 Multi Cardware (FDI/O & 32K)	CALL
CP/M 2.2 Operating System for NEC	129
WordStar configured for NEC	299
SuperCalc configured for NEC	279
NEC Wordprocessor & Accounting Software	CALL

Many more software packages and languages; (Pascal, Fortran, Cobol, etc) are available configured for the NEC 8001A Computer.
Please call or write for a product price list.

ATARI 800 16K $749

Atari 400 w/16K	349
410 Program Recorder	65
810 Disk Drive	449
825 80 col. 7x8 Dot matrix impact printer	699
822 40 col. Quiet Thermal Printer	349
850 Interface Module	159
Atari 16K Ram Module	69
Axlon Ramcram 32K Module	189

Video Monitors

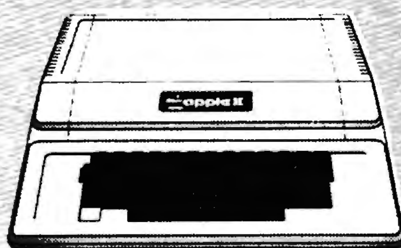
Amdek/Leedex Video 100 12" B&W	155
Amdek/Leedex Video 100G 12" Green Phospher	179
Amdek (Hitachi) 13" Color w/audio output	389
NEC 12" Green Phospher Display JB-1201M	CALL
NEC 12" Lo-Res Color Display	CALL
NEC 12" Hi-Res RGB Color Display	CALL
Sanyo 9" B&W Display	185
Sanyo 9" Green Phospher Display	CALL
Sanyo 12" B&W Display	269
Sanyo 12" Green Phospher Display	285
Sanyo 13" Color Display	449
Zenith 12" Green Phospher Display ZVM-121	149

ZENITH 12" GREEN $149

VIC20 $259

Personal Computer
Color • Sound • Graphics
Call or write for more info.
Disk drives available soon!

apple computer
Authorized Dealer



APPLE II PLUS

16K NOW $1025
48K NOW $1089
64K* NOW $1199

*48K Apple with 16K RamBoard

APPLE DISK DRIVES $439

w/controller and DOS 3.3 **$499**

Apple Cards and Hardware

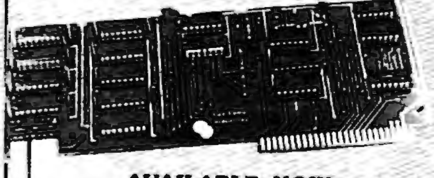
Language System w/Pascal & BASICS	379
Silentype Printer w/Interface card	349
Hayes Micromodem II	299
Novation Apple-Cat	339
Videx Videoterm 80 column card	269
Videx Keyboard Enhancer	115
Z-80 Softcard by Microsoft	299
16 K RamCard by Microsoft	169
CPS Multi-function card	189

Software for the Apple

VisiCalc version 3.3	159
VisiFile (NEW data base manager)	199
VisiTrend/VisiPlot	219
DB Master	169
WordStar (Apple 80 col. version)	249
Dow Jones Portfolio Evaluator	45
Apple Post	45
Apple Writer	65
Dow Jones News & Quotes Reporter	85
Apple Plot	60
Tax Preparer	99
Real Estate Analyzer	129

16K RAMBOARD *by ConComp* for Apple II Computers

FOR ONLY $129.95



AVAILABLE NOW

S-100
California Computer Systems

Floppy Disk Controller	**$369**
64K Dynamic Ram Board, 200ns	**$499**
Z-80 CPU board w/monitor ROM	**$269**
16K Static memory board, 200ns	369
32K Static memory board, 200ns	599
S-100 12 Slot Mainframe	475
4-Port Serial Interface	299
2-Port Serial/2-Port Parallel Interface	299
4-Port Parallel Interface	229

Printers

Silentype w/Apple II interface **$349**

Epson MX-80 or MX-80 FT CALL

Anadex 9501 w/2K Buffer	1349
C. Itoh Starwriter 25 CPS daisywheel	1449
C. Itoh Starwriter 45 CPS daisywheel	1649
Epson MX-70	CALL
Epson MX-80 & MX-80 F/T	CALL
Epson MX-100	CALL
NEC 8023 Impact Dot Matrix	695
NEC Spinwriters (Latest models)	CALL
Paper Tiger IDS-445G w/graphics	699
Paper Tiger IDS-460G w/graphics	949
Paper Tiger IDS-560G w/graphics	1249
Silentype Printer w/Apple interface	349
Qume Sprint Daisywheels (Latest models)	CALL

ORDER TOLL FREE
800-854-6654

In California and outside continental U.S.
(714) 698-8088
Telex 695-000 Beta CCMO

Send Orders To:

Ordering information Phone orders using VISA, MASTERCARD, AMERICAN EXPRESS, DINER'S CLUB, CARTE BLANCHE, bank wire transfer, cashier's or certified check, money order, or personal check (allow ten days to clear) Unless prepaid with cash, please add 5% for shipping, handling and insurance (minimum 5 00) California residents add 6% sales tax We accept CODs OEM's, Institutions and corporations please send for a written quotation All equipment is subject to price change and availability without notice All equipment is new and complete with manufacturer's warranty (usually 90 days) Showroom prices may differ from mail order prices

consumer computers Mail Order

8314 Parkway Drive
La Mesa, Calif. 92041

Encyclopedia to cooperate in developing the on-line, electronic version of the encyclopedia.

Robert King, chairman of the board of directors at World Book, a subsidiary of Scott & Fetzer Company, said, "We expect to develop significant new services utilizing our existing database, both for the home and business markets. This service will add a new dimension to the printed edition of World Book."

CompuServe subscribers can access the World Book service for the standard fee of $5 per hour weekday evenings, all day weekends and holidays. Weekday daytime access is also available. To use the service, a subscriber needs a personal computer or terminal, a telephone and a modem.

CompuServe also offers the latest news from major newspapers and a national wire service, corporate stock and commodities information, home banking, electronic mail and real-time communications, computer games, family information and computing power for programming activities.

CompuServe is available through a local telephone call in more than 260 US cities. CompuServe is an H&R Block Company.

K-BYTE™ Introduces K-RAZY SHOOT-OUT™ Between Space Commanders And Alien Droids

It's a battle of wits and fast action, this game in ROM form for Atari® 400™ or Atari® 800™ Personal Computer Systems. K-BYTE's K-RAZY SHOOT-OUT is available in a solid-state cartridge (ROM pack), individually tested to assure reliability and customer satisfaction.

K-BYTE's K-RAZY SHOOT-OUT is destined to be a favorite of Atari enthusiasts across the country.

For details on K-RAZY SHOOT-OUT™, contact Chuck Miller at K-BYTE™, Division of Kay Enterprises Co., 1705 Austin, Troy, Michigan 48099. Phone: (313)524-9878.

The Stock Market Goes Home

Marketscan, a new personal online stock market service was formally announced today by Barbara Hyland, Manager of Info Globe.

"We're introducing Marketscan to meet the demands for online information by personal computer users," says Hyland. With Marketscan a user can get daily quotations for all stocks listed on the Toronto, Montreal, Alberta, Vancouver, New York and American Stock Exchanges. The high, low, close, and trading volume are displayed in familiar formats. Marketscan provides a variety of services. A user can analyze a specific stock on one or several exchanges or several stocks at one time, monitor an

entire exchange, or compare a stock's activity within a specific range of dates. Marketscan requires no specialized training.

For more information contact: Barbara Hyland 416-598-5250.

CUE Fall Conference Proceedings Available

Over 1400 people attended the Second Annual Fall Conference of Computer-Using Educators (CUE) in San Jose, California on October 2 and 3, 1981.

The conference led off on Friday, October 2 with 31 hands-on workshops at schools throughout the San Francisco Bay area and 19 field trips to local computer related companies. Over 650 people participated in these activities. Friday evening the keynote speaker was John D'Angelo of Texas Instruments, who discussed the language LOGO and displayed on a projection TV screen his company's implementation of that language.

On Saturday attendees heard Steve Jobs, Chairman of the Board of Apple Computer, Inc. and one of the inventors of that machine, discuss his image of the future of computers in education. For the remainder of the day conference-goers chose from 85 curriculum sessions, 50 commercial presentations, and 45 commercial exhibits.

The Proceedings for this conference will contain contributions from most of the speakers, and will be a valuable reference for those unable to attend, or those who missed particular sessions. To receive a copy of the Proceedings of the 1981 Fall Conference, send a check for $10 (no purchase orders, please) to:

Don McKell
Conference Proceedings
Independence High School
1776 Education Park Drive
San Jose, CA 95133

Proceedings from past conferences are still available in limited quantities: Fall 1980 – $10; Spring 1981 – $10. Send requests to the same address.

Computer-Using Educators is a non-profit California corporation founded in 1978 to promote the educational uses of computers in schools and colleges. It sponsors 4 major conferences per year, issues a bi-monthly newsletter, and maintains a library of non-commercial, teacher-developed educational software for 5 popular microcomputers. Dues are $6 per year, payable to CUE, c/o Don McKell, Independence High School, 1776 Education Park Drive, San Jose, CA 95133.

Commodore Enters Into Memory Disc And Tape Drive Manufacturing

Mr. Irving Gould, Chairman of the Board of Commodore International Limited (NYSE:CBU) announced Commodore's entry into the microcomputer memory disc and tape drive manufacturing business with the introduction of four new multi-megabyte storage devices.

Mr. Gould stated that the "first two storage devices will be 5¼-inch Winchester 6.4 megabyte and 9.6 megabyte magnetic rigid disc drives capable of storing, respectively, up to 6.4 million and 9.6 million bytes of information. These two memory storage devices, which will be built and packaged to Commodore specifications, will be introduced at COMDEX '81. The exact price of these two microcomputer memory storage devices will be announced at that time," noted Mr. Gould, "but they will be priced considerably lower than any com-

ATARI Computers for people.

800™ $699

410 Recorder	$59 00
810 Disc Drive	$444 00
822 Printer	$359 00
825 Printer	$629 00
830 Modem	$159 00
820 Printer	$269 00
850 Interface	$159 00
New DOS 2 System	$21 00
CX70 Light Pen	$64 00
CX30 Paddle	$18 00
CX40 Joy Stick	$18 00
CX853 16K RAM	$89 00
Microtek 16K RAM	$75 00
Microtek 32K RAM	$169 00
One year extended warranty	**$50.00**

ATARI 400

16K. . . . $329
32K. . . . $478
48K. . . . $555

Intec 48K Board. $249

ATARI SOFTWARE

CX404 Word Processor	$119 00
CX404 PILOT	$68 00
CX413 Microsoft Basic	$68 00
CX4101 Invitation To Programing I	$17 00
CX4102 Kingdom	$13 00
CX4103 Statistics	$17 00
CX4104 Mialing List	$17 00
CX4105 Blackjack	$13 00
CX4106 Invitation to Programing 2	$20 00
CX4107 Biorythm	$13 00
CX4108 Hangman	$13 00
CX4109 Graph It	$17 00
CX4110 Touch Typing	$20 00
CX4111 SPACE INVADERS	$17 00
CX4112 States & Capitals	$13 00
CX4114 European Countries & Capitals	$13 00
CX4115 Mortgage & Loan Analysis	$13 00
CX4116 Personal Fitness Program	$59 00
CX4117 Invitation To Programing 3	$20 00
CX4118-20 Conversational Languages (ea.)	$45 00
CX4121 Energy Czar	$13 00
CXL4001 Educational Master	$21 00
CX6001-17 Talk & Teach Series (ea.)	$23 00
CX8106 Bond Analysis	$20 00
CX8107 Stock Analysis	$20 00
CX8101 Stock Charting	$20 00
CXL4002 Basic Computing Language	$46 00
CXL4003 Assembler Editor	$46 00
CXL4004 Basketball	$24 00
CXL4005 Video Easel	$24 00
CXL4006 Super Breakout	$30 00
CXL4007 Music Composer	$45 00
CXL4009 Chess	$30 00
CXL4010 3-D Tic-Tac-Toe	$24 00
CLS4011 STAR RAIDERS	$39 00
CXL4012 MISSLE COMMAND	$32 00
CXL4013 ASTEROIDS	$32 00
CXL4015 TeleLink	$20 00
Visicalc	$149 00
Letter Perfect (Word Processor)	$109 00
Source	$89.00

commodore

CBM 8032 $1149

4016	$799 00
4032	$999 99
8096	$1795 00
CBM4022 Printer	$629 00
Tally 8024	$1699 00
CBM C2N Cassette Drive	$69 00
CBM4040 Dual Disk Drive	$1039 00
CBM8050 Dual Disk Drive	$1349 00
CBM 2031 Single Disc Drive	$525.00
CBM 8300 Letter Quality Printer	$1799 00
CBM 8023P 132 Column Printer	$799

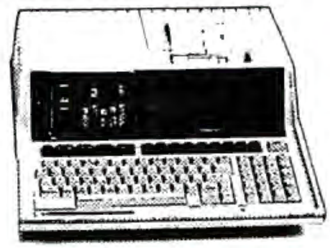
SOFTWARE

WordPro3 Plus	$229 00
WordPro4 Plus	$329 00
Commodore Tax Package	$399 00
Visicalc	$149 00
BPI General Ledger	$329 00
OZZ Information System	$329 00
Dow Jones Portfolio	$129 00
Pascal	$239 00
Legal Time Accounting	$449 00
Word Craft 80	$289 00
Create-A-Base	$249 00
Power	$89 00
Socket 2 Me	$20 00
Jinsam	$Call
MAGIC	$ Call

VIC 20 $259

Vic TV Modual	$19 00
Vic Cassette	$69 00
Vic 6 Pack Program	$44 00
VIC1530 Commodore Datassette	$69 00
VIC1540 Disk Drive	$499 00
VIC1515 VIC Graphic Printer	$399 00
VIC1210 3K Memory Expander	$32 00
VIC1110 8K Memory Expander	$53 00
VIC1011 RS232C Terminal Interface	$43 00
VIC1112 VIC IEEE 488 Interface	$86 00
VIC1211 VIC 20 Super Expander	$53 00
VIC1212 Programmers Aid Cartridge	$45 00
VIC1213 VICMON Machine Language Monitor	$45 00
VIC1901 VIC AVENGERS	$23 00
VIC1904 SUPERSLOT	$23 00
VIC1906 SUPER ALIEN	$19 00
VIC1907 SUPER LANDER	$23 00
VIC1908 DRAW POKER	$23 00
VIC1909 MIDNIGHT DRIVE	$23 00
VT106A Recreation Pack A	$44 00
VT107A Home Calculation Pack A	$44 00
VT164 Programmable Character/Gramegraphics	$12.00
VT232 VICTerm I Terminal Emulator	$9.00

HEWLETT PACKARD



HP•85 $2595

NEW! HP•125	$3295 00
HP•83	$1795 00
HP•85 16K Memory Module	$249 00
5¼" Dual Master Disc Drive	$2129.00
Graphics Plotter (7225B)	$2079 00

Call for HP Software Prices & Information
Call for Calculator prices

Texas Instruments

TI-99/4A $379

PHC 004 TI-99/4 Home Computer	$399 00
PHP 1600 Telephone Coupler	$169 00
PHP 1700 RS-232 Accessories Interface	$169.00
PHP 1800 Disk Drive Controller	$239 00
PHP 1850 Disk Memory Drive	$389 00
PHP 2200 Memory Expansion (32K RAM)	$239 00
PHA 2100 R.F. Modulator	$43 00
PHP 1100 Wired Remote Controllers(Pair)	$31 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	$18.00
PHM 3039 Yahtzee	$22 00
PHM 3017 Terminal Emulator I	$39 00
PHM 3026 Extended Basic	$88 00
PHM 3035 Terminal Emulator II	$45 00

Call for the best prices on
PRINTERS
by Epson, Diablo, TEC and Tally.

DISCS
by Atari and Maxell.

NO RISK • NO DEPOSIT ON PHONE, C.O.D. OR CREDIT CARD ORDERS.

computer mail order

OVER 40 YEARS EXPERIENCE IN SOPHISITICATED ELECTRONICS

east
800-233-8950
501 East Third Street
Williamsport, PA 17701
(717) 327-9575

HOW TO ORDER:
Phone orders invited or send check or money order and receive free shipping in the continental United States. PA residents add 6% sales tax. Add 3% for VISA or MC. Equipment subject to price change and availability without notice.

west
800-648-3351
P.O. Box 6689
State Line, Nevada 89449

parable products now available."

The first storage device to be manufactured by Commodore will be a 5¼-inch Winchester 5 megabyte magnetic rigid disc drive capable of storing up to 5 million bytes of information. This device is scheduled for introduction in April. It too, will be priced considerably below any comparable product now available.

The fourth and final microcomputer memory storage device is based upon Commodore technology and is a state-of-the-art development. It is a ½-inch wide magnetic tape multi-megabyte drive capable of storing up to 43 million bytes of information.

The 5 megabyte and 43 megabyte Commodore manufactured microcomputer memory storage devices are in a 2.2-inch low profile format, meaning they are only 2.2 inches tall, or approximately two-thirds the size of a box of cigarettes.

More Powerful Apple III Features Mass Storage, New Software

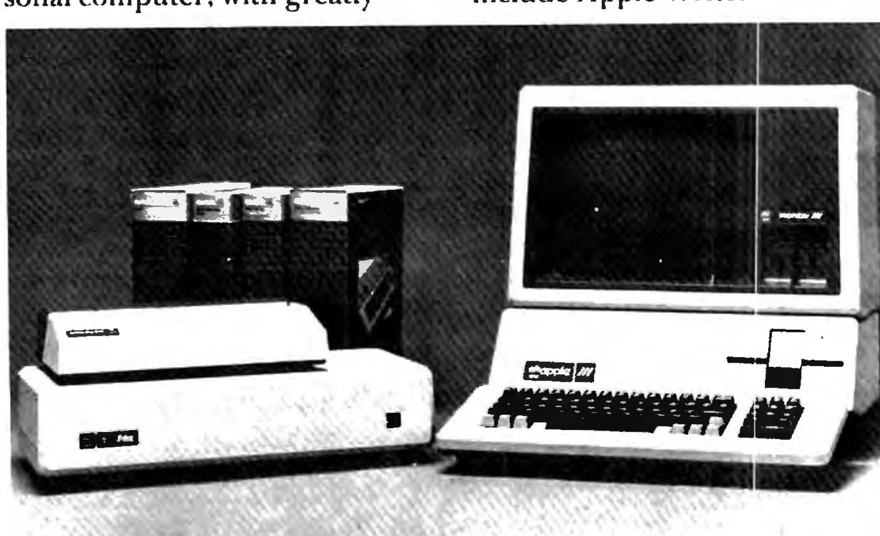
Cupertino, CA — A more powerful version of the Apple III personal computer, with greatly expanded mass-storage capability and professional application programs, will be delivered beginning in mid-December by Apple Computer Inc.

The new Apple III features an improved operating system, more reliable hardware, seven new or enhanced software packages and lower prices than the earlier product. In addition, Apple III supports up to 256K bytes of internal memory.

The expanded storage capability is provided by the new Apple III/ProFile Personal Mass-Storage System. Designed to be integrated into Apple III systems, ProFile is a five-million-byte, Winchester-type, hard-disk system which gives Apple users nearly 35 times the mass-storage capacity of a single floppy disk.

The usefulness of the Apple III is further expanded by a total of seven new or enhanced application programs. One such program, Access III, allows Apple III computers to communicate with large mainframe computers. Apple IIIs for the first time can be used as remote data processing work stations, accessing information from the larger data base and returning completed work to the central computer.

Other new or enhanced programs introduced today include Apple Writer III for



word processing; VisiCalc™ III and Business Graphics III for preparing and analyzing detailed business information; Business BASIC and Pascal III for developing advanced application programs; and Script III for formatting Pascal text into layout-perfect printouts. In addition, an updated version of Mail List Manager will soon be available for maintaining mailing lists and printing labels using an Apple III/Profile system.

Numeric Keypad With Calculator And VisiCalc™ Commands For Apple II™

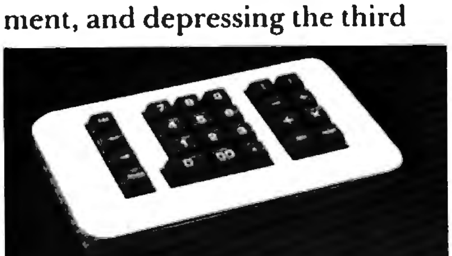
The Keyboard Company's Numeric Keypad lets hard-working professionals enter numbers, carry out arithmetic operations and input VisiCalc™ commands quickly and easily.

The keypad, clearly coordinated with the Apple computer in both color and design, increases operator efficiency by concentrating essential keys in a familiar format, and allowing the operator to place the keypad in a comfortable position.

In addition to the standard keypad with double zero and decimal point, The Keyboard Company's product has a full set of operator keys, complete with parenthesis, print, return and four basic arithmetic functions.

The VisiCalc section of the keypad uses three keys to control cursor movement. Two keys control the directional movement, and depressing the third key will change the cursor horizontal movement to vertical. Holding down either directional movement key initiates the auto-repeat mode, which moves the cursor across the screen until the key is released. A fourth key deletes entries.

The keypad, complete with interface board, cord and directions, is available in Apple dealerships nationwide for $149.95.



PET TINY BASIC COMPILER

Abacus Software's TINY BASIC COMPILER (TBC) supports a floating point subset of the PET BASIC programming language. The compiler reads your program and writes out a file containing 6502 object code that you then load and execute.

The TBC supports all floating point arithmetic and functions that are available in the full PET BASIC language. You can write, test and debug your program using the built-in interpreter before using the TBC to compile it.

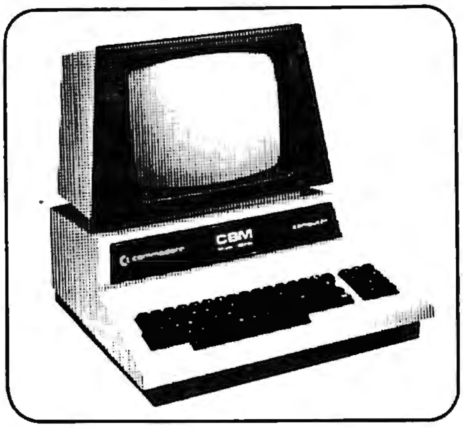
The TBC package will run on all 40-column model PET/CBMs with a minimum of 8K of memory. If you have at least 16K of memory, then there is also a version (included in package) which will give you a full assembly listing of the compiled code.

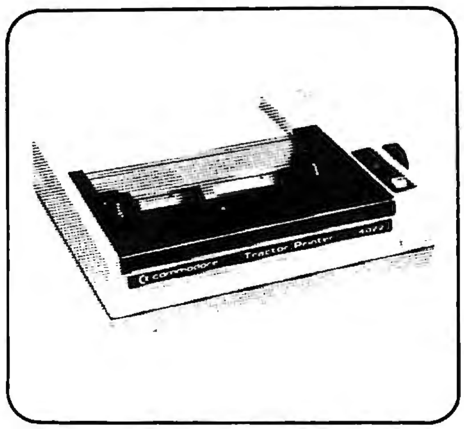
Price is $25.00 on cassette or diskette in US and Canada, and $30.00 foreign. For further information contact Abacus Software, P.O. Box 7211, Grand Rapids, MI 49510 (616)241-5510.

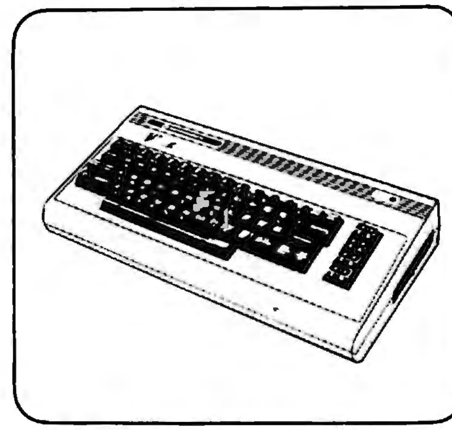
New Product releases are selected from submissions for reasons of timeliness, available space, and general interest to our readers. We regret that we are unable to select all new product submissions for publication. Readers should be aware that we present here some edited version of material submitted by vendors and are unable to vouch for its accuracy at time of publication.

commodore SPECTACULAR

8032-32K 80 COL CRT	REG $1495	$1095
64K ADD-ON MEMORY	REG $500	$395
9000 134K SUPER PET	REG $1995	$1795
4032 32K 40 COL CRT	REG $1295	$995
4016 16K 40 COL CRT	REG $995	$795
8050-DUAL DISK 950K	REG $1795	$1395
4040-DUAL DISK 343K	REG $1295	$995
2031-SINGLE DISK 170K	REG $695	$555
C2N-CASSETTE DRIVE	REG $75	$65
4022-80 COL PRINTER	REG $795	$649
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	$269
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

SOFTWARE

	LIST	YOU PAY
OZZ- The Information Wizard	$395	$299
Wordcraft 80	$395	$299
IRMA-Info Retrieval & Mgmt Aid	$495	$399
Dow Jones Portfolio Mgmt.	$149	$119
Pascal Development Pkg.	$295	$229
EBS-Receivables, Inventory	$750	$579
BPI-General Ledger	$395	$299
Word Pro 3-40 Column	$250	$179
Word Pro 4-80 Column	$375	$269
Word Pro 4 Plus	$450	$329

PHILADELPHIA COMPUTER DISCOUNT ©

P.O. Box 170 St. Davids, PA 19087 (215)687-8540

MAIL ORDER PHONE 1-800-345-8112

IN PA CALL 1-800-345-1289

PREPAID ORDERS SHIPPED FREE
VISA & MASTERCARD ADD 3%
COD - UPS
H1181A PA RESIDENTS ADD 6% SALES TAX

BRING IN THE NEW YEAR!

LOOK CLOSELY AT THESE SUPER SAVINGS!!

SAVINGS!

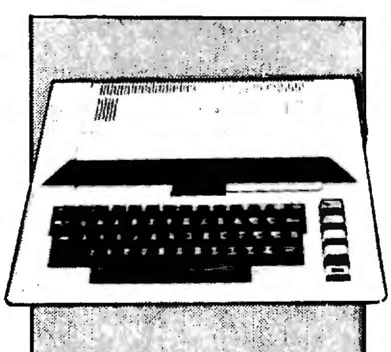
AMAZING SPECIALS!

OMEGA SALES CO.

• We Accept C.O.D.'s • Stock Shipments Same Day or Next • No Surcharge For Credit Cards • All Equipment Factory Fresh w/MFT Warranty • We Carry the Complete Line of Personal Software

NEC Color Monitor
JC 1201 $319

EPSON MX-80 $ 469



ATARI 800 16K $ 749

INTERTEC SUPERBRAIN 64K RAM	**$2799**
QD SUPERBRAIN	$2999
NEC 5510 SPINWRITER (7710)	**$2345**
NEC 5520 SPINWRITER (7720)	$2695
NEC 5530 SPINWRITER (7730)	$2345
NEC 12" MONITOR	$ 189
NEC COLOR 12" MONITOR	$ 339
NEC PC 8023 Printer 100 CPS Tractor & Friction	$ 639
OKIDATA MICROLINE-80	**$ 379**
OKIDATA MICROLINE-82A	$ 529
OKIDATA MICROLINE-83A	$ 749
DIABLO 630	$1995
APPLE II PLUS 48K	**$1139**
APPLE DISK w/3.3 DOS Controller	$ 525
APPLE DISK w/o Controller	$ 449
EPSON MX-80	$ 469
Interfaces: IEEE $55, TRS-80 CABLE $35, APPLE INTERFACE & CABLE $90, RS-232 $70	
HAZELTINE 1420	$ 799
NORTHSTAR HORIZON II 32K QD	$2925
ANADEX DP-9500/9501	$1199
TELEVIDEO 912C	**$ 699**
TELEVIDEO 920C	$ 729
TELEVIDEO 950	$ 929

WE CARRY THE COMPLETE LINE OF ATARI SOFTWARE, PERIPHERALS AND ACCESSORIES.

EAST COAST
1-800-556-7586

WEST COAST
1-800-235-3581

PRICES ARE SUBJECT TO CHANGE W/O NOTICE.

NEC Green 12 Inch Monitor
JB 1201 $179

NEC Friction Tractor Printer
100 CPS (Graphics, Bi-directional) $635

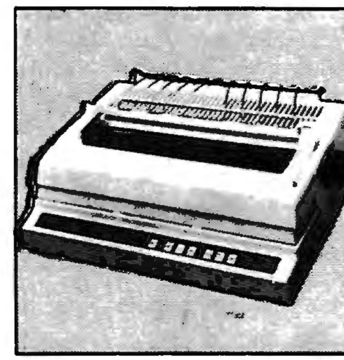
CBM 8032 COMPUTER	$1149
CBM 8050 DISK DRIVE	$1349
CBM 4032 COMPUTER	$1029
CBM 4040 DISK DRIVE	$1029
CBM 4022	$ 649
CBM VIC-20	$ 269
LEEDEX/AMDEK 100	**$ 139**
LEEDEX/AMDEK 100G	$ 169
LEEDEX/AMDEK COLOR-1 13" Color Monitor	$ 329
MICROTEK 16K RAM BOARD for Atari	$ 79
MICROTEK 32K	$ 149
ATARI 800 16K	**$ 749**
ATARI 400 16K	$ 349
ATARI 810 DISK DRIVE	$ 449
ATARI 820 40 Column Printer	$ 299
ATARI 822 40 Column Thermal Printer	$ 349
ATARI 825 80 Column Printer	$ 599

OMEGA SALES CO.

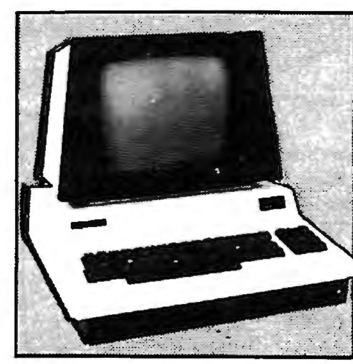
EAST COAST
1-800-556-7586
OMEGA SALES CO.
12 Meeting St.
Cumberland, RI 02864
1-401-722-1027

WEST COAST
1-800-235-3581
OMEGA SALES CO.
3533 Old Conejo Rd. #102
Newbury Park, CA 91320
1-805-499-3678
CA. TOLL FREE 1-800-322-1873

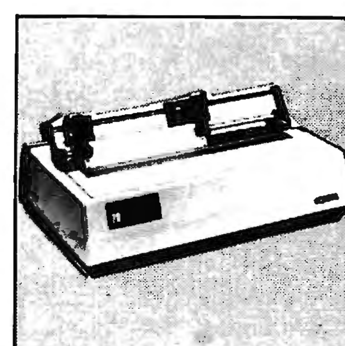
OMEGA SALES CO.



DIABLO 630 $1995



CBM 8032 COMPUTER $1149



OKIDATA MICROLINE-83A $ 749

ANADEX DP-9500/9501 $1199

Advertisers Index

AB Computers ... 40,41,161
AG Associates ... 52
Aardvark Technical Services, Ltd. ... 137
Abacus Software ... 57
Actek ... 29
Adventure International ... 15
Alpha Logic Business Systems, Inc. ... 91
Alternate Reality Software ... 119
Andromeda Incorporated ... 85
Apparat, Inc. ... 87
Artworx Software Co. ... 23
Atari, Inc ... 6,7
Automated Simulations, Inc. ... 25
Basic Software Service ... 151
Batteries Included ... 153
Beta Computer Devices ... 57
The Bit Bucket ... 126
R. J. Brachman Assoc., Inc. ... 57
C-Mart ... 193
CE Software ... 107
CFI ... 151
CGRS Microtech ... 67
CMS Software Systems ... 2,3
Canadian Micro Distributors, Ltd. ... 61,63,65
Cascade Computerware ... 149
Color Computer Concepts ... 111
Commodore Computer Systems ... BC
Computer House ... 115
Computer Mail Order ... 201
Computer Mat ... 175
The Computerist's Directory ... 27
Connecticut microComputer, Inc. ... 157
Consumer Computers ... 197
Creative Software ... 27
Crystal Computer ... 190,191
Cursor, The Code Works ... 24
Cyberia ... 51
Data Resource Corp. ... 139
Don't Ask ... 127
Dr. Daley's Software ... 33,142
Dynacomp ... 74,75
ETC Corp ... 187
Eastern House Software ... 47
Eclectic Systems Corporation ... 101,199
Elcomp Publishing, Inc. ... 202
Electronic Specialists, Inc. ... 47
Esplanade Enterprises ... 111
Execom Corporation ... 159
FSS ... 157
Falk-Baker Associates ... 73
The Great Western Software Company ... 159
HW Electronics ... 125
Hayes Microcomputer Products, Inc. ... 11,13,126
High Country Microsystems ... 119
Horizon Simulations ... 188
Human Engineered Software ... 185
Huntington Computing ... 31
Impact Computer Sytems ... 53
Inhome Software ... 124
Intec Peripherals Corp. ... 107
Interlink, Inc. ... 29
Jini Micro Systems ... 16
Krell Software Corporation ... 77
LJK Enterprises ... 49
Leading Edge Products, Inc. ... IBC
LemData Products ... 161
Lo-Ball Computers ... 192
Lyco Computers ... 203
Macrotronics, Inc. ... 204
MED Systems Software ... 121
Meta Software Engineering ... 101
Micro Business World, Inc. ... 195
Micro-Ed, Inc. ... 82
Micro Spec Ltd. ... 179
Micro Technical Products ... 69
Micro Technology Unlimited ... 55
Micrograms, Inc. ... 81
Microsoft Consumer Products ... 4
MicroComputer Industries ... 19
Microperipheral Corp. ... 131
Microtek, Inc. ... 21
Mosaic Electronics ... 102
Mountain Computer, Inc. ... IFC
Muse Software ... 88
New England Electronics Company ... 34,35
Netronics ... 200
Olympic Sales Co. ... 58
Omega Sales Co. ... 206,207
On-Line Systems ... 12,129
Oppenheimer Software ... 173
Optimal Technology, Inc. ... 67
Optimized Data Systems ... 144
Pacific Exchanges ... 47,52,58,72
Percom Data Company, Inc. ... 71
PETTED Microsystems ... 153
Philadelphia Computer Discount ... 205
Poquette's ... 33
Pretzelland Software ... 141
Professional Software ... 1,9
Program Design, Inc. ... 126,131
The Program Store ... 194
The Programmer's Institute ... 33
Protronics ... 67
Quality Software ... 121
Quantum Data Inc. ... 204
Questar International Inc. ... 147
Santa Cruz Educational Software ... 105
Sebree's Computing ... 126
Skyles Electric Works ... 145,198
The Software Connection ... 188
Software by Sasso ... 173
Software Street ... 131
Spectrum Computers ... 111
Starbound Software ... 117
Street Electronics Corp. ... 99
Syncro, Inc. ... 115
Synergistic Software ... 107
T'Aide Software Company ... 12
T.H.E.S.I.S. ... 109
TIS Inc. ... 17
TNW Corporation ... 29
Unicomm, Inc. ... 196
United Microware Industries, Inc. ... 177
Virginia Micro Systems ... 149
Voicetek ... 20

The 6502 Resource Magazine

COMPUTE! 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...**

☐ Please enter my 1 year (12 issue) subscription to **COMPUTE!**

☐ New subscription ☐ Renewal subscription

☐ $20.00 US subscription
☐ $25.00 Canada and international surface mail. Payment in US funds drawn on US bank or money order
☐ Bill me (US only)

Name, Address: ____________________

Charge my: ☐ Visa ☐ MC

Number____________ Expires_____/_____/

1 2 3 4 5 6

The 6502 Resource Magazine

COMPUTE! PET•ATARI•APPLE OSI•KIM•SYM•AIM

Foreign Readers

New subscription rates for **COMPUTE!** readers outside of the US.

☐ $25 Anywhere/Surface Mail (2-4 months delivery)
☐ $38 Europe/Air Delivery (7-10 days)
☐ $48 Middle East, North Africa, Central America/Air Mail
☐ $88 South America, South Africa, Far East, Australia/Air Mail

Name ____________________

Address ____________________

☐ Check here if renewal subscription.

Payment must accompany this card.
Payment in US funds drawn on a US bank, International Money Order or charge card. VISA ☐ MasterCard ☐.
Account No.____________, Expires_____/_____/.

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	$15
☐ All About 278 Small Business Computers	$15
☐ All About 246 Minicomputers	$15
☐ All About 149 Microcomputers	$15
☐ All About 114 Microprocessors	$15
☐ User Ratings of Proprietary Software Packages	$25
☐ A Buyer's Guide to Data Base Management Systems	$15
☐ All About 136 User-Programmable Terminals	$15
☐ All About 210 Alphanumeric Display Terminals	$15
☐ All About Data Communications Facilities	$15
☐ Management Guidelines for Office Automation	$15
☐ All About 90 Word Processing Software Packages	$15
☐ All About 125 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

datapro DATAPRO RESEARCH CORPORATION 1805 Underwood Blvd., Delran, NJ 08075

All About Personal Computers — Only $29

Newly updated for 1981 — the industry's first unbiased objective report on the 18 most popular personal computers.

Datapro's latest report — **All About Personal Computers** — provides valuable information that will help you compare and evaluate 18 of the most popular personal computers including five new systems — Bell & Howell Microcomputer System, Commodore VIC 20, Hewlett-Packard 83, Radio Shack's TRS-80 Model III and TRS-80 Color Computer. Over 68 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 18 most popular in **All About Personal Computers**. Available for only $29.

18 MOST POPULAR PERSONAL COMPUTERS

Apple II & Apple II Plus
Atari 400 & 800
Bell & Howell Microcomputer System
Commodore PET
Commodore VIC 20
Exidy Sorcerer
Heath H8
Heath All-In-One Computer
Hewlett-Packard 83
Hewlett-Packard 85
Mattel Intellivision
North Star Horizon
Ohio Scientific Challenger I
Ohio Scientific Challenger II
Radio Shack TRS-80 Model I
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

FREE with this report: Datapro's latest catalog of feature reports on EDP and office systems selected from our monthly updated information services

Place
Stamp
Here

COMPUTE! Magazine
515 Abbott Drive
Broomall, PA 19008
USA

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 236 BROOMALL, PA

POSTAGE WILL BE PAID BY ADDRESSEE

COMPUTE! Magazine
515 Abbott Drive
Broomall, PA 19008

COMPUTE! 1/82

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY CARD
FIRST CLASS / PERMIT NO 178 / DELRAN, NJ

POSTAGE WILL BE PAID BY ADDRESSEE

datapro

DATAPRO RESEARCH CORPORATION
1805 Underwood Boulevard
Delran, New Jersey 08075

COMPUTE! 1 82

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY CARD
FIRST CLASS / PERMIT NO 178 / DELRAN, NJ

POSTAGE WILL BE PAID BY ADDRESSEE

datapro

DATAPRO RESEARCH CORPORATION
1805 Underwood Boulevard
Delran, New Jersey 08075

REMEMBER.

Elephant™ floppies.
They're guaranteed to meet or beat every industry standard for quality. They come standard with reinforced hub rings at no extra cost. They come in every popular 5¼" model, in both hard and soft sector. And they sell at some of the lowest prices in the business.
Elephant Flexible Disks.
They're heavy duty. They work for peanuts. They never forget. Get yourself a trunkful.

HEAVY DUTY DISKS.

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.