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

$2.50
December, 1981
Issue 19
Vol. 3, No. 12
63379

COMPUTE!

The Journal For Progressive Computing™

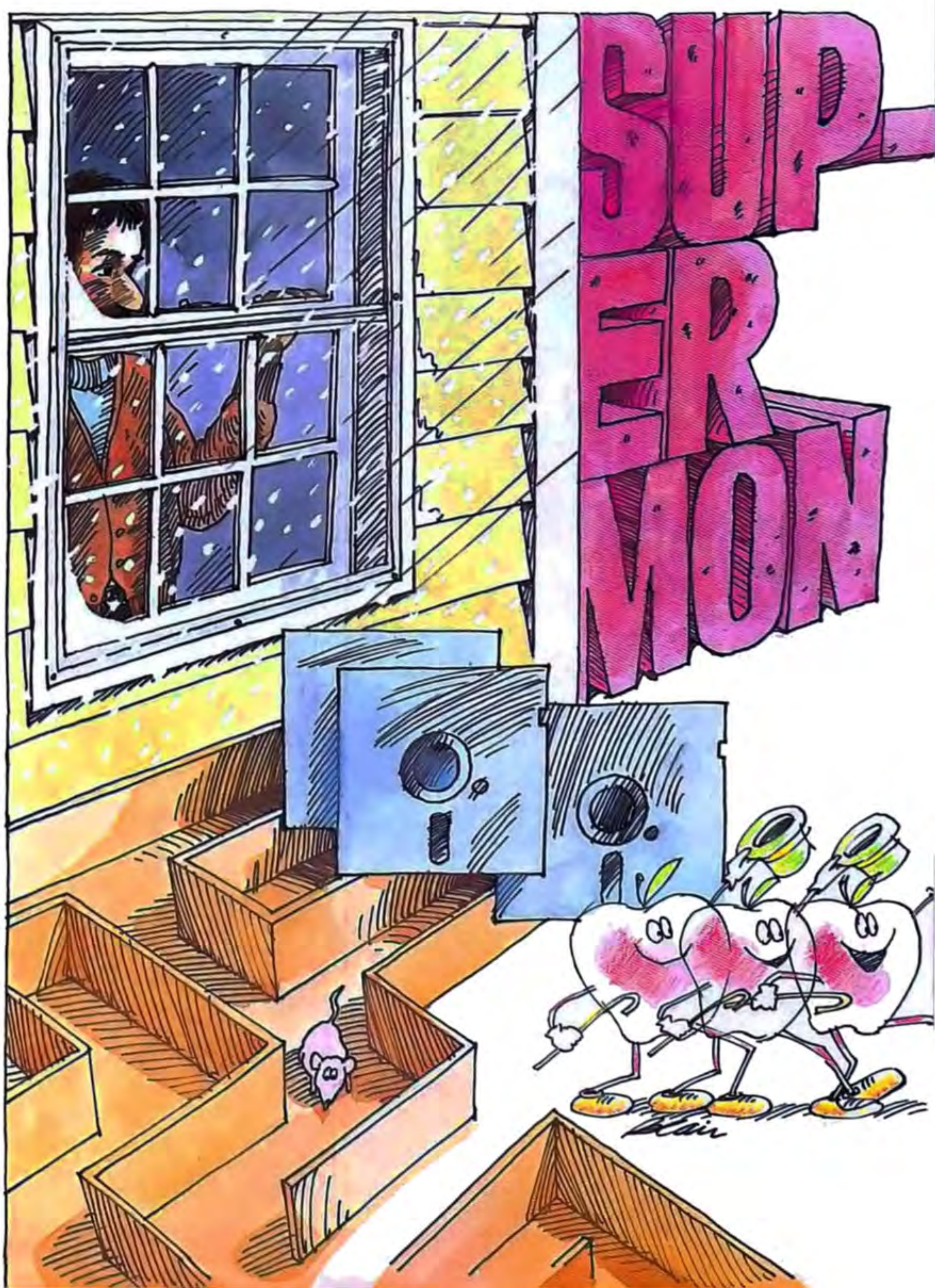
Window Analysis:
A Home Heating/Cooling Application Program

Supermon
Machine Language Monitor For CBM/PET Computers

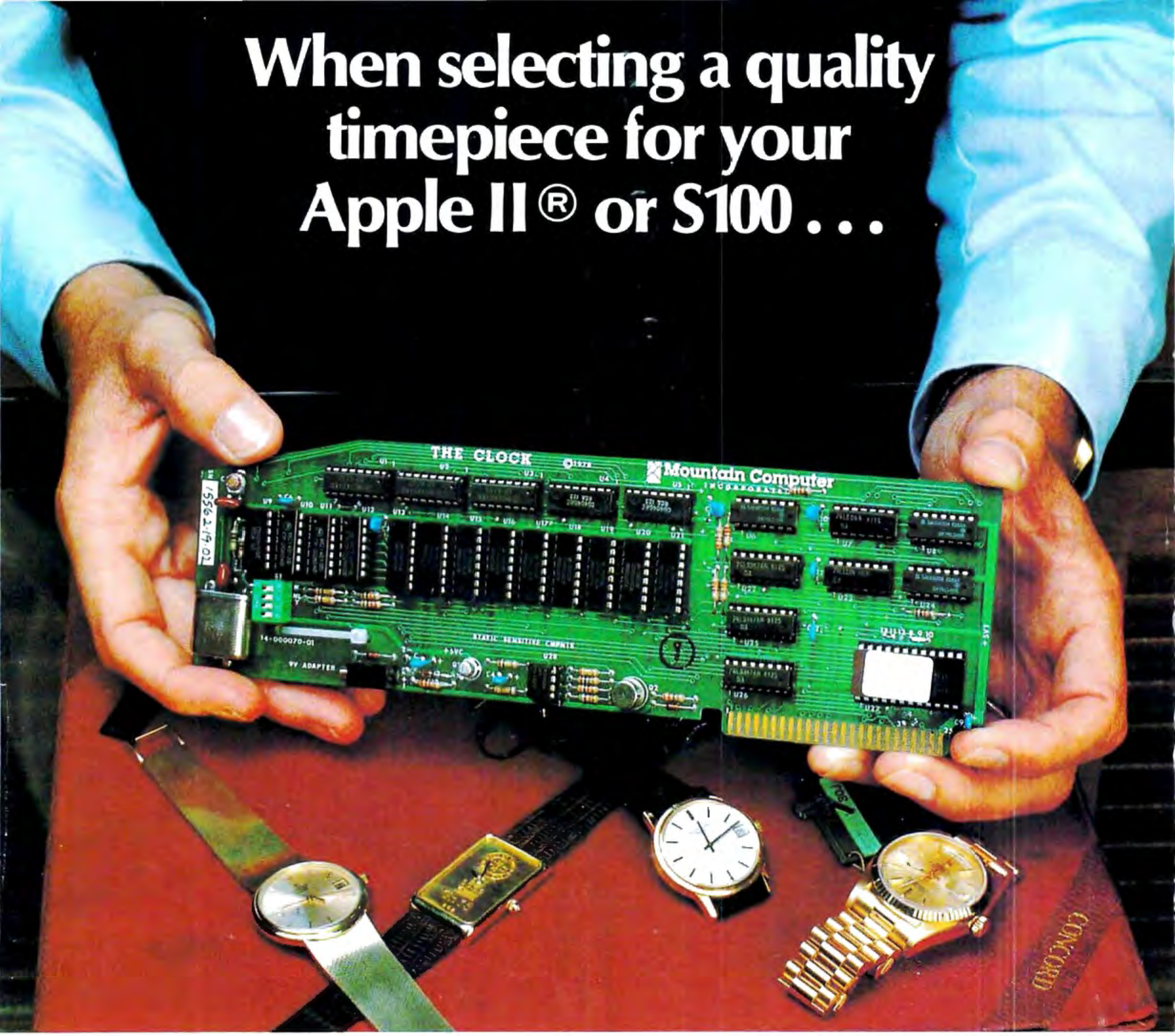
INSIGHT: Atari
Part II Of The Mysteries Of I/O

Animating Applesoft Graphics

Maze Maker:
A Maze Generator For Game Applications



0 74470 63379 12



Select The Clock™ or 100,000 Day Clock™ from Mountain Computer.

100,000 DAY CLOCK

- Utilizes 16 I/O ports of a Z-80 or 8080 system.
- Crystal controlled circuit for accurate time: ± .001%.
- Time in 100mS increments for periods up to 100,000 days (273 years).
- Software selectable interrupts for intervals of 100mS, 1mS, . . . 1 sec., 10 sec, 1 min, 1 day, 10 days, . . . etc.

THE CLOCK

- Crystal controlled circuit for accurate time: ± .001%.
- Software and hardware selectable interrupts.
- The standard for software compatibility.

See your Apple® Dealer or contact us for more information.

Mountain Computer
INCORPORATED

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

Turn your Apple into the world's most versatile personal computer.

The SoftCard™ Solution. SoftCard turns your Apple into two computers. A Z-80 and a 6502. By adding a Z-80 microprocessor and CP/M to your Apple, SoftCard turns your Apple into a CP/M based machine. That means you can access the single largest body of microcomputer software in existence. Two computers in one. And, the advantages of both.

Plug and go. The SoftCard system starts with a Z-80 based circuit card. Just plug it into any slot (except 0) of your Apple. No modifications required. SoftCard supports most of your Apple peripherals, and, in 6502-mode, your Apple is still your Apple.

CP/M for your Apple. You get CP/M on disk with the SoftCard package. It's a powerful and simple-to-use operating system. It supports more software than any other microcomputer operating system. And that's the key to the versatility of the SoftCard/Apple.

BASIC included. A powerful tool, BASIC-80 is included in the SoftCard package. Running under CP/M, ANSI Standard BASIC-80 is the most powerful microcomputer BASIC available. It includes extensive disk I/O statements, error trapping, integer variables, 16-digit precision, extensive EDIT commands and string functions, high and low-res Apple graphics, PRINT USING, CHAIN and COMMON, plus many additional commands. And, it's a BASIC you can compile with Microsoft's BASIC Compiler.

More languages. With SoftCard and CP/M, you can add Microsoft's ANSI Standard COBOL, and FORTRAN, or Basic Compiler and Assembly Language Development System. All, more powerful tools for your Apple.

Seeing is believing. See the SoftCard in operation at your Microsoft or Apple dealer. We think you'll agree that the SoftCard turns your Apple into the world's most versatile personal computer.

Complete information? It's at your dealer's now. Or, we'll send it to you and include a dealer list. Write us. Call us.

MICROSOFT CONSUMER PRODUCTS

Microsoft Consumer Products, 400 108th Ave. N.E., Bellevue, WA 98004. (206) 454-1315

Table of Contents

December, 1981, Vol. 3, No. 12

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 Editor's notes...

Robert Lock, Publisher/Editor

The Price Wars

We've recently been hearing increasingly interesting rumors regarding the problems/ solutions of "conflict" between mail order houses and "store-front" dealers. We've discussed the matter before on these pages and, without taking sides, the situation is simply this.

Retail outlets, with higher operating expenses, potentially greater overhead, customer service and training in varying degrees, and so on, tend to lean mainly in the direction of "Manufacturer's Suggested List Price."

Many mail order houses, on the other hand, with perhaps less expectation of personnel intensive support, training, and so on (plus potentially far greater volume) tend to discount.

This has been the nature of retailing in the industry for a long time with the lines of argument frequently becoming rather heated. Some stores, for example, refuse to carry magazines, feeling that there's truth, we suppose, in the age old adage: see no evil, buy no evil. For our part, we understand perfectly why a retail store would be completely frustrated by a customer who makes three or four visits to explore hardware and then buys direct, at lower cost, from a mail order house.

The Competition Increases (Decreases?)

Now we hear that Apple is seeking to squelch some of their mail order discounters, a move we would venture is calculated to, among other things, increase dealer loyalty in the face of the recent IBM entry.

Now the interesting part of this analysis is that Apple sells direct to dealers, while Atari has set up regional distributors. In the past, the discount mail order business could be classified as a fair fight, in the sense that dealers were venturing volume against home town support, etc., etc. Pricing, to them, was roughly similar.

Now, however, we're hearing that a "third" level is being added to the fray: essentially distributor supported and sponsored mail-order houses who, because they have better initial profit margins, can have the best of both worlds. They sell to dealers, both store-front and mail order, plus sell to the end-user through their own mail-order house. Naturally enough, their discount pricing can be more than competitive.

Where Does It Go From Here?

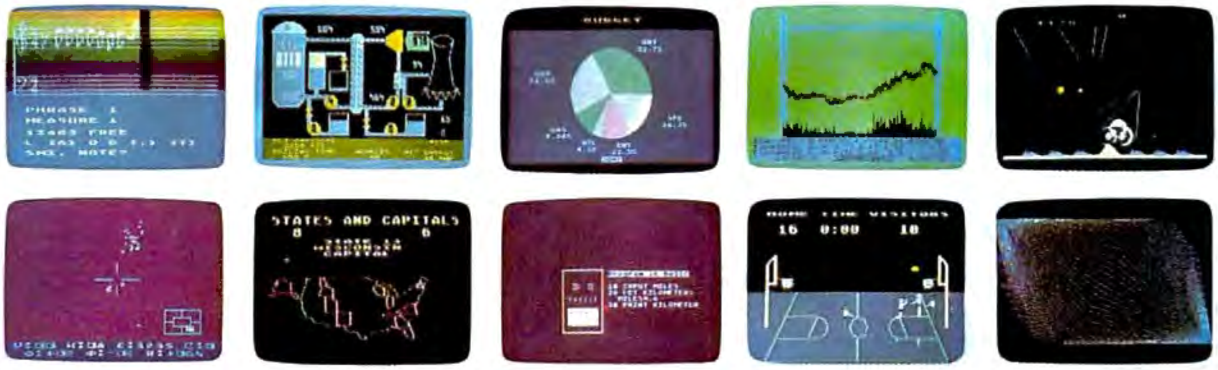
We would guess that time will bring changes in the basic methods of distribution, with the needs and demands of the consumer for fair pricing and support balancing with the needs and demands of the dealers for reasonable profit margins and competitive business practices. We'd like to hear from end users, dealers, and others involved in the distribution process for your comments and suggestions.

Home Applications

Can you really use this machine for some practical applications at home? We certainly think so, and our recent requests for such articles has been well received by you reader/users. This issue presents "Window Analysis." In two versions, one for Microsoft BASIC and one for Atari BASIC, you'll be able to explore how efficiently your house is using solar power to cut heating/cooling costs. Using several vari-

THE GRAPHIC DIFFERENCE

BETWEEN ATARI® COMPUTERS AND ALL OTHERS.



3.7 million reasons why the ATARI Personal Computer is something to see.
The display screen used with our computers is composed of 192 horizontal lines, each containing 320 dots. Delivering color and luminosity instructions to each dot for a second requires 3.7 million cycles... a lot of work for the normal 6502 processor.

That's why the ATARI computer has equipped its 6502 with its own electronic assistant. It's called ANTIC, and it handles all the display work, leaving the 6502 free to handle the rest. What this means to you is uncompromisingly spectacular display capabilities without loss of computer power needed to carry out the demands of your program.

That's a quality you just don't find in ordinary personal computers. And it's one of the reasons some computer experts say that ATARI computers are so far ahead of their time.

There's more... which is what you'd expect from ATARI.
Language. The ATARI Personal Computer uses several programming languages to give the user maximum control of its extraordinary capabilities. PILOT, Microsoft BASIC,* and ATARI BASIC are understood and spoken by the ATARI computer. You'll also find our Assembler Editor cartridge indispensable for machine language programming.

Sound. An ATARI computer has four sound generators, or voices, activated by a separate microchip. This leaves the principal microprocessor chips free to perform other tasks. And you can take full advantage of this capability which is designed for easy programming.

Change. ATARI Personal Computers have been designed to make change and expansion easy. The ATARI computer has a modular operating system* that can be easily replaced as new technology develops. If you need it, memory expansion requires no more than inserting additional RAM modules.* And the ATARI ROM cartridge system also makes it easy to change languages. In short, your ATARI computer won't be obsoleted by future developments... *because it already incorporates the future.*

Sharing. To learn more about the amazing capabilities of ATARI computers, visit your local computer store for a demonstration. Or send for our Technical User's Notes, intended for the serious programmer. They are only $27 and contain a lot more information about our computers' special capabilities than most companies could tell. See your ATARI dealer, or send $30 ($27 plus $3 postage and handling), payable to ATARI, to Technical User's Notes, c/o ATARI Customer Service, 1340 Bordeaux Avenue, Sunnyvale, CA. 94086.

*ATARI 800™ computer only.

ATARI®

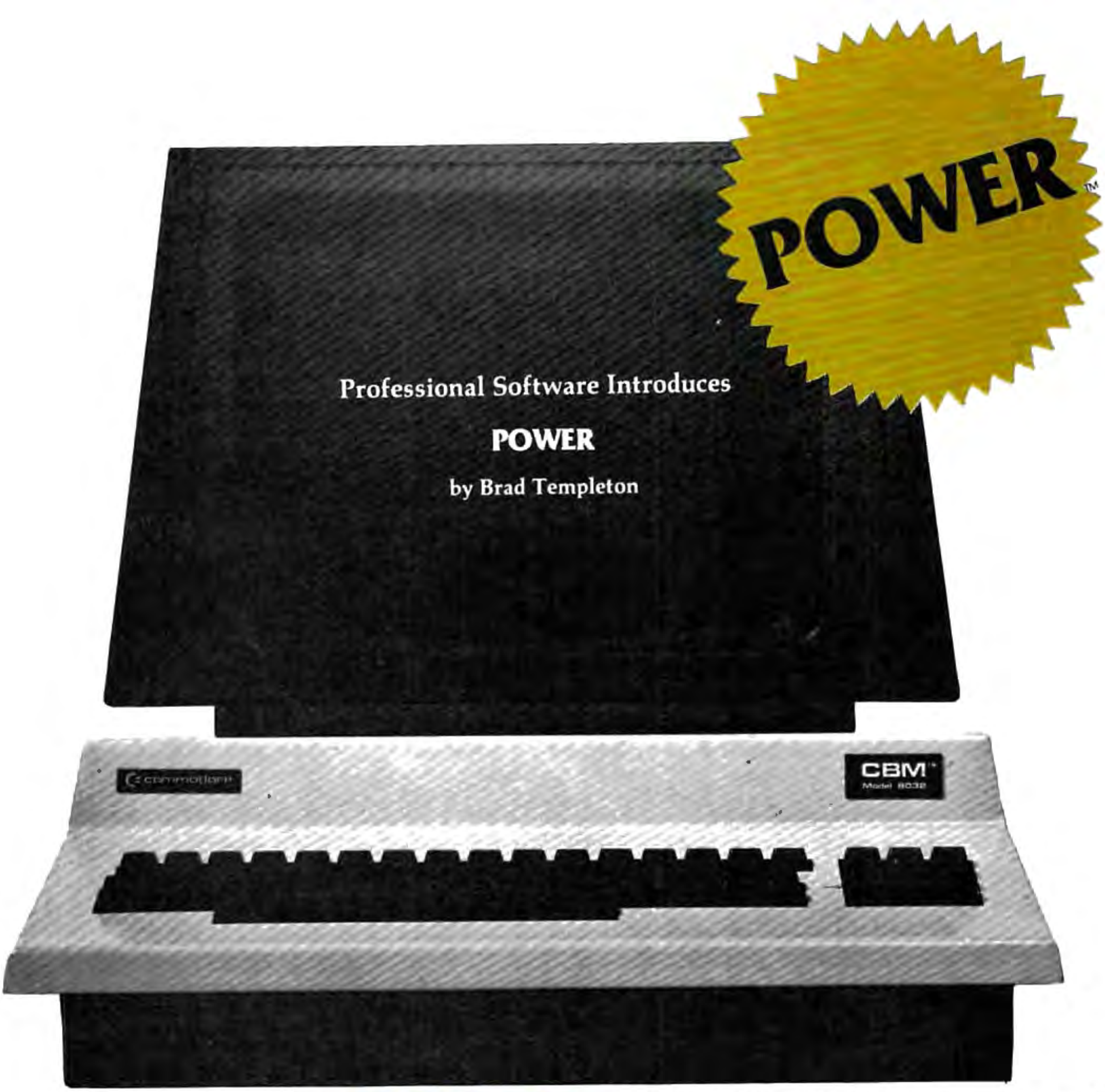
Computers for people.™

ables, you can decide how to optimize the benefits of free sunlight by making computer-assisted changes; shades, awnings, shrubs, etc.

If you've developed an interesting, useful home or small business application such as this, please write it up and send it in. Where possible, we'll "homogenize" it in-house (we developed the Atari BASIC version here), and present it for all readers to enjoy. If you take an application in **COMPUTE!**, such as "Maze Maker" in this issue, and develop an interesting game around it, we'd like to see that too.

Trade Shows And More Trade Shows

We mentioned in last issue's editorial that more regional/local dealers and distributors were in evidence at the last Boston Show, and fewer manufacturers. This pattern repeated itself at the recent show in San Francisco. With our industry generating new trade shows faster than new computers these days, I suspect we'll see this quickly become the norm. Manufacturers and principals will attend a few shows each year; local distributors and dealers will handle the rest. ©



ADD POWER™ TO YOUR COMMODORE COMPUTER

$89.95

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

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

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

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

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

Robert C. Lock, Publisher/Editor
Kathleen Martinek, Managing Editor
Richard Mansfield, Assistant Editor
Alice S. Wolfe, Publisher's Assistant
Charles Brannon, Editorial Assistant

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 Thornburg, P.O. Box 1317, Los Altos, CA 94022
Bill Wilkinson, Optimized Systems Software, 10379-C Lansdale, Ave. Cupertino, CA 95014
Gene Zumchak, 1700 Niagara St., Buffalo, NY 14207

Georgia Papadopoulos, Art Director/ Production Manager
Terry Cash, Typesetting/Production Assistant
Dai Rees, Production Assistant
Harry Blair, Director, Advertising and Promotion
Joretta Klepfer, Manager, North American Retail Sales

Bonnie Valentino, Circulation Assistant
Sonja Whitesell, Office Assistant

Subscription Information

(12 Issue Year):
COMPUTE! Circulation Dept.
P.O. Box 5406
Greensboro, NC 27403 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

Home and Educational 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 January Issue of COMPUTE!:

Create Your Own Commands To Customize Atari BASIC

Apple, Atari, and PET: A Real Estate Investment Analysis Program

Assembly Language On The SuperPET

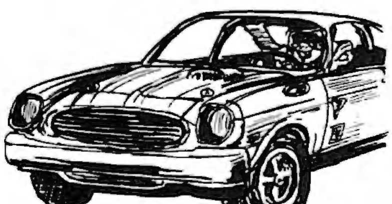
And Much, Much More...

THE PROGRAM STORE

CALL TOLL FREE 800 424-2738

By John Harris from On-Line
If you like the PAC-MAN arcade game, you'll love JAW BREAKER! Guide your chompers through a candystore maze, eating "wife-savers" as you go. You are pursued by variously colored "smilies," determined to knock your teeth out. If you eat a jawbreaker, the "smilies" turn to blue "frownies," and you try to eat them! Sound familiar? It's one of the best uses of Atari graphics and sound we've seen yet -- you'll love it!

16K disk...$29.95



DODGE RACER

From Synapse Software
You guide your racecar through six lanes of speedway with your joystick, scoring points as you pass over the white dots in each lane. Chasing you in the opposite direction is the phantom jam car with one purpose -- to meet you head-on in a fiery crash!

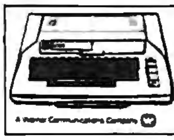
Full of sound, color, and excitement, DODGE RACER can be played with one to four competitors and has 12 play options.

16K tape...$22.50 24K disk...$22.50

CONFLICT 2500

From Avalon Hill
While checking your orbiting minefields, one of your Hyperfighters detects the presence of enemy Planet Pulverizers. Get set for a star battle in the 26th century. This new space game contains all the elements to hold your fascination: graphics, strategy, and challenge. Using the unique "Variable Scenario System" you set the size of your fleet and the level of complexity. Play one-on-one with the computer or with up to 10 players.

Tape for 16K Apple, 32K Atari,
TRS-80 (16K/I, 32K/III)...$14.95



Programs for ATARI 400/800
Unless Otherwise Noted

LUNAR LANDER

By Wall, Moncrief & Jessee from A.I.
Grab your joystick and sweat out the touch-down of your LEM. This fascinating and challenging game makes good use of Atari's player/missile graphics, color, and four-voice sound.

Choose from four lunar landscapes, with many choices for landing sites. Select a more difficult site and get more points -- if you can land successfully. You have complete control of your LEM via main engines and small side thrusters, and five successful landings are heralded with a flag raising ceremony. Great fun!

24K tape (disk compatible)...$14.95

POKER SOLITAIRE

By Jerry White from Artsci
You are dealt 25 cards which are then turned up one at a time. As each card is faced, you must decide where to place it on a 5 X 5 grid. The final results are scored from each of the 10 poker hands formed. From one to four players can compete and for added fun, a time penalty can be invoked. It's fun!

16K tape...$14.95

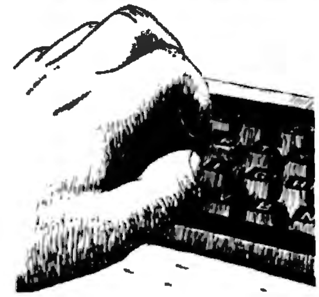
3-D SUPERGRAPHICS

By P. Lutus from United Software
A program development system that allows you to project 3-dimensional images of color, high resolution images and manipulate them at will.

Images are drawn at a speed that makes animation not only possible but almost easy! Routines developed with this system can be incorporated into your BASIC programs. Includes manual and demonstration programs.

40K disk...$39.95

TYPING TUTOR



By Ainsworth & Baker from Microsoft
Speed up your programming and word processing with this excellent touch-typing instructional program. First proper finger positioning is taught, the program adding new characters as you progress. Then practice paragraphs are evaluated for accuracy and speed. Continuously adjusting to your increasing skill, TYPING TUTOR senses any keys you're not sure of and points them out.

16K tape...$19.95

TOUCH TYPING

From Atari
Three program modes plus separate data tape guide you from "hunt-n-peck" beginner through expert typing skills. On-screen graphics aide finger placement and drills progress from character patterns to sentences to full paragraphs. Start at your own level; learn at your own pace.

16K 2-tape package...$24.95

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

ASSEMBLER EDITOR

From Atari
Write your own 6502 assembly language programs. The editor and assembler options operate alone or with tape or disk storage, and a comprehensive debugger/disassembler is included.

ROM cartridge & manual...$54.95

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.

Prices Subject to Change

THE PROGRAM STORE
4200 Wisconsin Avenue NW, Dept. U12 Box 9609
Washington, D.C. 20016

Ask The Readers

Robert Lock, Richard Mansfield
And Readers

This is *your* column — readers ask the questions and other readers answer. Please address any questions or answers to: Ask the Readers, **COMPUTE!**, P.O. Box 5406, Greensboro, NC 27403.

"I have a suggestion for contributors with programs:

When listing programs it would be exceedingly helpful if lines were numbered in regular increments. Since most of us have Tool Kits it would save considerable time if we who enter our own programs would not have to check that a number 182 or 287 didn't suddenly sneak in after a regular sequence of 10,20,30,40 etc.

[Not using] regularly incremented line numbers negates the value of the Tool Kit's AUTO feature. As an example, recently I was so intent on making sure the entries were correctly typed in that I overlooked checking the numbers. When I was on line 620 the program listing was about 560.

How frustrating!!!!" Edmund N. Ricchezza

We couldn't agree more. **COMPUTE!** has recently published "dynamic keyboard" methods for Atari and Microsoft BASICs (which can be used to generate automatic line numbering during program entry) and a number of software products also provide for it. When generating listings here we are often tempted to renumber programs which arrive with irregular line numbering, but frequently this is impossible because the author has referred to various program lines in his article. Please save everyone time, though, and renumber your programs as Mr. Ricchezza suggests before sending them in to **COMPUTE!**.

"Some of your readers may be interested to know about a 'bug' in a Pet/CBM program called "BASIC AID". Suppose we have this very simple program:

```
1 GOTO 2
2 END
```

If we use the machine language monitor to examine the content of memory beginning at $0401 we will find that the line is as in (a) below. The 04 08 is the link to the next line, the 01 00 is the line number, 89 is the keyword for GOTO, 32 is ASCII 2 and $00 indicates the end of the line.

If we invoke BASIC AID and (re)NUMBER 1,1 and re-examine memory we will find the program looks like line (b) below. Notice that we have picked up a garbage byte after the ASCII 2. If we NUMBER 1,1 again we will find the program content is as in line (c). Note that we have picked up an additional garbage byte. Each time the NUMBER routine is invoked a new garbage byte will be added.

```
(a) 04 08 01 00 89 32 00
(b) 04 09 01 00 89 32 02 00
(c) 04 0A 01 00 89 32 32 02 02 00
```

Consider the following program line:

```
3 ONX GOTO 4, 5, 6, 7
```

The first time the NUMBER routine is used on this program line each of the four line numbers will pick up a garbage byte. However, the second time the NUMBER routine is invoked only the first line number will pick up an additional garbage byte and subsequent line numbers will be left as is.

As a result of this problem the NUMBER routine cannot reliably be used in the BASIC AID program. In particular I have discovered that the "ON-GOTO" statement will tend to bomb in long programs after use of the (re)NUMBER routine." Hal W. Hardenbergh

---

"One day I was programming and I tried to make a variable called COMBAT [on the Atari]. I got an error and after a little deduction I found a command called COM which has something to do with DIMensioning variables, but there is no explanation of it in the Manual. Does anyone know what COM is?" Jeffrey Naiman

COM is, as far as we know, identical to DIM. DIM A$ (30) is the same as COM A$ (30). We do not know why Atari BASIC contains this "extra."

"Does anyone have or know where data is available on the Apple I? Both program data and interface data. We have the monitor listing, but the code seems odd."
Frank Anderson

"A recent article in **COMPUTE!** *#16 began by alluding to the 'hundreds of free systems in operation across the country' (STP-488 A Smart Terminal Program,* et cetera*; P. 108). I assume that the author was referring to information services that one can enter in order to get different kinds of information stored in those systems. If you have a list, complete or partial, of such services, or if you can refer me to a source where I can obtain such information, I would be most appreciative."*
George Liskow

1. CompuServe
   500 Arlington Centre Blvd.
   Columbus, OH 43220
2. Source
   1616 Anderson Rd.
   McLean, VA 22102

For local nets, contact a local computer club.

"Were it not for **COMPUTE!** *it is quite possible that I would not have chosen the Atari in the first place, or having done so, that I would not have kept it.* **COMPUTE!**, *for a novice like me, who does not have time to be a hobbyist, is without peer the finest computer magazine I have encountered; and it appears to be getting better. I anxiously await its arrival every month.*

I am considering upgrading my Atari 800 with more memory and a disk drive. While canned programs are very nearly all written for the 810 disk drive, the 815 double density/double disk is attractive because of its capacity. The questions are these: will the 810 and 815 disk drives work together in a system? Will the canned programs (Visicalc, wordprocessing, etc.) operate with an 815 disk drive?" John Thrash

You are clearly a man of taste and acute judgment. We cannot recommend the 815. Atari has cancelled its production. Because it is double-density, the disk operating system would need to be different from the 810 and having them work together would be very difficult indeed. Although we're sure Atari will continue to support the 815, the cancellation would raise a question in our minds regarding continued support by outside software houses.

---

*"I'm having problems with John H. Palevich's SHOOT Program [***COMPUTE!** *#16]. The* BOOT TAPE MAKER *(Program 1) works fine, all those DATA statements check out okay and it beeps twice, I press RETURN and CSAVE a copy of the* BOOT TAPE.

Now here's where the problem comes in. When I remove the ROM cartridge, rewind the tape to the beginning of the BOOT TAPE, *press "PLAY" on the 410, press down on the START button (what's this for?) and turn the 800 back on, I* don't *get that BEEP that John says I should (in his article). I really want to see this program work, so I can try something of my own, but I'm stuck with joystick in hand and fire button poised.*

Anyone know what I'm doing wrong, or having the same problems, or is my Atari 800 down with something?" Fred Corsale

You first turn power off and remove the cartridge. Then hold down the START button *while* turning on the computer. Press PLAY and hit RETURN. This complex and lengthy program is worth the effort, but, because it was so complex, many beginners had difficulty. The BASIC program (before the DATA statements) must be typed in *exactly* as it appears. For further suggestions, see last month's **COMPUTE!**, "Typing in SHOOT," an article written to aid those who might experience difficulties. ©

COMPUTE! Magazine

When we first started **COMPUTE!** magazine in the Fall of 1979, we made a commitment to build a strong, ongoing users resource. Since that beginning, we've provided thousands of personal computer owners with breakthroughs, innovations, practical software, and in-depth reviews of new and significant products. In every issue of **COMPUTE!**, you'll find dozens of pages showing you exactly how to make better use of your computer; articles which tutor the beginner and challenge the advanced.

We're your specialists, whether your interests lie in home applications, game development, educational insights, or better business planning. Here are just a few facts to tell you more about **COMPUTE!**:

In every issue, you'll find clearly, cleanly printed software you can type right into your computer and use...

Here's a comparison of average machine specific articles in the last three issues of **COMPUTE!**, matched against the average number of articles in the "Big Four" ...the personal computer magazines with circulations of 100,000 or more:

Computer	COMPUTE!		"Big Four"	
Type	Articles	Programs	Articles	Programs
Atari	10.67	13.3	[illegible]	[illegible]
PET/CBM	11.33	11.0	[illegible]	[illegible]
Apple	4.0	6.0	[illegible]	[illegible]
OSI	2.0	4.0	[illegible]	[illegible]

(Numbers are average number of articles or programs in **COMPUTE!** issues 16, 17, and 18 compared with the average number of articles or programs in each of the same three issues of the four largest personal computing magazines.)

Subscribe to **COMPUTE!** today! We're the Resource...

Announcing Home and Educational COMPUTING!, the newest publication from the publishers of COMPUTE!

We've expanded the scope of our new magazine and, beginning in January, you'll find resourceful coverage and helpful information for the **VIC-20, the Atari 400, the TI 99/4A, the Radio Shack Color Computer, the Sinclair ZX-81,** and more. We'll be covering personal applications, education, new thresholds, the pocket computers, and a great deal more.

COMPUTE! Books

Our **First Book of Atari** is now shipping (if you've already ordered one it's on the way). If you haven't, check with your dealer, or order directly from us by using the convenient toll-free numbers below.

In the continental US, Call TOLL FREE 800-345-8112. In PA call 800-662-2444.

SUBSCRIPTION RATES

	COMPUTE! (monthly) Twelve issues	Home And Educational COMPUTING! (bimonthly) Six issues
US:		
One Year	$20.00	$10.00
Two Years	$36.00	
Three years	$54.00	
Canada	$25.00	$12.00
Europe, Air Delivery	$38.00	$18.00
Elsewhere, Surface	$25.00	$12.00
Air:		
Middle East, Central America and North Africa	$48.00	$24.00
South America, South Africa, Far East and Australia	$88.00	$24.00

All subscriptions from outside the US must be prepaid in US funds drawn on a US bank or international money order. Visa/MasterCard also accepted.

Please enter my subscription to: ☐ **COMPUTE!**
☐ 1 year ☐ 2 years ☐ 3 years
☐ **Home and Educational COMPUTING!**

Please send me ______ copy(s) of **COMPUTE!'s First Book of ATARI** at $12.95 each + $2.00 shipping/handling.

Name ____________________

Address ____________________

____

For fastest service in the US call the toll free number above, or send this coupon to: **COMPUTE!**, 515 Abbott Drive, Broomall, PA 19008.

Bill me orders (US only) are charged a $1.00 billing fee.

COMPUTE!
515 Abbott Drive
Broomall, PA 19008

Computers And Society

David D. Thornburg
Innovision
Los Altos, CA 94022

The Personal Computer As A Tool For Creative Expression ...

While machines are probably incapable of what we would call creative thought or invention, there is no question in my mind that the personal computer will become the next major tool for creative expression. In fact, the micro has already become invaluable to many artists whose medium is the written word. The ability of word processing software to simplify the capture and subsequent manipulation of words is of exceptional benefit to many writers, be they poets, essayists, or novelists. Of course, the word processor was not created for this audience — it was created for business users. It is the similarities of the text manipulation needs of both these audiences which has allowed this one tool to be so versatile.

In looking at other fields of expression, the artist is not so fortunate. Software packages for music and graphics are in their infancy. Nonetheless, it is clear that the development of additional software tools can expand the personal computer from a word processor into an *idea processor*. There is no intrinsic reason why small computers can't provide the means for capturing and editing musical or graphical ideas with any less facility than for "words."

In addition to its role as an idea capturing device, the personal computer is fast becoming an appropriate medium through which artistic ideas can be expressed. The temptation of many people working in this area is to try to make the computer emulate existing media. I think that this is a mistake. The computer should be thought of as a *new* medium which is as different from other media as the pencil is from oil paints.

Most of the computer generated music I have heard on micros has attempted to copy the sounds of existing instruments. I would guess that, given the choice, most of us would be less impressed by hearing a computer synthesis of Bach's *Toccata and Fugue in D Minor* than we would by hearing this same piece performed on a 6700 pipe Ruffatti organ (for a superb example of the latter, I recommend the direct to disk recording Virgil Fox made for Crystal Clear Records). There is no way that any synthesized sound can accurately model the depth and richness of even the most modest pipe organ.

A New Class Of Instrument

This doesn't mean that composers and musicians should avoid computers — only that they should consider the computer to be a new and different tool for musical expression — a new class of instrument which can double as a composition tool.

Even if the computer had no capabilities to assist in the synthesis of sounds, imagine the tremendous benefit which would come from the existence of a well written music editor. If you have ever composed music, you have undoubtedly noticed the tremendous expenditure of effort required to capture your melody on paper. A well written music editor might let you play at a special keyboard. As you played, each note and duration would be stored in a file for later editing. After the basic melody has been captured, you would then be able to "clean up" the musical score, align chords, repeat melodic phrases, perform transpositions, inversions, etc. The existence of such an editing tool would benefit existing composers as well as those performers who want to create new compositions on their own.

... the development of additional software tools can expand the personal computer from a word processor into an idea processor.

I find it quite heartening, in my ramblings around various college campuses, to see Apples and other personal computers located in music departments. The work, so far, is most crude, but at least some people recognize the potential hidden in these machines.

If music editing and performance are appropriate domains for the personal computer, then these machines are even more appropriate tools for the graphic artist. The resolution and color capabilities of the Atari and Apple computers are

extremely poor when compared to lower-cost, familiar tools, such as watercolors or oil paints. But, as with music, it is a mistake to think of the computer as a replacement for existing media. The computer will no more displace the canvas than the pencil replaced the charcoal stick. The artist who uses the computer will be creating works of art which are not expressable in other media. Graphic tools such as realtime animation, dynamic hue or luminance variation, or responsiveness to the viewer, are just not available through media like water colors. Provided that the user interface is appropriate, the artist is able to do any of these things with a computer as inexpensive as the Atari 400.

In order for the personal computer to be useful to the graphic artist, the interface between artist and machine needs to be most carefully crafted. In addition to input devices such as graphics tablets and output devices such as color bit-map printers, the artist needs a graphic idea-capturing and editing tool which does not interfere with the flow of creative expression. Normally, one associates human interaction with a computer keyboard with "left-brained," linear, analytical thinking. The creative flow of ideas, on the other hand, is generated by "right-brained" thought patterns. Somehow, the software through which the artist communicates with the computer must be designed to keep the artist in a creative frame of mind. This will probably make useful graphic editors harder to create than the programs which presently facilitate the generation of "business" graphics (pie charts, bar braphs, etc.).

The Graphics Gathering

It is my pleasure to be part of an informal group, centered around Stanford University, called the Graphics Gathering. This group assembles every month or so to exchange ideas and to show films, slides, or "live" demonstrations of art which has been created with the assistance of technology — primarily computer technology. The most exciting aspect of this group is that artists who are interested in technology converse freely with computer professionals who are interested in graphics. The exchange of ideas benefits everyone.

I recently gave a presentation on Turtle Geometry to this group. (The interested reader is encouraged to explore the "Friends of the Turtle" column which will be a regular feature in **COMPUTE!**'s sister publication, *Home and Educational COMPUTING!*). The simple syntax of the graphics commands used in user-friendly languages such as Atari PILOT and TI LOGO convinced some artists that the day would soon arrive when they could use personal computers for their own artistic creations.

There are few impediments to the use of computers by artists. Cost is no longer much of a factor, although a full-blown system can cost as much as you want it to. Still, with an entry fee of $400 or so, motivated artists can start experimenting with this medium. The real limitation is simply the absence of appropriate software. Once high quality, user friendly, and versatile editors are generated, we can expect to see many artists adding the computer to their tools of expression. Within a decade we might expect to see a projection display in every major gallery, with artists opening shows in several cities by sending their works over the telephone lines. Art collectors may start collecting disks!

How and when this happens may depend on you. As someone who uses and is interested in personal computers, you might be in a good position to experiment with the creation of some of the tools needed by artists of all types.

Notes From All Over ...

Judging from the letters and phone calls I have received, the September editorial on Artificial Intelligence was of interest to many of you. To all who took the time to contact me, I extend my sincere thanks. Your comments, both pro and con, were most valuable. In light of your interest in this area, I will devote the next column to a few recently published books on this topic, including *The Mind's I* by Douglas Hofstadter and Daniel Dennett, *Brainstorms* by Daniel Dennett, and *Mind Design*, edited by John Haugeland. Until then, I extend my wishes for a happy holiday season and a most propserous new year. ©

Guest Commentary:

The "World Computer" Revisited

Marvin DeJong
The School Of The Ozarks
Pt. Lookout, MO

This letter is written in connection with the responses I have received to my guest editorial in **COMPUTE!** #14 (page 18). In particular, I am concerned with Mr. Vern L. Mastel's letter in **COMPUTE!** #17 (page 16).

Mr. Mastel implies that my ideas regarding standardization would be the Armageddon of the personal computer industry. The questions he raises and the scenario he depicts are the products of his imagination, not mine.

Let me respond to some of his concerns by pointing out that *standardization* has long been an important factor in the electronics industry as a whole and in the computer industry as well. The IEEE has numerous committees working to standardize various components of the electronics industry, including bus structures, interfaces, and languages.

I can purchase a record from *any* manufacturer (RCA, Columbia, etc.) and put it on *any* turntable or record changer, connected to *any* amplifier, and I will hear music. This is the blessing that results from industry standardization of recording format, speed, and frequency response curves. It is neither "horrifying," nor is it a "nightmare" to operate. (Words in quotes were used by Mr. Mastel.)

Likewise, the industry standards for transmission, reception, and formatting of television pictures have not produced any "monstrous" results. On the contrary, the fact that any TV set (in the United States) can receive any network, all channels, and any local TV station, has been a boon to the industry. My 15-year old black and white set is perfectly compatible with the new color sets. I can purchase a video monitor from any manufacturer and it will work with almost any personal computer as a result of standardization.

Another person who responded to my editorial claimed that the "standard" computer would be restricted to a single microprocessor. Nonsense! The microprocessor and its unique assembly language are completely transparent to anyone who programs in a high-level language. Microsoft has written a BASIC interpreter for almost every microprocessor, it seems. The problem is that *BASIC is not standardized.* There are many different kinds of BASIC. The people who wrote ADA are apparently making efforts to insure that this does not happen to it.

Mr. Mastel implies that we must make a choice between one of the many high-level languages (he included CP/M, which is not a language). I do not think it is an either/or situation. Interpreters are either in ROM or on a disk, and may easily be changed. My idea of a standard computer would be one in which language cards could be plugged in or removed.

It might be well to reiterate my original points. In the context of *educational* uses of the personal computer (an elementary, middle, or high school classroom for example):

1. The cassette recorder is an unacceptable device for storing programs and the industry, including software vendors, should be realistic about its weaknesses.

2. Compatible disk operating systems and *standard* versions of any high-level language would allow software to be easily transported from one machine to another, resulting in reduced software costs and increased incentives for the people who like to write software.

3. *Standardized* graphics commands (with the origin of the coordinate system in the lower left-hand corner where it was for several hundred years before the computer arrived on the scene) would also make transporting a program from one machine to another an easy task. Standardized graphics commands must be built into the interpreter.

4. Standard printer, disk, modem, and plotter interfaces would make assembling a computer system much easier. In a sense this is history, since the RS-232C is already standardized for serial interfaces and Centronics handshaking has become a *de facto* standard for parallel interfaces, while the IEEE-488 bus is used for instrumentation.

My comments were not intended to unveil a poorly disguised communist plot to bring the personal computer industry to an untimely demise. On the contrary, I would like to see the industry become more standardized so that the use of a computer by any elementary school teacher or pupil is simple, inexpensive, trouble-free, educational, and entertaining.

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

The Beginner's Page

Richard Mansfield
Assistant Editor

Checksum, Terabytes, And Disaster Avoidance

In many ways, your brain is an ideal data storage device. It is in a dust-free case, it can hold an estimated twelve-and-one-half-million terabytes (12,500,000,000,000,000,000 eight-bit bytes. An average microcomputer disk holds about 170 thousand bytes), it self-regulates temperature, and it uses about the same amount of electricity as a twenty watt lightbulb. All in all, an impressive memory.

Until we can manufacture memory devices of this excellence, we will have to follow some rules to make sure that our data and programs are safely stored on tape or disk. Most of our computers rely on memory chips which hold only a few K. The "K" means *kilobyte*, 1,024 bytes. This is not much, really. One kilobyte could hold about 175 English words; less than a double-spaced, typewritten page. To hold this page of **COMPUTE!** we would need about 6K RAM. In an 8K computer, that would leave little space left over for a word processor program to allow corrections, additions, and everything else.

The future of memories looks bright though. 64K on a single chip will be available to us fairly soon — even greater densities, at lower prices, seem inevitable. In fact, there is a possibility that memory cells might actually be *grown*, like mushrooms. Efforts are now underway to create *protein* memory cells. But, for now, we must do without unlimited, inexpensive memory. For now, we compose programs and enter data into a limited RAM and then SAVE what we've created onto cassette tapes or disk drives.

The word SAVE implies a kind of safety, a secure storage. It *can* be secure, but you should observe some precautions. Last month we looked into the management of files. Normally, a file of data is typed into the computer, SAVEd as a file, and then used by a program or programs. The data is kept on a disk or a tape because the computer wipes its RAM memory clean each time power is turned off or each time a new program or set of data comes in.

Backup

Redundancy is an important feature of SAVEing. On your part, this means keeping a backup copy of each program or file. When you write a program (or buy one), the first thing to do is to make a second tape/disk copy of it and put it in a cool place in a dust-free, plastic box. Dirt, smoke, heat or extreme cold, and the oils on fingers are all enemies of magnetic data because both tape and disks are a thin plastic which is easily deformed.

Another danger is vacuum cleaners, TVs, or nearly anything which uses electricity and can generate electric fields. This can remagnetize (erase) tapes and disks. So you cannot safely put a cassette on top of a TV or a refrigerator.

Computers can help us by using their own redundant method of data backup. When a program is sent to a tape machine, some computers record the entire program *twice*. Then, when the program is LOADed back into the computer, the two versions can be compared. The computer then can use the "best" version if there are differences. How does it know which is best?

Data is transferred very fast and many things can degrade it. Often, a *checksum* is used to see if the data made the trip intact. There are various checksum schemes, but here's a simple one. Imagine that we were sending the word *face* to a cassette. The computer would send the numbers 70-65-67-69-271. The letters of the alphabet are each given a code number in computing (the ASCII code). Uppercase A is the number 65, B=66, C=67, D=68, E=69, F=70 and on up. Computers work *only* in numbers. The word *face* means nothing to the computer — it is merely a pattern of numbers. It can print the pattern, alphabetize it (which, to a computer, is merely putting the numbers in *numerical* order), search for it in a paragraph, and all the rest — without ever thinking of the word as anything other than a particular number sequence.

So, it is easy to see why the computer sends 70 65 67 69 271 to the cassette. The number 271 is the sum of the previous numbers. While sending them to the tape, the computer is also adding them up and sending the total at the end. Then, when LOADing, it also adds them up and checks its sum against the one that comes in from the tape. If the sums are not the same, then there was an error in the data transfer. An error of addition is nearly as impossible for a computer as taking a wrong turn would be for a roller coaster. It has been known to happen, but we can be almost certain that it will never happen to you. The computer can be virtually sure that mismatched checksums are the result of bad data on the tape.

This is how it knows which is best of the two versions it recorded on tape. If version one had a bad checksum on the word *face*, but a good checksum on the word *lift* it could keep the word *lift*, but

WHY THE MICROSOFT RAMCARD™ MAKES OUR SOFTCARD™ AN EVEN BETTER IDEA.

Memory — you never seem to have quite enough of it.

But if you're one of the thousands of Apple owners using the SoftCard, there's an economical new way to expand your memory dramatically.

16K ON A PLUG-IN CARD.

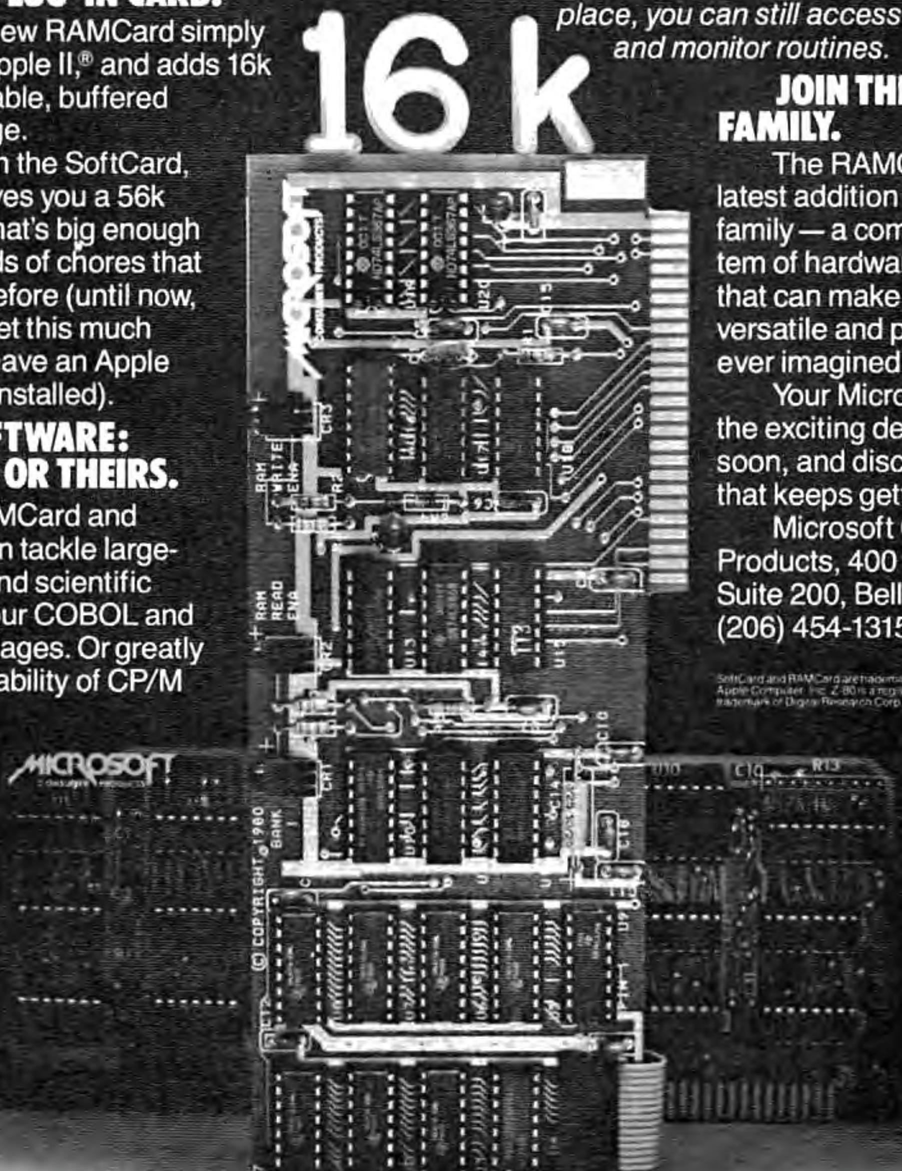
Microsoft's new RAMCard simply plugs into your Apple II,® and adds 16k bytes of dependable, buffered read/write storage.

Together with the SoftCard, the RAMCard gives you a 56k CP/M® system that's big enough to take on all kinds of chores that would never fit before (until now, the only way to get this much memory was to have an Apple Language Card installed).

GREAT SOFTWARE: YOURS, OURS, OR THEIRS.

With the RAMCard and SoftCard, you can tackle large-scale business and scientific computing with our COBOL and FORTRAN languages. Or greatly increase the capability of CP/M applications like the Peachtree Software accounting systems. VisiCalc™ and other Apple software packages can take advantage of RAMCard too.

And RAMCard gives you the extra capacity to develop advanced programs of your own, using the SoftCard and CP/M. *Even with the RAMCard in place, you can still access your ROM BASIC and monitor routines.*



JOIN THE SOFTCARD FAMILY.

The RAMCard is just the latest addition to the SoftCard family — a comprehensive system of hardware and software that can make your Apple more versatile and powerful than you ever imagined.

Your Microsoft dealer has all the exciting details. Visit him soon, and discover a great idea that keeps getting better.

Microsoft Consumer Products, 400 108th Ave. N.E., Suite 200, Bellevue, WA 98004. (206) 454-1315.

SoftCard and RAMCard are trademarks of Microsoft. Apple II is a registered trademark of Apple Computer, Inc. Z-80 is a registered trademark of Zilog, Inc. CP/M is a registered trademark of Digital Research Corp. VisiCalc is a trademark of Personal Software, Inc.

MICROSOFT

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

ALTOS

ACS 8000-2 64K 1M	$ 4500
ACS 8000-15 64K 1M	$ 5990
ACS 8000-6 208K 14.5M	$10490
ACS 8000-7 208K 29.0M	$11690
ACS 8000-10 208K 10M	$ 8500
ACS 8000-10/MTU	$10990

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

DIABLO 630 PRINTER

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

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

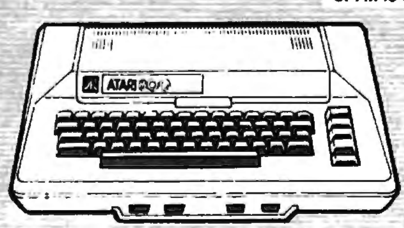
AMDEK MONITORS

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 ANY ADVERTISED PRICE ON PRODUCTS LISTED UNDER SIMILAR "IN STOCK" CONDITIONS.

NEECO
679 HIGHLAND AVE.
NEEDHAM, MA 02194

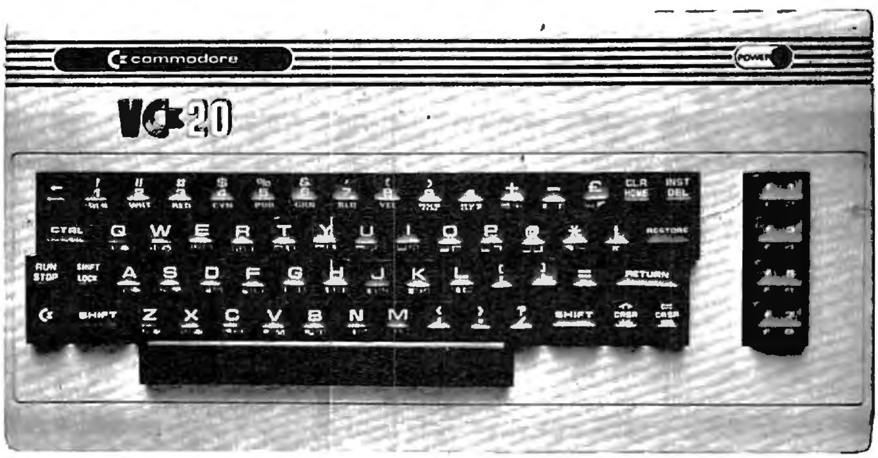
(617) 449-1760
Telex: 951021

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

NEECO INTRODUCES THE CBM VIC-20 COMPUTER!

Commodore breaks the computer price barrier —

$299.95

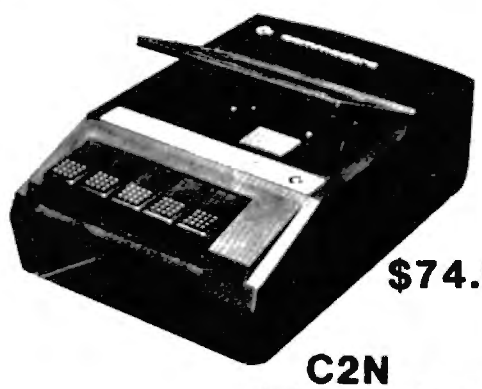


CBM VIC-20 PERSONAL COMPUTER

VIC-20 SPECIFICATIONS

- 8 colors – built in
- sound generation – built in
- programmable function keys
- 5K memory expandable to 32K
- standard PETBASIC in ROM
- full-size typewriter keyboard
- graphics character set
- plug-in program/memory cartridges
- low-priced peripherals
- joystick/paddles/lightpen
- self-teaching materials

* WORKS WITH ANY HOME TELEVISION



$74.95

C2N TAPE CASSETTE DRIVE

CALL NEECO TODAY FOR ADDITIONAL VIC-20 INFORMATION . . .

As the CBM VIC-20 is a "new" product, prices and specifications are subject to change w/o notice.

NEECO

NEECO WILL MATCH ANY ADVERTISED PRICE ON CBM EQUIPMENT FROM ANY OTHER COMPANY WITH PRODUCT IN STOCK.

679 HIGHLAND AVE.
NEEDHAM, MA 02194

(617) 449-1760
Telex: 951021

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

wait for the word *face* in the second version. Checksums are done on longer samples than individual words, but the technique is the same.

Computer Wrestling

All of this is an effort, by the computer as well as by the computerist, to protect data. If you make a backup and the computer makes two versions — there are four copies of a program or file. There are two more ways to prevent problems: scratchpad SAVEs and respect for your computer.

When you write your first database program you might want to consider what you are up against. Building a database means typing in lots of records. You do not want to do it twice. Last month we set up a database management system which would permit instant indexing of **COMPUTE!** articles by author or by topic. If you are planning to type hundreds of records (each subject-author-issue number is a record) you don't want to work for hours only to have a fuse blow or someone trip over the computer's electric cord. In a flash, your data is destroyed.

To avoid this, it's a good idea to keep a cassette or disk which is labelled "SCRATCH." It is a temporary scratchpad which is left in the tape or disk drive and SAVEd to every half hour while you rest your fingers.

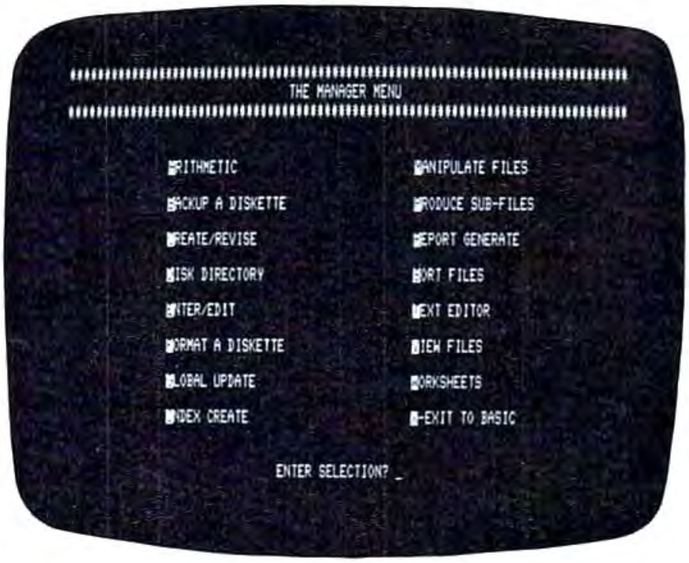
Finally, the machines themselves, the computers and disk/tape drives, deserve respect. This means gentle treatment. We all know someone who has problems with machines — knobs break off, keyboards malfunction, things jam and fail. They are frustrated by constant "bad luck" with machinery, but, if you watch them make a tape copy, you'll see what's wrong. They move quickly, they force a balky lid down, they *fight* their machinery. To further compound the problem, this same personality type usually avoids instruction manuals. They don't learn that placing electronic devices in direct sunlight, transferring finger oil via disks to drive heads, plugging in peripherals with the power on all invite disaster. We all have our faults, but computer wrestling is an expensive fault. Repairs are slow and expensive. Computer technicians are in short supply.

Transistorized devices are among the most reliable machines man has ever built. A bit of caution and care will keep your data intact and your machine out of the repair shop — until we can buy those disposable terabyte protein box memories for $1.

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

Basically Useful BASIC

A Quick-Fix Approach To Calculating Tables

Edward Heite
Camden-Wyoming, DE

Some programs that purport to solve simple problems are, in themselves, too complicated to justify the effort of keying them in. A quickie program should, by definition, be simple and to the point. In my work as an archaeologist, I am often called upon to convert archaic units of measurement to modern units. Old surveys, for example, are expressed in "poles" or "perches", which are 16½ feet long.

To create a quickie conversion table from poles to feet, I wrote this jiffy program:

```
5 OPEN 1,4,0
6 PRINT#1,CHR$(147)
10 FOR F=1 TO 320
20 R=F*16.5
30 PRINT#1,F "POLES EQUAL" R " FEET."
40 NEXT F
50 CLOSE 1
60 END
```

Program 1.

```
1 POLES EQUAL  16.5  FEET.
2 POLES EQUAL  33  FEET.
3 POLES EQUAL  49.5  FEET.
4 POLES EQUAL  66  FEET.
5 POLES EQUAL  82.5  FEET.
6 POLES EQUAL  99  FEET.
7 POLES EQUAL  115.5  FEET.
8 POLES EQUAL  132  FEET.
9 POLES EQUAL  148.5  FEET.
10 POLES EQUAL  165  FEET.
11 POLES EQUAL  181.5  FEET.
12 POLES EQUAL  198  FEET.
13 POLES EQUAL  214.5  FEET.
```

Figure 1.

My 2022 printer obediently produced a table to convert poles to feet, from one pole to 320, which is a mile. It's a totally unremarkable program; there are no fancy columns, headings, or symbols.

But such fancy programming would have been time-consuming, and would have defeated the initial purpose of providing a quick chart. Since the program is so short, it can be typed for each use, more quickly than it could be loaded from tape.

For those who must frequently calculate conversion tables, a library of quickie programs can be kept on Rolodex cards, ready for instant reference. ©

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

Window Analysis:
Saving Fuel $$ With Your Computer

David Pitts
Houston, TX

In a typical home, the sunlight transmitted through the windows accounts for 10-14% (ref. 1) of the total air conditioning cost. This can be equal to the savings accrued by installing storm windows or adding attic insulation in some regions of the United States. Furthermore, shielding windows by planting trees or using solar screen, is usually much less expensive than adding insulation or storm windows (especially if one treats only the windows which need shading). The window analysis program described here will allow the homeowner to calculate how much is saved by shading windows in the summer and augmenting the winter heating by allowing sunlight into the home. Also, the program can be used for planning solar collector systems, designing greenhouses, evaluating the merit of adding skylights, or enclosing porches with glass.

As shown in the example run, the user inputs the latitude, the size of the window, the tilt of the window from horizontal, the azimuth [compass directions] that the window faces, and chooses either heating or cooling analysis to be performed. If cooling analysis is desired, the user inputs the capacity (tons) of the cooling system, the current it draws (amps) and the cost of the electricity. If the user chooses heating analysis, he must input the cost of natural gas. Both fuel savings, economic savings and the accrued energy in BTU/sq. ft. are printed by month and season. Because the window azimuth and elevation angle permit any angle window to be analyzed, a variety of applications are possible. In the author's residence, the east-facing windows cause almost $100 in excess cooling cost, whereas the winter gain is about a factor of three smaller. At the low latitude of the author's residence, south-facing windows do not contribute significantly to the heat load in the summer, but are important in reducing heating cost when the sun is lower in the southern sky.

The Calculations

The program was written in Microsoft BASIC on an OSI 4PMF using simple I/O so that the program could be easily converted to other systems. However, lines 372-373 should be replaced for other microprocessors since they provide a flashing cursor on the OSI 4P. The program utilizes eight basic equations which describe the physical amount of sunlight and the angle at which it falls on the window's surface (ref. 2 and 3). The day of the year (DOY) is calculated from the month (M) and the day of month (D) in line 227. The solar declination (DE) is calculated from the day of year in line 350-360. The cosine of the zenith angle of the sun (A1) is calculated in line 440 from the solar declination, the hour angle, and solar elevation angle (AL). The direct solar irradiance is calculated in equation 480 from the apparent solar irradiance at zero air mass (AO), the atmospheric extinction coefficient (BETA) and solar elevation angle (AL). The diffuse irradiance is calculated in line 490 from the tilt of the window (TI) and the direct solar flux (GN). The cosine of the angle between the vector perpendicular to the window and the vector to the sun is calculated in lines 560-570, based on the window tilt (TI), the window azimuth (BI), the sun's azimuth (AZ), and the sun's zenith angle (Z). Finally the total flux transmitted through the window (GL) is calculated in line 600 and summed by month (TT) and by season (SL).

The integration of transmitted energy during a day is accomplished in the FOR loop from line 370 to 712. In this loop, calculations are made during a day for hour angles (HE) of minus 120 degrees (4 AM local solar time) to plus 120 degrees (8 PM local solar time). It is assumed that this calculation is valid for ten days. The integration by month is accomplished between lines 348 and 713 in which three ten-day intervals are calculated per month.

The conversion from energy to utility usage is made assuming that 1100 BTU are produced by each cu. ft. of natural gas and air conditioner run time can be calculated from BTUs by the factor 12,000 BTU/(hr. ton). Kilowatt hours are calculated from volts times amps times time divided by 1000. The program is designed to be used at any latitude (except 0). However, if southern hemisphere calculations are desired, the seasons must be switched in line 225 (the starting month M for heating =11, and for cooling =5). Likewise, the length of the heating and cooling seasons must be modified from 152 days (line 715) for printing routine (line 719) should be modified for heating and cooling seasons appropriate for the long season regions. Special transmissionn functions for double glazed glass or solar film may be substituted for the subroutine in lines 2000–2050 as desired.

The Portable Computer.

ATS Makes A Case for Versatility - Convenience - Protection

VERSATILITY: ATS allows you to get a handle on enhancing the utility of your personal computer — now you can have it everywhere you'd like to. Your wish to have your home computer at the office or to have your office computer at the sales appointment has finally been granted. Now you can take your Apple or other computer to **work, home, on trips, to conferences, for demonstrations, estimates,** or **other tasks.** In fact, our cases enable you to use your computer

ANYWHERE, ANYTIME.

CONVENIENCE: Our cases are quick and easy to use. We have eliminated the necessity of disconnecting the computer parts so you can pack or unpack in a matter of seconds or, if desirable, you can even operate your computer without removing it from the case.

Lite Duty Case for Apple II
with two disk drives

only $159

(for information on other types of cases contact your dealer)

PROTECTION: Our case will shield your valuable computer from bumps, scratches, dust, and the elements whether in transit or in storage.

- Sturdy construction - High impact ABS plastic laminated to solid core plywood
- Aluminum reinforced edges
- Industrial heavy duty hardware
- Custom fitted foam interior holds, cushions, and protects contents
- Available in standard and custom sizes to fit **micro-computers, terminals, monitors, printers, keyboards,** and **others**
- Inexpensive

Lite Weight Carrying Cases — optimum portability for daily or routine travel. Sturdy construction. Full foam interior.

Heavy Duty Shipping Cases — designed to withstand the rigors of air and surface transportation assuring safe arrival of valuable computer systems.

System Shipping Cases
standard or custom designed to transport complete computer systems including monitor, motom, printer, plus storage compartments.

ATS CASES

25 Washington Ave. • Natick, MA 01760 • 617/653-6724

References

1 *Houston Lighting and Power Residential Conservation Services*, copyright 1981, Planenergy Inc., Austin Texas.

2 Yellot, John I.: 1974, *Solar Energy Utilization for Heating and Cooling*, NSF 74-41, available from the Superintendent of Documents, U.S. Government Printing Office, Washington, DC 20402.

3 Klem, David C., 1980: Solar Specs, *Microcomputing*, pp. 68-70, 1980.

Program 1. Microsoft Version

```
10 REM **********WINDOW HEATING ANALYSIS**************
30 REM PROGRAM CALCULATES SOLAR RADIATION TRANSMITTED
31 REM THROUGH A WINDOW GIVEN LATITUDE, AZIMUTH AND ZENITH AND
32 REM ANGLE OF WINDOW-DAVID PITTS 16011 STONEHAVEN DR HOUSTON TX 77059
35 REM AL=SOLAR ALTITUDE,HE=HR ANGLE,DE=DECLINATION
36 REMTR=TRANSMISSION,SL=SEASONAL TOTAL BTU/SQ FT,TT = MONTHLY
37 REM TM=TIME(HRS),AO=APPARENT SOLAR IRRADIANCE AT ZERO AIR MASS
38 REM BETA=ATMOSPHERIC EXTINCTION COEFFICIENT
44 FORI=1TO20:PRINT:NEXT
45 PRINTTAB(15);"WINDOW ANALYSIS - SOLAR TRANSMISSION"
46 PRINTTAB(25);"D. E. PITTS"
47 PRINT:PRINT
48 PI=3.14159:P2=PI/2:DIMBETA(12),AO(12)
50 DEFFNRAD(A)=A*PI/180
51 DEFFNASN(B)=ATN(B/(SQR(1-B^2)))
52 DEFFNACS(C)=ATN((SQR(1-C^2))/C)
53 DEFFNDEG(D)=INT((D*180)/PI)
54 DEFFNTRC(E)=INT(E*100)/100
55 DEFFNFUN(F)=F*180/PI
200 INPUT"LATITUDE(DEG)";LAT:L1=LAT:LAT=FNRAD(LAT)
223 PRINT"ANALYSIS DESIRED":PRINT" 1) HEATING":PRINT" 2) COOLING"
225 M=11:INPUTX:D=1:IFX=2THENM=5\
226 IFM<3THENDOY=M*31-31+D:GOTO240
227 DOY=INT(M*30.6-32.3+D):REM DAY OF YEAR
240 FORI=1TO12:READAO(I),BETA(I):NEXT
250 INPUT" # SQ FT OF WINDOW FOR EVALUATION";FT:PRINT
260 PRINT:INPUT"WINDOW TILT FROM HORZ, NORMAL=90";TI:T1=TI
261 INPUT"WINDOW AZIMUTH(N=0,S=180), DEG";BI:B1=BI:TI=FNRAD(TI)
262 BI=FNRAD(BI):IFX=1THEN310
263 INPUT"AIRCONDITIONER TONS";T:T=T*12000
264 INPUT"ENTER AMPS OF AIRCONDITIONER, IF NOT KNOWN ENTER 0";SE
265 IFSE<1THENSE=25
267 INPUT"COST FOR ELECTRICITY, CENTS/KWH";C:GOTO335
310 PRINT:INPUT"COST OF NATURAL GAS (CENTS/CU FT)";C
335 PRINT:PRINT:PRINT:PRINTTAB(5);"WINDOW HEATING ANALYSIS BY DIRECT
    SUNLIGHT"
340 PRINTTAB(17);"LATITUDE = ";L1;"DEG"
345 PRINT:PRINT"WINDOW ANGLE ";T1;"DEG";TAB(29);"WINDOW AZ= ";B1;" DEG"
348 PRINT:TT=0:FORJ=1TO3
350 X=FNRAD(DOY-82)*180/182.5:X=23.5*SIN(X):HE=-135
355 M=1:IFDOY>31THENM=INT((DOY+32.3)/30.6)
360 DE=FNRAD(X)
370 FORI=0TO16:AM=4+I:HE=HE+15:AN=FNRAD(HE)
372 IFPEEK(55104)<>95THENPOKE55104,95:GOTO410
373 IFPEEK(55104)=95THENPOKE55104,161:REMCURSOR
410 A1=COS(DE)*COS(AN)*COS(LAT)+SIN(DE)*SIN(LAT)
420 X=FNACS(A1):AL=P2-X
425 IFAL>P2THENAL=AL-PI
440 A2=COS(DE)*SIN(AN)/COS(AL)
441 REM IFA2<-1THENA2=-.9999
```

It's time Your Computer stopped just playing games

and started doing some work around the house!

Let Creative Software's home programs turn your ATARI® or VIC® into a really useful household appliance—the results may well amaze you!

TITLE	ATARI 400/800	VIC (cassette only)
• Household Finance	34.95 cassette 39.95 disk	34.95
• Home Inventory	19.95 cassette 24.95 disk	14.95
• Car Costs	19.95 cassette 24.95 disk	14.95

CREATIVE SOFTWARE

201 San Antonio Circle, #270
Mountain View, CA 94040
(415) 948-9595

Ask about our many other recreational and home applications!
TO ORDER: VISA/MasterCard, check or money order accepted. If charge, please include expiration date of card. Add $1.50 for shipping and handling. Calif. residents add sales tax.

LETTER QUALITY WORD PROCESSOR PRINTER/TYPEWRITER

FOR APPLE, ATARI, COMMODORE, NEC, OSBORNE 1, TRS 80 and others

OLYMPIA ES100/101

- 92 character electronic keyboard
- 8 character buffer memory
- Dual pitch, 10 and 12
- 17.5 C.P.S.
- All settings from keyboard
- Auto. correction
- Daisy type print mechanism
- Cartridge ribbons
- 14 1/8 inches writing line
- 1400 dealers nationwide

REN TEC ES

- Installation in 15 minutes using existing ES100/101 cables
- CMOS logic for minimal drain on ES100 power supply
- Hi or low true status bits
- Accepts RS232 serial with 7 crystal controlled Baud rates
- Accepts Centronics parallel interface
- Selectable auto. line feed

$1495.00*ES100 TYPEWRITER & INTERFACE **$295.00 INTERFACE**

RENAISSANCE TECHNOLOGY CORPORATION

3347 VINCENT ROAD
PLEASANT HILL, CALIFORNIA 94523
(415) 930-7707

* * * * *

	Sug. Retail
NEC Dot Matrix Printer	795.00
100 CPS	
Bidirectional printing	
Friction and tractor feed	
Parallel Interface	
Single ribbon cartridge	
NEC Monitors	
12" Green Screen	285.00
12" RGB Color	1095.00
12" Composite Video	430.00
ATARI 10-Key Accounting Pad	124.95

DEALER INQUIRIES WELCOME

```
442 REM IFA2>1THENA2=.9999
445 X=(COS(DE)*COS(AN)-SIN(AN)*COS(LAT))/(COS(AN)*SIN(LAT))
450 AZ=FNASN(A2)+PI:Z=P2-AL:IFX<0THENAZ=PI-AZ
470 IFAL<FNRAD(1)THENGN=0:GOTO490
480 GN=A0(M)/EXP(BETA(M)/SIN(AL))
490 GD=GN*.75*(1+COS(TI))/12
560 A3=COS(Z)*COS(TI)+SIN(Z)*SIN(TI)*COS(AZ)*COS(BI)
570 A3=A3+SIN(Z)*SIN(TI)*SIN(AZ)*SIN(BI)
575 IN=FNACS(A3):IFIN<0THENTR=0:GOTO600
590 GOSUB2000
600 GL=(GN*A3*TR+GD)*10:TT=TT+GL:SL=SL+GL
712 NEXT:DD=DD+10:DOY=DOY+10:IFDOY>365THENDOY=DOY-365
713 NEXT:PRINT"MONTH= ";TAB(9);M;TAB(29);"TOTAL = ";TAB(38);FNTRC(TT);
714 PRINTTAB(48);"BTU/(SQ FT)"
715 GOSUB719:IFDD<152THEN348
717 TT=SL:PRINT"--------------------------------------------------------
718 PRINT"ANNUAL SAVINGS!":PRINT:GOSUB719:FORI=1TO1000:NEXT:RUN48
719 IFM>4ANDM<11THEN800
720 P=TT*FT/110000:PRINT"NATURAL GAS SAVED ";FNTRC(P);"      100 CU FT"
730 PRINT"DOLLAR SAVINGS ";FNTRC(P*C)
740 PRINT:PRINT"--------------------------------------------------------
750 PRINT:RETURN
800 TM=TT*FT/T:P=220*SE*TM/1000
805 PRINT"POWER EXPENDED ";FNTRC(P);"KWH"
810 PRINT"COOLING COST DUE TO WINDOW";FNTRC(C*P/100);"DOLLARS"
820 PRINT:PRINT"-------------------------------------------------------
830 PRINT
1000 RETURN
2000 REM GET TRANSMITTANCE FOR SINGLE GLAZED GLASS
2010 IFIN<.87266THENTR=.87:GOTO2100
2020 IFIN>1.2218THEN2050
2030 CI=(IN-.8726639)*4.5:TR=.16*COS(CI)+.68:GOTO2100
2050 TR=3.0599-1.948*IN:IFTR<0THENTR=0
2100 RETURN
4000 DATA390,.142,385,.144,376,.156,360,.18,350,.196,345,.205,344,.207
4002 DATA351,.201,365,.177,378,.16,387,.149,391,.142
5000 END
```

Program 2. Atari Version

```
10 REM ***WINDOW HEATING ANALYSIS***
30 REM PROGRAM CALCULATES SOLAR
       RADIATION TRANSMITTED
31 REM THROUGH A WINDOW GIVEN LATITUDE,
       AZIMUTH AND ZENITH AND
32 REM ANGLE OF WINDOW-DAVID PITTS
  16011 STONEHAVEN DR HOUSTON TX 77059
35 REM AL=SOLAR ALTITUDE,HE=HR ANGLE,
       DE=DECLINATION
36 REM TR=TRANSMISSION, SL=SEASONAL
       TOTAL BTU/SQ FT, TT=MONTHLY
37 REM TM=TIME(HRS),A0=APPARENT SOLAR
       IRRADIANCE AT ZERO AIR MASS
38 REM BETA=ATMOSPHERIC EXTINCTION
       COEFFICIENT
44 PRINT CHR$(125)
45 ? "WINDOW ANALYSIS - SOLAR TRANSMISSI
ON"
46 POKE 85,14:? "D. E. PITTS"
47 ? :?
48 PI=3.14159:P2=PI/2:DIM BETA(12),A0(12
):RAD
50 FRAD=100
51 FASN=110
52 FACS=120
53 FDEG=130
54 FTRC=140
55 FFUN=150
60 GOTO 200
100 V=V*PI/180:RETURN
```

```
110 V=ATN(V/(SQR(1-V*V))):RETURN
120 V=ATN((SQR(1-V*V))/V):RETURN
130 V=INT((V*180)/PI):RETURN
140 V=INT(V*100)/100:RETURN
150 V=V*180/PI:RETURN
200 PRINT "LATITUDE(DEG)";:INPUT LAT:L1=
LAT:V=LAT:GOSUB FRAD:LAT=V
223 PRINT "ANALYSIS DESIRED":PRINT "1) H
EATING":PRINT "2) COOLING"
225 M=11:INPUT X:D=1:IF X=2 THEN M=5
226 IF M<3 THEN DOY=M*31-31+D:GOTO 240
227 DOY=INT(M*30.6-32.3+D):REM DAY OF YE
AR
240 FOR I=1 TO 12:READ A:AO(I)=A:READ B:
BETA(I)=B:NEXT I
250 PRINT "#SQ FT OF WINDOW FOR EVALUATI
ON";:INPUT FT:?
260 ? :? "WINDOW TILT FROM HORIZ, NORMAL
=90":INPUT TI:T1=TI
261 ? "WINDOW AZIMUTH(N=0,S=180),DEG";:I
NPUT BI:B1=BI:V=TI:GOSUB FRAD:TI=V
262 V=BI:GOSUB FRAD:BI=V:IF X=1 THEN 310

263 ? "AIR CONDITIONER TONS";:INPUT T:T=
T*12000
264 ? "ENTER AMPS OF AIR CONDITIONER":?
"IF NOT KNOW ENTER 0";:INPUT SE
265 IF SE<1 THEN SE=25
267 ? "COST FOR ELECTRICITY, CENTS/KWH";
:INPUT C:GOTO 335
310 ? :? "COST OF NATURAL GAS (CENTS/CU
FT)";:INPUT C
335 ? :? :? :? "WINDOW ANALYSIS BY DIREC
T SUNLIGHT"
340 ? "LATITUDE=";L1;" DEG"
345 ? :? "WINDOW ANGLE ";T1;" DEG   WIND
OW AZ=";B1;" DEG"
348 ? :TT=0:FOR J=1 TO 3
350 V=DOY-82:GOSUB FRAD:X=V*180/182.5:X=
23.5*SIN(X):HE=-135
355 M=1:IF DOY>31 THEN M=INT((DOY+32.3)/
30.6)
360 V=X:GOSUB FRAD:DE=V
370 FOR I=0 TO 16:AM=4+I:HE=HE+15:V=HE:G
OSUB FRAD:AN=V
410 A1=COS(DE)*COS(AN)*COS(LAT)+SIN(DE)*
SIN(LAT)
420 V=A1:GOSUB FACS:X=V:AL=P2-X
425 IF AL>P2 THEN AL=AL-PI
440 A2=COS(DE)*SIN(AN)/COS(AL)
441 REM IF A2<-1 THEN A2=-.9999
442 REM IF A2>1 THEN A2=.9999
445 X=(COS(DE)*COS(AN)-SIN(AN)*COS(LAT))
/(COS(AN)*SIN(LAT))
450 V=A2:GOSUB FASN:AZ=V+PI:Z=P2-AL:IF X
<0 THEN AZ=PI-AZ
470 IF AL<PI/180 THEN GN=0:GOTO 490
480 GN=AO(M)/EXP(BETA(M)/SIN(AL))
490 GD=GN*0.75*(1+COS(TI))/12
560 A3=COS(Z)*COS(TI)+SIN(Z)*SIN(TI)*COS
(AZ)*COS(BI)
570 A3=A3+SIN(Z)*SIN(TI)*SIN(AZ)*SIN(BI)

575 V=A3:GOSUB FACS:IN=V:IF IN<0 THEN TR
=0:GOTO 600
590 GOSUB 2000
600 GL=(GN*A3*TR+GD)*10:TT=TT+GL:SL=SL+G
L
712 NEXT I:DD=DD+10:DOY=DOY+10:IF DOY>36
5 THEN DOY=DOY-365
713 NEXT J:PRINT "MONTH=";M;"  TOTAL=";:
V=TT:GOSUB FTRC:PRINT V;" BTU/(SQ FT)"
715 GOSUB 719:IF DD<152 THEN 348
717 TT=SL:? "-------------------------
----------"

718 ? "ANNUAL SAVINGS!":? :GOSUB 719:FOR
 I=1 TO 1000:NEXT I:CLR :GOTO 48
719 IF M>4 AND M<11 THEN 800
720 P=TT*FT/110000:? "NATURAL GAS SAVED
";:V=P:GOSUB FNTRC:PRINT V;" 100 CU FT"
730 PRINT "DOLLAR SAVINGS ";:V=P*C:GOSUB
 FTRC:PRINT V
740 ? :? "----------------------------
--------"

750 ? :RETURN
800 TM=TT*FT/T:P=220*SE*TM/1000
805 ? "POWER EXPENDED ";:V=P:GOSUB FTRC:
? V;" KWH"
810 ? "COOLING COST DUE TO WINDOW $";:V=
C*P/100:GOSUB FTRC:? V
820 ? :? "----------------------------
----------"

830 ?
1000 RETURN
2000 REM GET TRANSMITTANCE FOR SINGLE GL
AZED GLASS
2010 IF IN<0.87266 THEN TR=0.87:GOTO 210
0
2020 IF IN>1.2218 THEN 2050
2030 CI=(IN-0.8726639)*4.5:TR=0.16*COS(C
I)+0.68:GOTO 2100
2050 TR=3.0599-1.948*IN:IF TR<0 THEN TR=
0
2100 RETURN
4000 DATA 390,.142,385,.144,376,.156,360
,.18,350,.196,345,.205,344,.207
4002 DATA 351,.201,365,.177,378,.16,387,
.149,391,.142
5000 END
```

Example Run

```
LATITUDE(DEG)? 30
ANALYSIS DESIRED
 1) HEATING
 2) COOLING
? 2
 # SQ FT OF WINDOW FOR EVALUATION? 70


WINDOW TILT FROM HORZ, NORMAL=90? 90
WINDOW AZIMUTH(N=0,S=180), DEG? 90
AIRCONDITIONER TONS? 4
ENTER AMPS OF AIRCONDITIONER, IF NOT KNOWN ENTER 0? 30
COST FOR ELECTRICITY, CENTS/KWH?     6.55


     WINDOW HEATING ANALYSIS BY DIRECT SUNLIGHT
                 LATITUDE =  30 DEG

WINDOW ANGLE  90 DEG          WINDOW AZ=  90  DEG

MONTH=    5                   TOTAL =   27086.46 BTU/(SQ FT)
POWER EXPENDED  260.7 KWH
COOLING COST DUE TO WINDOW 17.07 DOLLARS
---------------------------------------------------------

MONTH=    6                   TOTAL =   27118.47 BTU/(SQ FT)
POWER EXPENDED  261.01 KWH
COOLING COST DUE TO WINDOW 17.09 DOLLARS
---------------------------------------------------------

MONTH=    7                   TOTAL =   26652.02 BTU/(SQ FT)
POWER EXPENDED  256.52 KWH
COOLING COST DUE TO WINDOW 16.8 DOLLARS
---------------------------------------------------------

MONTH=    8                   TOTAL =   26268.8  BTU/(SQ FT)
POWER EXPENDED  252.83 KWH
COOLING COST DUE TO WINDOW 16.56 DOLLARS
---------------------------------------------------------

MONTH=    9                   TOTAL =   25223.88 BTU/(SQ FT)
POWER EXPENDED  242.77 KWH
COOLING COST DUE TO WINDOW 15.9 DOLLARS
---------------------------------------------------------

MONTH=    10                  TOTAL =   23689.78 BTU/(SQ FT)
POWER EXPENDED  228.01 KWH
COOLING COST DUE TO WINDOW 14.93 DOLLARS
---------------------------------------------------------
---------------------------------------------------------
ANNUAL SAVINGS!

POWER EXPENDED  1501.87 KWH
COOLING COST DUE TO WINDOW 98.37 DOLLARS
---------------------------------------------------------
```

```
LATITUDE(DEG)? 30
ANALYSIS DESIRED
 1) HEATING
 2) COOLING
? 1
 # SQ FT OF WINDOW FOR EVALUATION? 70


WINDOW TILT FROM HORZ, NORMAL=90? 90
WINDOW AZIMUTH(N=0,S=180), DEG? 90

COST OF NATURAL GAS (CENTS/CU FT)? .37


     WINDOW HEATING ANALYSIS BY DIRECT SUNLIGHT
                 LATITUDE =  30 DEG

WINDOW ANGLE  90 DEG          WINDOW AZ=  90  DEG

MONTH=     11                 TOTAL =   19554.29 BTU/(SQ FT)
NATURAL GAS SAVED  12.44       100 CU FT
DOLLAR SAVINGS  4.6
--------------------------------------------------------------

MONTH=     12                 TOTAL =   17299.96 BTU/(SQ FT)
NATURAL GAS SAVED  11       100 CU FT
DOLLAR SAVINGS  4.07
--------------------------------------------------------------

MONTH=     1                  TOTAL =   17660.93 BTU/(SQ FT)
NATURAL GAS SAVED  11.23       100 CU FT
DOLLAR SAVINGS  4.15
--------------------------------------------------------------

MONTH=     2                  TOTAL =   21405.49 BTU/(SQ FT)
NATURAL GAS SAVED  13.62       100 CU FT
DOLLAR SAVINGS  5.04
--------------------------------------------------------------

MONTH=     3                  TOTAL =   25082.35 BTU/(SQ FT)
NATURAL GAS SAVED  15.96       100 CU FT
DOLLAR SAVINGS  5.9
--------------------------------------------------------------

MONTH=     4                  TOTAL =   26858.1  BTU/(SQ FT)
NATURAL GAS SAVED  17.09       100 CU FT
DOLLAR SAVINGS  6.32
--------------------------------------------------------------

----------------------------------------------------------
ANNUAL SAVINGS!

NATURAL GAS SAVED  81.36       100 CU FT
DOLLAR SAVINGS  30.1
--------------------------------------------------------------

LATITUDE(DEG)?
```

©

Subscript Heap Sort

Elizabeth Deal
Malvern, PA

This article describes a one-level-deep, ascending, alphanumeric subscript heap sort. It is written for the PET/CBM computer. It should work on systems that use Microsoft BASIC and permit arrays of character strings (Pet, Apple, OSI, Radio Shack).

Sort vs Subscript Sort

"Subscript sort" may be called *tag* sort, *pointer* sort, *index* sort or whatever you wish. The principle behind this type of ordering is that elements in a list are never moved and are not actually sorted. What gets rearranged into an ascending sequence are the subscripts of the array. The neat thing about this trick is that, as we are sorting records with several fields, we never need to move masses of data around. The corresponding fields are carried with the field that is being sorted. Subsequent to sorting, the access to the elements of the array is through the ordered list of subscripts.

For people with garbage collection problems, there is an additional advantage if they are sorting character strings. Since character strings do not have to move, time-consuming garbage collection during the sort will not need to occur. For further information on that subject consult Jim Butterfield's article in **COMPUTE!** #10, p. 96.

Sorting in BASIC takes considerable time no matter which of many available sorting methods is selected. I like heap sort because its performance is "even" no matter what the order of the original list is and the sorting time is almost linear relative to the list size. The algorithm itself is interesting, fun to study, and efficient on long lists. On short lists (N<25) there is, however, some time penalty as compared to several other sorting methods.

Don't Reinvent The Wheel

If you haven't done so already, you might want to look into a classic on the subject of sorting, merging, and general data management — Knuth, *The Art of Computer Programming*, vol. 3: Sorting and Searching, Addison and Wesley, 1973. The book looks mathematical and forbidding. But the appearance is deceptive, for there are no Greek letters in it and the sentences that look mathematical are, simply, ideas for the lines of a program. The illustrations are clear and the explanations are not at all complicated.

Book in hand, the algorithm is possible to follow if you practice the binary tree logic and the entire process with pencil and paper. It is then possible to modify the program from the book or the one from **COMPUTE!** #2 with some degree of assurance that it will successfully sort by subscripts. This program does just that.

... work on systems that use Microsoft BASIC and permit arrays of character strings ...

Suggestions On Data Management

The demonstration program consists primarily of sorting multifield records. The sort routine sorts field HV. The field type (alpha or numeric) is in HT, number of records to be sorted in H1. The resulting list of subscripts is placed in the SB array, their placement being determined by the comparative numeric or string value of the corresponding elements of the V or V$ array, depending on HT.

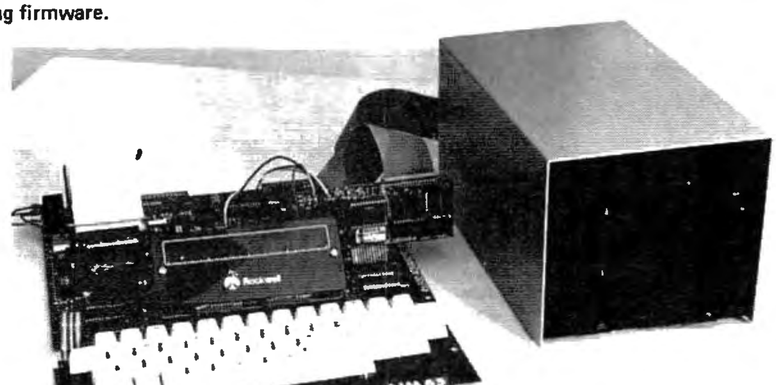
When sorting has been finished, in order to use the undisturbed, unsorted list, we ask for V$(f,r) as shown in lines 680-710. To use the list in sorted order we ask for V$(f,SB(r)) as coded in lines 630-661. In plain English, it means to print a value pointed to by the r-th subscript.

The program also contains some suggestions pertaining to general management of data. Take these nonsorting suggestions with a grain of salt. Vary them. These are some of the methods I use, find adequate and which fit most things I do on my PET. It does not mean that your arrangement of data or its parameters has to be like that. These ideas and the following details of the program are given mainly for people who are starting and don't know where to begin.

The program is originally set up (line 760) for NN = 20 estimated number of records and VV = 15 fields per record. You may change those estimates. The actual count of variables (NV) is performed in lines 770-810 while reading in data descriptors contained in the first DATA line. The actual count of records (KN) is done in lines 840-852 while reading in the six records from the remaining DATA lines.

There are two alphabetic and two numeric fields in each of the six records. The field type is stored in array TP. Type is 1 (one) for alpha (A) and 0 (zero) for numbers (N). TP is developed in lines 770-810 using the first DATA line. Since the ASCII collating sequence is irrelevant to sorting unaligned or non-integer or signed numeric values


commodore
CBM
computer


AIM 65

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**
4116-200 ns RAM (NEC)				**8 for 19**
Zero Insertion Force 24 pin Socket				2.00
6550 RAM (PET 8K)				12.70
S-100 Wire Wrap Socket				2.40

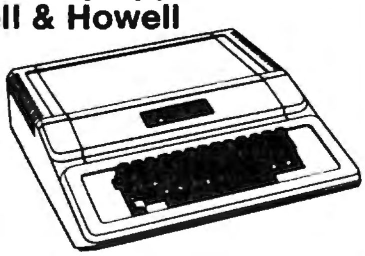
A P Products 15% OFF
A P Hobby-Blox 15% OFF

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 Chipaway - a breakout game using Atari paddles	7
VIC Moon Mission	7
VIC Micro-Coder - a machine language utility program	10
VIC Blue Meanies from Outer Space	9
VIC 21 - casino style Blackjack	9
VIC Slither/Super Slither	9
VIC Biorhythm	9
VIC Space Math	9
VIC Car Chase	9
VIC viCalc - sophisticated visible calculator	14
VIC Simon - audible and visual memory game	9
VIC 3K RAM (with 2 ROM Sockets)	65

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)	269
VIC 1515 Graphic Printer	(395)	315
Used CBM/PET Computers		CALL
8024-7 High Speed Printer	(1995)	1345

WE WILL MATCH ANY ADVERTISED PRICE

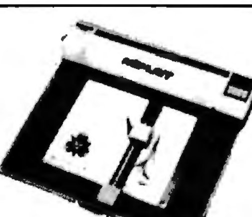
*** EDUCATIONAL DISCOUNTS ***

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.

ZENITH DATA SYSTEMS

Z19 Video Terminal (VT-52 compatible)	725
ZVM-121 Green Phosphor Monitor	115
Z89 with 48K	2150
Extra 16K RAM Board	115
Z47 Dual 8" Drive	2775

Synertek Systems

SYM-1 Microcomputer SALE	**205**
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

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

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

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.

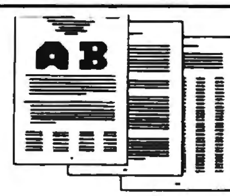
Available soon:

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

Microchess	17.00
Checker King	17.00
Gammon Gambler	17.00
Time Trek	13.45
Bridge Partner	13.45
Visicalc	155.00

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.

in character string form, these fields are not used in their string form. The values are placed in a one-dimensional work array V which has been set up in line 840. Should you be short of space for this extra array, you may change the program like this: omit all references to the V array by deleting its DIM in line 840 and lines 591 and 650. In the sort routine change all V(SB(*)) to VAL(V$(HV,SB(*))).

In any case, when HT is set to zero in line 600, the sort routine sorts numbers. This coding is in each second line of the subroutines which begin in lines 310, 330 and 350. The main routine (lines 560-711) handles these numbers as character strings, however, so that the output can look tidy while permitting a messy, unaligned input (sometimes useful in files for space-saving reasons).

Ranking fits in this sorting scheme automatically.

Two different output methods are used, depending on the type of variable. You'll see different coding for alpha fields (line 640) from that for numeric fields(lines 650-652). The output format is controlled by arrays V1 and V2 which specify the field width. In case of alpha variables, only V1 is required (see the first DATA line where A-12 and A-14 sequences specify alpha fields of 12 and 14 characters to be left-justified by line 530). In case of numbers, both V1 and V2 are needed (see line 450 and N-2-0 and N-4-3 sequences in line 870 which specify right-justified numeric output formats of xxx and xxxx.xxx respectively). The Butterfield formatting procedure from **COMPUTE!** #9 is used for printing numbers in a neat column.

Why not, you may ask, just read the values into a numeric array since that's what has to be used in sorting? There are several reasons. (1) This data might be an example of an existing disk file containing only character strings. (2) This might be a larger task requiring character by character data checking. Hence there is the need for input of character strings. Editing of data is a story outside the scope of this article, but it's a good idea to remember the issue every once in a while. (3) Unless you enjoy looking at unaligned columns of numbers the output ought to be formatted. Here, again, the easiest way is to work with character strings. Again, these are the methods I am comfortable with. Your opinions may differ and lead to a totally different approach.

Ranking

Finally, there exists a short ranking routine within the listing that might be useful to statistics people who would like to use this for nonparametric tests and suspect tied scores. Ranking fits in this sorting scheme automatically. Note that if there are *no* tied values then, by definition, at the end of sort *the subscripts are the ranks*, otherwise an average of ranks is given. Thus the rank routine is needed only when tied values are obvious or suspected. This routine creates an array of ranks (RV) while doing one extra pass through the list in subscript-sorted order. Needless to say, since you get a chance in this demo program to sort on any one of the four variables, the rank values are meaningless in some situations.

Figure 1.

```
* SORTED ON FIELD 4 *

CHARLOTTE   FARM           74  -93.000
FATHER FOX  VERMONT       100     .003
WILBUR      FARM            1     .488
MOUSE       TOOTLETOWN     84   33.700
TEMPLETON   FARM           98  647.000
TANKER      TOOTLETOWN     84  647.000

* UNSORTED, RANKED ON FIELD 4 *

TANKER        5.5
MOUSE         4.0
FATHER FOX    2.0
CHARLOTTE     1.0
WILBUR        3.0
TEMPLETON     5.5


* SORTED ON FIELD 3 *

WILBUR      FARM            1     .488
CHARLOTTE   FARM           74  -93.000
TANKER      TOOTLETOWN     84  647.000
MOUSE       TOOTLETOWN     84   33.700
TEMPLETON   FARM           98  647.000
FATHER FOX  VERMONT       100     .003

* UNSORTED, RANKED ON FIELD 3 *

TANKER        3.5
MOUSE         3.5
FATHER FOX    6.0
CHARLOTTE     2.0
WILBUR        1.0
TEMPLETON     5.0
```

```
100 REM-----------------------------

110 REM SUBSCRIPT HEAP SORT
140 REM ELIZABETH DEAL
150 REM-----------------------------

160 GOSUB76Ø:GOSUB56Ø:END
170 REM-----------------------------

180 REM SORT SUBSCRIPTS OF FIELD HV

181 REM FIELD TYPE HT (Ø=N 1=A)
182 REM NUMBER OF RECORDS H1
183 REM PLACE SUBSCRIPTS IN SB ARRA
    Y
190 IFH1<2THENPRINT"NEED 2+":END
200 H2=INT(H1/2)+1:HA=1:HZ=Ø
210 IFH2>HATHENH2=H2-HA:H8=H2:GOSUB
    31Ø:HR=SB(H2):GOSUB26Ø:GOT
    O21Ø
220 H8=H1:GOSUB31Ø:HR=SB(H1):SB(H1)
    =SB(HA):H1=H1-HA
230 IFH1>=HATHENGOSUB26Ø:GOTO21Ø
240 SB(H3)=HR:RETURN
250 :
260 H4=H2
270 H3=H4:H4=H4+H4:IFH4>H1THENSB(H3
    )=HR:RETURN
280 IFH4<H1THENGOSUB33Ø:IFHLTHENH4=
    H4+HA
290 GOSUB35Ø:IFHGTHENSB(H3)=HR:RETU
    RN
300 SB(H3)=SB(H4):GOTO27Ø
310 IFHTTHENH5$=V$(HV,SB(H8)):RETUR
    N
320 H5=V(SB(H8)):RETURN
330 HL=HZ:IFHTTHENHL=-(V$(HV,SB(H4)
    )<V$(HV,SB(H4+HA))):RETURN

340 HL=-(V(SB(H4))<V(SB(H4+HA))):RE
    TURN
350 HG=HZ:IFHTTHENHG=-(H5$>=V$(HV,S
    B(H4))):RETURN
360 HG=-(H5>=V(SB(H4))):RETURN
370 REM-----------------------------

380 REM RANK FROM SUBSCRIPTS
390 RA=1:RB=2:RC=Ø:FORR1=1TOR3:RS=R
    1:RQ=RC:RF=RC
400 IFV$(RR,SB(R1))=V$(RR,SB(R1+RA)
    )THENRQ=RQ+RA:R1=R1+RA:RF=
    RA:GOTO4ØØ
410 FORR2=RSTOR1:RV(SB(R2))=RS+RQ*R
    F/RB:NEXTR2,R1:RETURN
420 REM-----------------------------

430 REM 'USING' ARRANGE IN COLUMNS
440 REM J. BUTTERFIELD
450 REM V IS VALUE; V1.V2 PRINTS
460 V4=INT(V*1Ø^V2+.5):REM ROUNDED
470 V$=RIGHT$("          "+STR$(V4),V
    1+V2+1):IFV2<1GOTO5ØØ
480 FORV5=V1+2TOV1+V2+1:IFASC(MID$(
    V$,V5))<48THENNEXTV5
490 V6=V5-V1-1:V$=MID$(V$,V6,V1+1)+
    LEFT$(".ØØØØØØ",V6)+MID$(V
    $,V5)
500 IFASC(V$)>47THENV$=LEFT$("*****
    *****",V1+V2+2+(V2=Ø))
510 RETURN
520 REM-----------------------------

530 PRINTLEFT$(VS$+B$,V1);:RETURN
     :REM A
540 GOSUB45Ø:PRINTV$;:RETURN
     :REM N
550 REM-----------------------------

560 PRINT:PRINT"SORT FIELD 1 -"NV"O
    R X"
561 INPUTF$:IFF$="X"THENRETURN
570 SI=VAL(F$):IFSI<1ORSI>NVTHENPRI
    NT"???":GOTO56Ø
590 FORI=1TOKN:SB(I)=I:NEXT  :REM I
    NIT SUBSCRIPTS
591 IFTP(SI)=ØTHENFORI=1TOKN:V(I)=V
    AL(V$(SI,I)):NEXTI  :REM C
    ONVERT V$ TO V
592 :
600 HV=SI:HT=TP(HV):H1=KN:GOSUB19Ø
     :REM SORT
610 RR=SI:R3=KN:GOSUB39Ø
     :REM RANK
611 :
620 PRINT:PRINT"* SORTED ON FIELD"S
    I"*":PRINT
630 FORI=1TOKN
631 FORJ=1TONV:VS$="":V$=""
640 IFTP(J)THENVS$=V$(J,SB(I)):V1=V
    1(J):GOSUB53Ø:GOTO66Ø  :RE
    M A
650 IFJ=SITHENV=V(SB(I)):GOTO652 :R
    EM SORTED N FIELD
651 V=VAL(V$(J,SB(I)))          :R
    EM OTHER N FIELDS
652 V1=V1(J):V2=V2(J):GOSUB54Ø
660 NEXTJ:PRINT
661 NEXTI
662 :
670 PRINT:PRINT"* UNSORTED, RANKED
    ON FIELD"SI"*":PRINT
680 FORI=1TOKN:VS$=""
690 VS$=V$(1,I):V1=V1(1):GOSUB53Ø
     :REM A
```

```
700 V=RV(I):V1=2:V2=1:GOSUB540
     :REM N
710 PRINT:NEXTI
711 GOTO560
720 REM----------------------------

730 REM INITIALIZE
740 REM READ DATA DESCRIPTORS FOR O
    UTPUT
750 REM READ IN KN RECORDS OF NV FI
    ELDS IN EACH RECORD
760 NN=20:KN=0:VV=15:NV=0:DIM TP(VV
    ):B$="                         "
770 READTP$:IFTP$="X"GOTO830
780 NV=NV+1
790 IFTP$="A"THENTP(NV)=1:READV1(NV
    ):GOTO770
800 IFTP$="N"THENTP(NV)=0:READV1(NV
    ),V2(NV):GOTO770
810 PRINT"BAD DATA DESCR":LIST870
820 :
830 DIM V$(NV,NN),SB(NN),RV(NN)
831 FORI=1TONN
840 READE$:IFLEFT$(E$,4)="XXXX"THEN
    KN=I-1:DIM V(KN):RETURN
850 V$(1,I)=E$
851 IFNV>1THENFORJ=2TONV:READV$(J,I
    ):NEXTJ
852 NEXTI
860 :
870 DATA A,12, A,14, N,3,0, N,4,3,
    X
880 :
890 DATA TANKER,TOOTLETOWN,84,647
900 DATA MOUSE,TOOTLETOWN,84,33.7
910 DATA FATHER FOX,VERMONT,100,.00
    3
920 DATA CHARLOTTE,FARM,74,-93
930 DATA WILBUR,FARM,1,.4876
940 DATA TEMPLETON,FARM,98,647
950 DATAXXXXXXXXXXXXXXXXXXXXXXXXXXX
    XXX
```

Review:

SYZYGY RS-232 Condition Testers

Sanford I. Gossman
San Rafael, CA

S Y Z Y G Y (pronounced "siz-a-gee") is a small, and relatively new, company in Covina California (256 West San Bernardino Road; 91723). They produce a line of RS-232 testing devices which includes two LED devices that monitor the condition of the connections that interface RS-232 devices, or ports. The quality of their products demonstrates what can be done when a manufacturer limits himself to one type of product and does a job right.

S Y Z Y G Y makes two RS-232 line-condition testers. Each are "pocket size," measuring approximately 3x2x½ inches. Each has a male RS-232 connector on one end, and a female connector on the other. Each has eight LED's wired so as to monitor the seven most commonly used signals (pins 2, 3, 4, 5, 6, 8, & 20: TD, RD, RTS, CTS, DSR, DCD, DTR). The eighth LED, labeled "TEST," can be jumpered to any pin not having a light wired to it, so that its status can be monitored.

No Batteries Needed

The devices are powered by the RS-232 signals, so no batteries are required. Current-limiting circuitry assures a constant current over the voltage range permitted by the RS-232 standard, and provides a simple "go, no-go" indication.

The least expensive of the two products is called the "Test Set" and sells for $89. Each of the 25 pins are wired through, and a single 26-pin header provides a means to connect one pin to another.

The "top-of-the-line" model is designated the RS-232 Patch Set. It sells for $111. It is the model I have been using for the past two months.

The difference between the two is that the Patch Set has *none* of the connectors feeding through. Instead, there is a 26-pin header on each end. The unit comes with a generous supply of single and double jumpers. The wires are used to connect the pins manually: either to their corresponding number, or to another pin, or pins, for testing. The advantage of this scheme is that it permits you to easily break the connection of a pin, by merely pulling a jumper. In this way, you can determine what signal is present from each device separately.

Easy Status Checks

I am in the process of writing a series of reviews of RS-232 modems. Most of the equipment represents a new product, and there is no software provided that will permit operation in my equipment environment. So, I have been busy writing software.

The Patch Kit has proven invaluable. With the documentation from some manufacturers being sketchy, at best, it has been imperative for me to know the status of each of several key connections. With the Patch Kit it was easy. Without it I would have been switching a logic probe between pins almost endlessly.

When your installation is complete, either the Test Set or Patch Kit can be left attached.

The ability to segregate signals helped me greatly when I was having difficulty reading a status signal from a modem, through software. Theory said that what I was doing was correct, but I was both baffled and frustrated. Because I was able to determine the status of the problem signal line at "each end" of the connecting cable, the source of the problem was discovered easily.

By disconnecting the corresponding jumper and "reading" the signal as it appeared (both coming from the modem and going to the computer), I was able to discover that portion of a chip inside my computer was inoperative. The condition has previously gone undetected because the line served by the pin had not been used by other equipment previously attached to the port. I'm sure that, had I not had the tester, I would have spent several hours trying to correct what I first believed was a software problem.

First-rate Construction

The construction of the product is "first-rate." Everything is soldered, all edges are smooth, and only high-quality materials are used. The LED's are mounted behind holes in the circuit board that makes up the "chassis" of the device. The holes are silvered on the inside. This treatment enhances the brightness of the glow, and makes it easy to see from virtually any angle.

A placard is conveniently attached to the rear, and provides a wealth of information constantly needed during a configuration or trouble shooting project. A cover retains the jumper wires when the unit is not in use.

The Patch Set has the function of similar products selling for nearly three times as much. Accordingly, it qualified in my book for "best-buy" status.

I recommend that such a device be purchased and used to make a thorough analysis of the status of each pin of each RS-232 device of your system. Then, when a problem develops, you will be able to quickly determine the cause.

When your installation is complete, either the Test Set or Patch Kit can be left attached. Its LED's will give you assurance when you need it and pinpoint a problem if, and when, one develops.

The company also supplies a series of color-coded (sex) adapters, and null-modem configurations, measuring 2x1¾x½ inches. They are priced at $25.00 each. The Anything Cable is a seven foot, 25-conductor, ribbon cable with both a male and female connector on each end. You can do just about "anything" with it, for a cost of $75.00. ©

Introducing:

Introducing LISP. The language that can think for itself.

With SmartWare, your microcomputer possesses intelligence. Solves problems like never before. Actually can learn from its mistakes. And educates itself in much the same way your brain operates. It's a new concept in the way information is handled in microcomputers.

We appropriately called our version of LISP, SmartWare. Because it's as limitless as the human imagination, mind, memory.

An advanced, high-level language, LISP was first developed for use in artificial intelligence on large computers. And now, LISP is available from Datasoft for use on the Apple II and Atari Computers.

Store multiple programs in memory. It can take it. Program other languages into LISP. It's no problem. Because LISP offers greater capacity and power. And, it's a faster, more streamlined language. For example, just 2 lines in LISP could equal hours of BASIC programming.

The fact: Relational data base capabilities. User and program definability. Pattern-directed invocation language. Uses syntax and data structures upon which esoteric applications may be implemented. Remembers data along with "relationships affecting it." Offers REAL power to micros.

At MIT, they say LISP is the language of the future.

At datasoft, we say, why wait. Ask your local computer dealer for SmartWare. Right now! And turn your computer on to thinking.

SmartWare by Datasoft Inc. T.M.
COMPUTER SOFTWARE

19519 Business Center Dr., Northridge, CA 91324
(213)701-5161

InterLISP/65 for ATARI 48K systems 149.95 & APP-L-1SP 1.7 for Apple II 48K systems 124.95.

Unscramble

Henry Kong
Singapore

The main routine is listed early between lines 80 and 90 for faster execution. These lines select the data randomly and break each word into individual letters. The sorting algorithm rearranges the letters in alphabetical order, providing an "unscramble."

With the main routine securely tucked in, we start the gaming section beginning with line 100. Lines 100 to 120 deal with the questions and responses. Both the number of new words and the total attempts are tallied by the counters Q and C. The counter V keeps score of the correct guesses. Two chances are given in this program before you are out.

The following lines deal with the continuation of the game and/or final result. The final touch is to add in DATA statements. This can be done anywhere, usually at the end of the program. 500 DATA"EAR", "TABLE", "KITCHEN", "MOTHER", "COMPUTER". Since line 80 multiplies the RND by 50, this program needs 50 DATA statements. This can be increased or decreased according to taste. If you have unusually long words such as "misunderstanding" etc. you will need to add:

11 DIMW$(25)

or a (Bad subscript) will interrupt your program.

As it is, the game UNSCRAMBLE works pretty well and provides an enduring challenge for the whole family. It allows you to choose the number of rounds, stop whenever you wish, and gives you a percentage score at the end of the game. The two chances that it allows help you to catch up on your score should you be unsuccessful in your first few attempts.

Adding the last few lines will keep the program running if you want another game (to better your score, perhaps?) or if another player wants to join in.

Program 1. Atari Version

```
10 REM UNSCRAMBLE-ATARI
15 DIM NA$(30),O$(25),W$(25),P$(1),A$(25
)
20 PRINT CHR$(125)
30 PRINT "THE GAME OF"
40 PRINT :PRINT "   ABCELMNRSU"
50 PRINT :FOR X=1 TO 1000:NEXT X
52 PRINT "...UNSCRAMBLE!"
75 PRINT :PRINT "NAME, PLEASE";:INPUT NA
$:GOTO 100
80 R=INT(RND(1)*12)+1:REM CHANGE '12' TO
 NUMBER OF WORDS IN DATA LINES 500-
81 FOR K=1 TO R:READ O$
82 NEXT K:RESTORE
84 LO=LEN(O$):W$=O$
85 FOR LL=2 TO LO:LI=LL-1:P$=W$(LL,LL)
86 IF P$>W$(LI,LI) THEN 90
88 W$(LI+1,LI+1)=W$(LI,LI):LI=LI-1:IF LI
>0 THEN 86
90 W$(LI+1,LI+1)=P$:NEXT LL:RETURN
100 PRINT CHR$(125)
102 Q=Q+1:PRINT "UNSCRAMBLE WORD # ";Q:G
OSUB 80
105 C=C+1:PRINT "ATTEMPT #";C
110 PRINT :PRINT "UNSCRAMBLE THIS WORD..
.":PRINT
114 PRINT W$
120 PRINT "ANSWER";:INPUT A$:IF A$=O$ TH
EN 200
125 PRINT :IF T=1 THEN PRINT "SORRY, THE
 WORD IS":PRINT :PRINT O$:GOTO 300
130 PRINT :PRINT NA$;", ONE LAST TRY":T=
1:GOTO 105
200 FOR X=1 TO 6:PRINT :NEXT X:V=V+1
250 PRINT :PRINT "CONGRATULATIONS, YOU W
IN"
300 PRINT :PRINT "ANOTHER WORD";:INPUT P
$:IF P$="Y" THEN T=0:GOTO 100
310 PRINT CHR$(125)
315 PRINT "OUT OF ";Q;" UNSCRAMBLES"
320 PRINT :PRINT NA$;" HAS ";V;" CORRECT
,"
325 PRINT :PRINT "USING ";C;" ATTEMPTS!"

330 P=INT((V/C)*100)
335 PRINT :PRINT "YOUR SCORE IS ";P
400 PRINT :PRINT "ANOTHER GAME";:INPUT P
$
405 IF P$="N" THEN END
410 V=0:Q=0:C=0:GOTO 20
500 DATA EAR,TABLE,KITCHEN,MOTHER,COMPUT
ER
510 DATA FACE,AUTOMOBILE,RUBBERBAND,DIAM
OND
520 DATA VIBRATE,TENACIOUS,MONSTER,ESCAP
E
```

Program 2. Microsoft Version (PET, Apple, etc.)

```
10 REM UNSCRABBLE
20 FOR X=1 TO 25:NEXT X
30 PRINT "THE GAME OF"
```

SOFTWARE HARDWARE FOR THE VIC 20®

SOFTWARE FOR THE VIC 20

TAPE CASSETTES

- **VicTerm A** — **$19.95**
  Be smart! Operate your VIC 20 as a dumb terminal.
- **viCalc I** — **$14.95**
  First in a series of useful calculator programs. 10 memorys with arithmetic, 4 stacked data registers. Math functions plus compound interest tables.
- **VPM** — **$14.95**
  Securities Portfolio Management. Important records on tape for 25 securities.
- **AMOK** — **$18.95**
  The halls of AMOK are populated by robots that follow one instruction ... get the intruder. That's you. To save yourself, you must be quick on the draw and fast on your feet.
- **SIMON** — **$9.95**
  Test dexterity and memory by repeating the flashing colors and tones—different every time.
- **PAC IT IN** — **$18.95**
  Fast action color game. Two players can ZAP ROBOTS before they "**PAC IT IN**."
- **SUPER ADDITION OR SUBTRACTION** — **each $9.95**
  Blackboard feature gives useful math practice with interest enhanced by color. Correct steps shown with carries.

HARDWARE FOR THE VIC 20

- **UMI RS232 COMMUNICATIONS INTERFACE** — **$49.95**
  For printers and telephone communications.
- **UMI 3K MEMORY EXPANSION** — **$79.95**
  With addressable and switchable ROM slots for ROM programs to 16K.
- **UMI 8K RAM EXPANDER** — **$99.95**
  Provides 11775 Bytes (characters) of user memory.
- **UMI EXPANSION CHASSIS** — **Price to be announced**
  For additional memory or program cartridges.

SOON TO BE AVAILABLE

SOFTWARE CARTRIDGES

- **SPIDERS OF MARS**
- **SATELLITES AND METEORITES**
- **ROBOT BLASTERS**
- **ASTRO TRANSPORTERS**
- **3D INVADERS**

ORIGINAL! EXCITING! UNIQUE!

DEALER INQUIRIES INVITED

Clip Coupon for **FREE CATALOG**
Catalog available in Spanish (Version Disponible en Español)
MasterCard/VISA Accepted

United Microware Industries Incorporated
3431 H Pomona Boulevard
Pomona, Calif. 91768
Phone (714) 594-1351

VIC 20 is a Registered Trademark of
Commodore Business Machines

united microware industries inc.
3431 H Pomona Blvd.
Pomona, CA. 91768

Please send me my FREE CATALOG describing your Hardware and Software Products.

NAME ______________________

ADDRESS ______________________

CITY ______________ STATE ________

ZIP ________ COMPANY ______________

```
40 PRINT:PRINT "   ABCELMNRSU"
50 PRINT:FOR X=1 TO 2000:NEXT X
52 PRINT "...UNSCRAMBLE!"
75 PRINT:INPUT "NAME, PLEASE";NA$:GOT~
   ~O 100
80 R=INT(RND(1)*12)+1:REM CHANGE '12'~
   ~ TO NUMBER OF WORDS IN DATA LINES~
   ~ 500-
81 FOR K=1 TO R:READ O$
82 NEXT K:RESTORE
84 LO=LEN(O$):FOR LI=1 TO LO:W$(LI)=M~
   ~ID$(O$,LI,1):NEXT LI
85 FOR LL=2 TO LO:LI=LL-1:P$=W$(LL)
86 IF P$>W$(LI) THEN 90
88 W$(LI+1)=W$(LI):LI=LI-1:IF L>0 THE~
   ~N  86
90 W$(LI+1)=P$:NEXT LL:RETURN
100 FOR X=1 TO 25:PRINT:NEXT X
102 Q=Q+1:PRINT "UNSCRAMBLE WORD # ";~
   ~Q:GOSUB 80
105 C=C+1:PRINT:PRINT "ATTEMPT # ";C
110 PRINT:PRINT "UNSCRAMBLE THIS WORD~
   ~...":PRINT
114 FOR LI=1 TO LO:PRINT W$(LI);:NEXT~
   ~ LI:PRINT:PRINT
120 INPUT "ANSWER";A$:IF A$=O$ THEN 2~
   ~00
125 PRINT:IF T=1 THEN PRINT "SORRY, T~
   ~HE WORD IS":PRINT:PRINT O$:GOTO 3~
   ~00
130 PRINT:PRINT NA$;", ONE LAST TRY":~
   ~T=1:GOTO 105
200 FOR X=1 TO 6:PRINT:NEXT X:V=V+1
250 PRINT:PRINT "CONGRATULATIONS, YOU~
   ~ WIN"
300 PRINT:INPUT "ANOTHER WORD";Y$:IF ~
   ~LEFT$(Y$,1)="Y" THEN T=0:GOTO 100~
   ~
310 FOR X=1 TO 25:PRINT:NEXT X
315 PRINT "OUT OF ";Q;" UNSCRAMBLES"
320 PRINT:PRINT NA$;" HAS ";V;" CORRE~
   ~CT,"
325 PRINT:PRINT "USING ";C;" ATTEMPTS~
   ~!"
330 P=INT((V/C)*100)
335 PRINT:PRINT "YOUR SCORE IS ";P
400 PRINT:INPUT "ANOTHER GAME";Y$
405 IF LEFT$(Y$,1)="N" THEN END
410 V=0:Q=0:C=0:GOTO 20
500 DATA "EAR","TABLE","KITCHEN","MOT~
   ~HER","COMPUTER"
510 DATA "FACE","AUTOMOBILE","RUBBERB~
   ~AND","DIAMOND"
520 DATA "VIBRATE","TENACIOUS","MONST~
   ~ER","ESCAPE"
READY.
```

©

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

VISA

12-ADVENTURE PACK- (#1-4) $112

14-THE WARRIOR- (#7-8) $64

13-SPACEOUT- (#5-6) $58

15-ARCADE- (#9-11) $60

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**

Editor's Note: This program (versions here for PET Microsoft and Atari) can be the basis for many excellent games. When you come up with something interesting — send it in to **COMPUTE!** *— RTM*

Maze Generator

Charles Bond
Sunnyvale, CA

Here's a remarkably short algorithm which produces random mazes of any desired size directly on your CRT screen. The program will create mazes on any microcomputer which allows memory mapped graphics. Details are provided for directly applying it to the COMMODORE PET and the Atari 400/800 computers. A typical maze generated by the program is shown in Figure 2.

To understand how it works, refer to the flowchart in Figure 1 and the program listing. The following explanation should clarify the details.

The Background Field

The algorithm operates on a background field which must be generated on the screen prior to line number 200 in Program 1. The field must consist of an odd number of horizontal rows, each containing an odd number of cells: a rectangular array. It's convenient to think of the field as a two dimensional array with the upper left corner having coordinates 'X' = 0 and 'Y' = 0, where 'X' is the horizontal direction and 'Y' is vertical. No coordinates are used to identify absolute locations by the program, but the concept is useful in configuring the field.

Given that the upper left cell of the field has coordinates 0,0 then the terminal coordinates both horizontally and vertically must be even numbers. In addition, the background field must be surrounded on all sides by memory cells whose contents are different from the number used to identify the field. That is, if the field consists of reversed (or inverse video) spaces, then the number corresponding to that character must not be visually adjacent to the field.

This could happen inadvertently if the screen RAM and system ROM have contiguous addresses. A sufficient precaution is to avoid covering the entire screen with field. Leave at least one space at the beginning or end of each line and, in general, leave the uppermost and lowermost lines on the screen blank.

The Maze Generator

The creation of the maze begins by placing a special marker in a suitable starting square. The program here always begins at the square just inside the upper left cell of the previously drawn field. (Note that with our coordinate scheme this would be cell 1,1). Any cell with odd numbered coordinates would work, however, as long as it is internal to the field.

Next, a random direction is chosen by invoking the random number generator in your machine and producing an integer from 0 to 3. This integer, with the aid of a short table, determines a direction and a corresponding cell just two steps away from the current cell. This new cell is examined (PEEKed) to see if it is part of the field. If it is, the direction integer is put there as a marker and the barrier between it and the current cell is erased.

In addition, the pointer to the current cell is moved to point to the new one. This process is repeated until the new cell fails the test; i.e., it is not a field cell. When this happens, the direction vector is rotated 90 degrees and the test is repeated. Thus, the path carved out of the field will continue until a "dead-end" is reached.

A dead-end, incidentally, could occur in as few as five steps. When it does occur, we can make use of the markers which were dropped along the way "Hansel and Gretel" style. These can be checked to determine which direction we came from, so that we can back up and look for untrodden paths. So long as none can be found, the program will back up, one step at a time, erasing the markers as it goes. When a new direction can be taken, the pointer is set off in that direction, and the process continues as before.

Ultimately, the pointer will return to the start, a condition which is detected by the recovery of the special starting (now "ending") marker. This cell is then blanked and the program is done, leaving the pointer as it was at the start.

The Program

Program 1 contains the complete program as implemented on the PET computer, but it is applicable to other machines. The direction table set up in lines 100 and 110 converts an integer to an address offset. In this case (40 column screen), we wish to be able to step two cells to the right, up, left, or down. The memory addresses of these cells differ from that of the current one by 2, –80, –2, and 80, respectively. For computers with 64 column displays, the 80's should be replaced by 128's; for the Atari no change is needed.

Line 120 contains machine-dependent variables. 'SC' is the memory address of the start of screen RAM. For the Atari use the following:

**120 WL=128:HL=0:SC=PEEK(88)+256*PEEK(89):
A=SC+43:REM THESE VALUES FOR ATARI**

Lines 130-160 establish the background field on the screen. For the PET we chose 23 rows of 39 cells each. The Atari, with default tab settings, will require a slightly smaller field. [*See Program 3 — Ed.*]

The rest of the program draws the maze, as previously explained. Line 310 is simply a convenient stopping point which prevents the screen from scrolling.

It may not be immediately obvious that this algorithm always produces a maze with only one non-trivial path between any two points, or that the maze will always be completely filled, but this can be proved. While the proofs will not be provided here, math buffs may find it interesting that for a maze of any size there will be exactly:

$$\frac{(H-1)(V-1)}{2}-1 \quad \text{empty cells in the completed maze,}$$

where H is the number of cells in each field row and V is the number of rows.

An interesting feature of this algorithm is that it works equally well in certain types of non-rectangular fields. U-shaped fields or fields with holes in them are quite suitable — as long as certain restrictions are observed. Just make sure that the coordinates of the upper left and lower right cells of any cut out area are pairs of odd numbers. Also, if there is a single row of field cells between any cut out areas and the outside of the original field, it may be removed. See Figure 3.

The Mouse

With slight modifications the Maze Generator can become an artificial "mouse." Programs 2 and 4 show a routine which can be appended to the Maze Generator and which create a mouse which roams the maze endlessly. The mouse adheres to a "left-hand rule" when a choice of directions is possible. That is, when it is confronted with a branch-point, it will move off to the left, if possible. Otherwise it will go forward. When no choice is available it will turn around.

Program 1: Microsoft Version

```
10 REM ******************
20 REM *                *
30 REM * MAZE GENERATOR *
40 REM * ============== *
50 REM *      1981      *
60 REM *                *
70 REM *   BY C. BOND   *
80 REM *                *
90 REM ******************
100 DIM A(3):REM SET UP DIRECTION T
    ABLE
110 A(0)=2:A(1)=-80:A(2)=-2:A(3)=80
    :REM THESE VALUES FOR 40 C
    OLUMN SCREEN
120 WL=160:HL=32:SC=32768:A=SC+81:R
    EM THESE VALUES FOR COMMOD
    ORE PET
130 PRINT "{CLEAR}":REM CLEAR SCREE
    N AND GENERATE MAZE BACKGR
    OUND FIELD
140 FOR I=1 TO 23
150 PRINT "{REV}
                                "
160 NEXT I
200 REM GENERATE THE MAZE!
210 POKE A,4
220 J=INT(RND(1)*4):X=J
230 B=A+A(J):IF PEEK(B)=WL THEN POK
    E B,J:POKE A+A(J)/2,HL:A=B
    :GOTO 220
240 J=(J+1)*-(J<3):IF J<>X THEN 230

250 J=PEEK(A):POKE A,HL:IF J<4 THEN
     A=A-A(J):GOTO 220
300 REM MAZE IS DONE! WAIT FOR KEYP
    USH
310 GET C$:IF C$="" THEN 310
```

Program 2: Microsoft Version

```
1000 REM MAZE TRAVERSAL ALGORITHM
1010 POKE A,81:J=2
1020 B=A+A(J)/2: IF PEEK(B)=HL THEN
    POKE B,81: POKE A,HL: A=B:
     J=(J+2)+4*(J>1)
1030 J=(J-1)-4*(J=0): GOTO 1020
```

Program 3: Atari Version

```
10 REM ******************
20 REM *                *
30 REM * MAZE GENERATOR *
40 REM * ============== *
50 REM *      1981      *
60 REM *                *
70 REM *   BY C. BOND   *
80 REM *                *
90 REM ******************
100 DIM A(3):REM SET UP DIRECTION TABLE
110 A(0)=2:A(1)=-80:A(2)=-2:A(3)=80:REM
THESE VALUES FOR 40 COLUMN SCREEN
120 WL=128:HL=0:SC=PEEK(88)+256*PEEK(89)
:A=SC+43:REM THESE VALUES FOR ATARI
130 PRINT "{CLEAR}";:POKE 752,1
140 FOR I=1 TO 23
150 PRINT "|
          |"
```

HUNTINGTON COMPUTING

ONE OF THE WORLD'S LARGEST INVENTORIES

GAMES

Hi-Res Soccer	$29.95	now	$25.39
Apple*-Oids	$29.95	now	$25.39
Wurst of Huntington Computing			**$19.99**
Gobbler	$24.95	now	$21.19
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
Star Mines	$29.95	now	$25.39
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
Computer Acquire	$20.00	now	$16.99
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
3-D Skiing	$24.95	now	$21.19
Creature Venture	$24.95	now	$21.19
Galaxy Space War I	$39.95	now	$33.99
Hodge Podge	$23.95	now	$21.29
Meteoroids in Space	$19.95	now	$16.99
Dragon Fire	$49.95	now	$42.49
Pool 1.5	$34.95	now	$29.69

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 Wolfenstein	$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**
UT-100 Emulator			**$65.99**
Space Quark	$29.95	now	$25.39
Beneath Apple DOS (book)	$19.95	now	$16.89
Dinosaurs (cass.)	$24.00	now	$20.39
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
Fantysland 2U41	$59.95	now	$50.99
Torpedo Terror	$24.95	now	$21.19
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
Apple* Pie (All Versions)	$129.95	now	$99.99
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

Welcome to the world of Huntington Computing and the first Softlights column.

To celebrate our first column we are offering **Space Eggs** and **Apple Panic** for only **$19.99** each (list $29.95). You must mention this ad to get the special price. Shipping (usually UPS) for software is $2.00, no matter how large the order. UPS Blue Label is $3.50. Foreign shipping (except for Canada and Mexico) and hardware shipping are extra.

Huntington Computing started out a year-and-a-half ago as a mom and pop outfit. We now employ close to fifty people. We think we have the largest selection of software for the Apple* in the world.

Check the reader service card for a free catalog of software for the Apple*. We also are now carrying a large selection of software for the Atari*, TRS-80* and Pet* but don't have a catalog yet.

We have fast service, the best guarantee in the business and friendly personnel.

Our store hours are 7 a.m. to 9 p.m. (P.S.T.) on weekdays and 10 a.m. to 5:30 p.m. on weekends and holidays (except Christmas). Usually, however, we're here much beyond those hours and can even be reached in the middle of the night occasionally. We do not employ an answering service or an outside ordering service.

We want your business and will do what we can to get it and keep it.

Got a pre-schooler? Our two-year-old loves **Hodge Podge** by Dynacomp – only **$20.99** (list $22.99). Our vote for favorite arcade-type game is **Sneakers.** A lot of people, however, tell us they get addicted to Apple Panic.

Business systems? We took a handful to our C.P.A. and he chose the **Continental General Ledger** for us - **$212.49** (list $250.00).

Even if you don't see the software you want listed here or in our catalog, chances are we already have it in stock or can get it for you at a discount. We stock more than 1,000 different programs and add new ones every day.

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
Visidex **Special**	$199.95	now	$139.00
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
Creativity Tool Box	$44.95	now	$38.19
Home Money Minder	$34.95	now	$29.69

HARDWARE

D.C. Hayes Micromodem	$375.00	now	$299.00
D.C. Hayes Smart Modem	$279.00	now	$249.00
MX-80 Printer/Cable/Card	$755.00	now	$575.00
MX-80 Replacement Head	$39.00	now	$34.99
MX-100FT with Graftrax	$995.00	now	$799.00
MX-80FT with Graftrax	$795.00	now	$669.00
460G Paper Tiger	$1094.00	now	$961.99
560G Paper Tiger	$1394.00	now	1225.00
Tiger Trax	$16.95	now	$15.25
Mark IV DOS Boot Switch	$24.95	now	$21.19
Z-80 Softcard	$395.00	now	$299.00
Expansion Chassis	$795.00	now	$675.00
Mt. Comp. Music System	$545.00	now	$479.00
NEC 12" Green on Black	$260.00	now	$209.00
Videx 80-col.	$350.00	now	$299.00
Videx Switchplate	$19.95	now	$17.99
Andromeda 16K Ram Expansion	$199.00	now	$179.00
Microsoft 16K Ram Expansion	$199.00	now	$179.00
Keyboard Plus	$119.00	now	$99.99
TG Joystick	$59.95	now	$50.89
TG Game Paddles	$39.95	now	$33.89
Paymar LCA Rev. 7			**$29.69**

MISCELLANEOUS

Memorex Disks			10 for $24.99
Verbatim Datalife-plain w/hubs			10 for $27.99
Dysans			10 for $49.99
Flipsort Box			$29.00
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

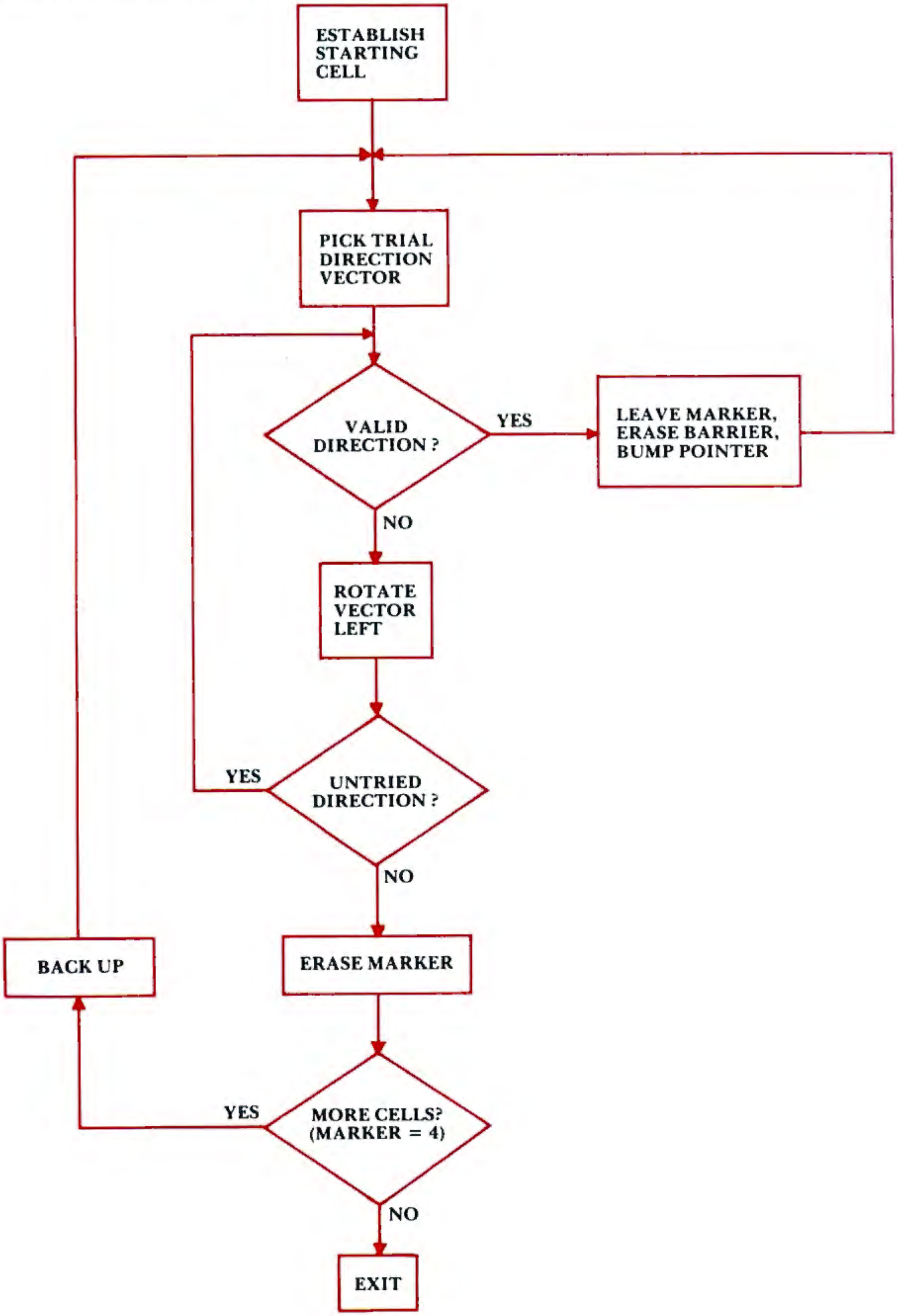
Order by Phone 800-344-4111
In California (209) 992-5411

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. Maze Generator Flow Chart



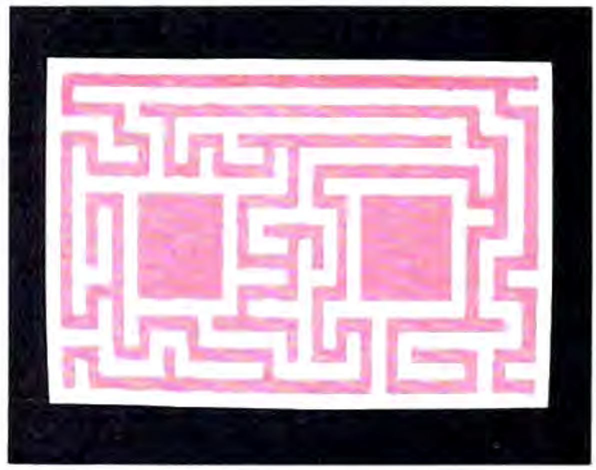
```
160 NEXT I
200 REM GENERATE THE MAZE!
210 POKE A,5
220 J=INT(RND(0)*4):X=J
230 B=A+A(J):IF PEEK(B)=WL THEN POKE B,J
+1:POKE A+A(J)/2,HL:A=B:GOTO 220
240 J=(J+1)*(J<3):IF J<>X THEN 230
250 J=PEEK(A):POKE A,HL:IF J<5 THEN A=A-
A(J-1):GOTO 220
255 IF J=128 THEN STOP
300 REM MAZE IS DONE! WAIT FOR KEYPUSH
310 IF PEEK(764)=255 THEN 310
320 POKE 764,255
```

Program 4: Atari Version

```
1000 REM MAZE TRAVERSAL ALGORITHM
1010 POKE A,84:J=2
1020 B=A+A(J)/2:IF PEEK(B)=HL THEN POKE
B,84:A=B:J=(J+2)-4*(J>1)
1030 J=(J-1)+4*(J=0):GOTO 1020
```

Figure 2.

Figure 3.



Part Two:

An Introduction To Binary Numbers

Charles Brannon
Greensboro, NC

This is the second in a series of articles on elementary computer arithmetic. The previous article, Part One, described the binary numbering system, as used on a microcomputer. We will now delve into the use of binary numbers — adding and subtracting.

We'll start with the simplest one first — addition. Besides, you have to know how to add before you can subtract. As you might have realized, binary addition should be rather simple, since you are only adding ones and zeros. The few complications involve the *carry*. Just to refresh you on that, here is a sample base ten addition:

```
  23
+ 51
  ??
```

To add 23 and 51, we add the numbers digit by digit, from right to left. So first we add 3 and 1 to get 4, which we place underneath the digits added. Next we add the 2 and the 5, and place a 7 under those digits to get:

```
  23
+ 51
  74
```

The carry comes in when we add two numbers and get a result too large to fit into a single digit, as in 6+8. In this case we have "four, carry the one," thus:

```
  1
   6
+  8
  14
```

Notice that the carried one drops down into the next place in the number. If we were adding 16 and 8, the carry would be added to the 1 in 16, resulting in an answer of 24.

Now all of this is very elementary, but it demonstrates all the necessary actions to add in binary. Here is the "truth table" for addition in binary:

```
0+0= 0
0+1= 1
1+1=10
```

The first three additions are "common sense," but the final one, 1+1=10 deserves a second look. We know that one plus one equals two, but we're working in binary, so two is expressed as one-zero, or 10. This is also equivalent to "zero, carry the one," since "10" cannot fit in a single digit.

Let's run through a sample addition in binnary:

```
     1111
  00000101     ( 5)
+ 00001011     (11)
  00010000     (16)
 (87654321)
```

1. **1+1=0, carry the one**
2. **0+1=1, plus carry of 1 gives 0, carry the one**
3. **1+0=1, plus carry of 1 gives 0, carry the one**
4. **0+1=1, plus carry of 1 gives 0, carry the one**
5. **0+0=0, plus carry of 1 gives 1 — no carry!**

As always, since we are working with eight-bit bytes, we fill all unused digits with zeros. This is important.

As you can see, a single one can cause a whole string of carries, almost like a chain reaction. It is possible that the carry could be continued past the seventh bit (marked 8 above). Therefore, most microprocessors have a special register, called the *carry bit* to hold and signal this runaway bit. This bit is essential in adding multibyte numbers, which we will cover in Part Three. Let's try another addition.

```
       11
  00011101     (29)
+ 00110010     (50)
  01001111     (79)
 (87654321)
```

This time we have an interesting effect of the carry. In step 5, we get 1+1=0, carry the one. In step 6, we add 1+1+ the carry of 1 to get 1, carry the one (1+1+1=11). The carry comes to rest at step 7. Incidentally, I have numbered the bits from 8 to 1 for convenience. In reality, they are numbered from 7 to 0, the exponents of the powers of two. (Bit $6=2^6=64$).

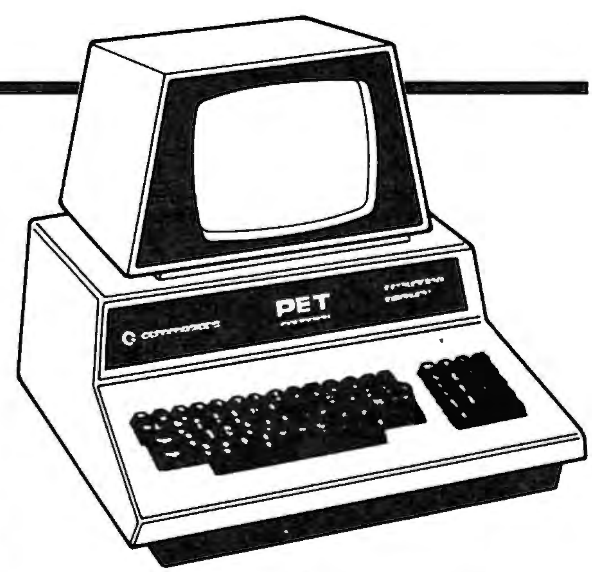
You now have the necessary information to add in binary, but in order for it to really "sink in," you will have to practice it until it becomes clear. You can make up your own exercises by randomly stringing a series of ones and zeros together to form two eight-digit numbers. Then add them in binary. To check your answer, convert the addends and the answer into decimal, which you can easily verify.

When you are confident that you can add in binary, you are ready to grasp this section on subtraction. When we perform subtraction in our normal, base ten system, we are really just adding the two numbers. For example, 8 - 5 is equivalent to 8+(-5). -5 is pronounced "negative five." It is assumed that you are aware of negative numbers, as it is taught as early as sixth grade, but we all can forget, right? All that is necessary is to know that,

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

Expect more from your PET®...



PET is the registered trademark for Commodore Business Machines, Santa Clara, CA.

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 ________

when you add a negative and a positive number, you can get the same result as subtracting the smaller number from the larger number, and giving the answer the sign of the larger number. When you add two negative numbers, the answer is the same as adding the numbers, disregarding their sign (the absolute value), and then giving the sum a negative sign (e.g. (-4)+(-3)=-7). Yet believe it or not, subtraction in binary is even easier than in decimal, as a comparison will show.

First we have to know how a negative number is expressed in binary. Since a binary number is composed of ones and zeros, there is no place for the minus sign. Therefore, the highest bit, bit seven, is used to show that the number is negative. Most microcomputers use a technique called "two's complement" to convert a number into its negative equivalent. If you add numbers using two's complement, the subtraction will be performed automatically. Two's complement has two steps — forming the complement, and adding 1 to it. Numbers properly represented using two's complement are called *signed binary*.

Let's form the signed binary equivalent of -5. Here is the binary equivalent of five: 00000101. To complement it, we turn all the zeros into ones, and all the ones into zeros to get: 11111010. Next we add 1 to it to get:

```
  11111010
+ 00000001
  11111011
```

Positive numbers in signed binary are expressed normally, with the restriction that they must not be greater than 127. If they were, bit seven would be "on," and the number would look as if it were negative. The number 205 in straight binary is 11001101. This is also -51 in signed binary. You can find the value of any negative number in signed binary by running it through the two's complement routine again. You'll get the positive value of the number. Similarly, if you try to make any number larger than 128 negative, it will end up positive. Therefore, in signed binary, the value in the byte must be between -128 and positive 127. Now that we have our background, let's try out our skills.

Subtract: 43-11. 43=001010011

```
-11 =   00001011
        11110100   complement
      + 00000001   plus one
        11110101
```

Add 43 and -11:

```
        1111111
        00101011     43
      + 11110101    -11
        00100000    -32        C:1
```

Notice that the carry was swept out of the byte (C:1). C: represents the imaginary carry register. This carry should be always disregarded in two's complement subtraction. The most wonderful thing about subtraction in binary is that it is seemingly "automatic." But once again, for complete understanding, you must practice subtraction until you feel sure of your comprehension. For this purpose, exercises are once more included at the end of this article.

Next time, we'll learn about *multibyte* numbers and even get into a wee bit of MACHINE LANGUAGE!

Answers to exercises in PART ONE:

1. a) 21 b) 51
   c) 60 d) 255
2. a) 00110100 b) 11101010
   c) 01000010 d) 00001111
3. The complete chart to sixteen bits:
   32768 16384 8192 4096 2048 1024
   512 256 128 64 32 16 8 4 2 1

EXERCISES

1. **Add:**

```
a)   00101011      b)   01000011
   + 00000111         + 00011000

c)   00111000      d)   10011010
   + 10100111         + 00111001
```

2. **Convert to binary and add:**
   a) 20+11
   b) 18+56
   c) 29+47
   d) 32+64
3. **Complement only:**
   a) 01010110 b) 01100011
4. **Form the two's complement**
   a) 01111001 b) 10111111
5. **Convert into signed binary:**
   a) -14 d) 108
   b) 22 e) -9
   c) -134
6. **Convert to binary and subtract:**
   a) 56-18 b) 99-33
   c) 58-78 c) -105 -12
7. **Why is -56 equal to 200? (Trick question)**

Book Review:

Microprocessors For Measurement And Control

If your business or pleasure is realtime control applications this could be a valuable book for you. Seven realtime control applications are described in complete detail. These include DC motor control, position control, control of temperature, an automatic weighing system, a plotter, a computer controlled saw, and a blending process control system.

Each application is described in great detail, including circuit diagrams, flowcharts, state-transition diagrams, timing diagrams, and a complete discussion of the algorithms. The book is replete with pictures and diagrams. Having studied the examples, readers will be able to think of and design their own control systems. Do not decide against the book simply because your application is not described: there are enough general principles to make the book valuable for anyone working on realtime control of a device by a computer (especially if the device is a robot that will mow lawns and shovel snow).

The book is not written for the novice. Some experience with microcomputers, machine language, binary numbers, and input/output operations is desirable. If you haven't worked with a single-board machine or peeked inside your Apple, PET, or Atari to see what makes it work, then this book is going to be tough sledding. To actually construct the projects described will require electronic test equipment such as an oscilloscope, signal generator, breadboarding equipment, and components.

I liked the book. I liked the idea of describing as application from first principles to the last detail, giving both the theoretical background and the practical implementation of the application. This is because my computer interests gravitate toward interfacing and control. On the other hand, if you are strictly a programmer who is happy with business applications, games, computer aided instruction or number crunching, then this book is out of the mainstream of your current interests.

Of great importance to the 6502 community is the fact that almost half of the book (approximately 155 pages) is devoted to program listings in BASIC, PASCAL, C, and FORTRAN, as well as 8080 assembly and machine language. This half of the book will be almost useless for the great majority of 6502 purists, unless you are familiar with several of these languages, particularly the 8080, Z80, or 8085 instruction set.

The book is characteristic of the generally fine quality of the computer literature published by OSBORNE/McGraw-Hill and, if you are interested in computer control of devices, this book is a good investment.

Reviewed by Marvin L. De Jong: the reviewer is Professor of Physics at The School of the Ozarks, Pt. Lookout, MO 65726. He is the author of the book "Programming and Interfacing the 6502, With Experiments," published by Howard W. Sams & Co., Inc., 4300 West 62nd Street, Indianapolis, Indiana 46268.

David M. Auslander & Paul Sagues
OSBORNE/McGraw-Hill, Berkeley CA
1981
(630 Bancroft Way, Berkeley, CA 94710)

CONTROL!!!

We have a complete and affordable controller development system.

The controller board: MMC/03 comes with 1K RAM, 2K EPROM socket, 2 6522s, 6503 CPU. Uses any AC or DC power supply.
Kits from $89.00, assembled and tested from $119.00.

IN-CIRCUIT EMULATOR: better than an EPROM simulator. Works with any 6502-based system including Apple, PET, AIM, KIM, SYM, OSI. Not $5,000.00 but under $100.00!

EPROM programming adaptor: complete with software driver, programs any 24-pin +5V EPROM including 2532s. Battery powered: $40.00, with AC power supply: $50.00.

See COMPUTE! April 1981 for Eric Rehnke's review.

Call or write us for more information:

R. J. Brachman Associates, Inc.
P.O. Box 1077
Havertown, PA 19083
(215) 622-5495

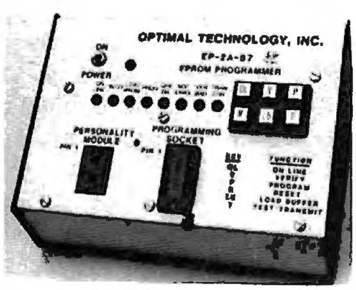
Coming soon: The MMC/02, complete with ICE: 1K RAM, 6K EPROM; or 3K RAM, 4K EPROM.

Registered TM-Apple-Apple Computer Co., PET, KIM-Commadore, AIM-Rockwell International SYM-Synertek

Model EP-2A-87

EPROM Programmer

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



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

Personality Modules

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

Optimal Technology, Inc.

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

"INSTANT PASCAL" AIM-65 SYSTEM

Employing "Instant Pascal" (A65-PS) on the AIM-65 requires the use of an expansion board for 4 of the 5 ROM's provided in the set.

EXCERT is offering assembled and tested systems with 20K RAM, warranteed for 6 months, that use either the BANKER™ Board (MEB3-2A) from Micro Technology Unlimited or the DRAM PLUS™ Board (MEB1-2A) from the Computerist and are *totally* enclosed within the Enclosure's Group case including the power supply (ENC3A), but excepting the interconnecting cable (CBL1 or CBL3). These systems include the following EXCERT Part Numbers:

"INSTANT" SYSTEM I		"INSTANT" SYSTEM III	
P/N: CSA65 - 4PS - 3A/1-2A		P/N: CSA65 - 4PS - 3A/3-2A	
A65-4	$469	A65-4	$469
A65-PS	100	A65-PS	100
ENC3A	130	ENC3A	130
— DRAM PLUS™ —		— BANKER™BD. —	
MEB1-2A	325	MEB3-2A	265
CBL1	25	CBL3	25
Total System Cost	**$1049**	**Total System Cost**	**$989**

Rockwell's recommended system using RM65 boards assembled in the Don-El large enclosure would cost nearly $1400 (suggested retail).

Call or write for complete list of AIM-65 Products and Accessories or other Custom AIM-65 Systems.

NEW PRODUCTS!

CUBIT featuring a 4½x6½ AIM-65* for $195
*less monitor, printer, display & keybd
APPLIED BUSINESS COMPUTER featuring
64K RAM Board - $495
80 column Video Controller - $325
5¼" Disk Controller w/ADOS™ - $485

Educational Computer Division
EXCERT INCORPORATED

- SALES
- SERVICE
- INSTALLATION
- CONSULTING

P.O. Box 8600
White Bear Lake, MN 55110
(612) 426-4114

Console Input/Output

Gene Zumchak
Buffalo, NY

Perhaps I'm stepping out of my domain to write an article on a software topic; however, since no "expert" has volunteered an article on the subject, I'd like to say a few words about the very important subject of console input/output.

Input/Output is the interface between the computer and the outside world. Simple input/output consists of switches, relay contacts, indicators, etc. Two other classes of I/O are console I/O and mass storage I/O. The latter would include tape or disk or any other method (usually using magnetic medium) of storing and retrieving large records or files. I'm limiting my discussion here to console I/O.

My experience is mainly with single-board computer types like the KIM, SYM, and AIM, and I will use them as examples, though the principles will apply as well to console systems like APPLE and PET.

A general-purpose computer system is of little value unless a user can communicate with it. This requires two things. First, the computer must have some minimal operating system to permit communication. Second, the computer must be connected to a console device. Traditionally, a computer's primary console device was a teletypewriter. This provides input (keyboard) and output (printer) and sometimes mass I/O in the form of punched paper tape. As a bonus the teletypewriter provides hard copy. More recently, the teletypewriter has been replaced by a CRT, or a CRT substitute, as the console device.

A CRT terminal, like the teletypewriter, is a serial device. It usually has a RS-232C voltage interface however, as opposed to the current loop interface of the TTY. There is, of course, no reason why console input cannot be a parallel keyboard, or the output a parallel or memory mapped display. Most computers with a built-in console device usually treat I/O directly in parallel.

The way that console I/O is treated is a function of the sophistication of the operating system software. At one extreme, some systems permit any devices to serve as console input or console output. At the other end, only a specific device pair can serve as console input and output. The earliest 6502 computer, the KIM, is between these two extremes.

How To Use Non-Serial Devices On KIM

The KIM has two console options: either the built-in keyboard and display or serial teletype format I/O. The choice is made by a jumper on the application connector. (The KIM actually uses separate programs to treat I/O from the two console options). The user cannot, however, communicate with the operating system with a non-serial I/O device (a parallel video display for example) since the KIM makes no provision for interfacing any non-serial console devices. There are other problems. The tape routines cannot be employed in user programs since they terminate with a jump to the MONITOR instead of an RTS. This is an important point. If you are going to write any kind of routine that might find use elsewhere, write it as a subroutine. Still, the KIM with its monitor is really quite remarkable, considering that it was available within weeks of the 6502 chip itself. The hex keyboard and display, the built-in serial interface, and the built-in tape interface were important innovations.

Two other classes of I/O are console I/O and mass storage I/O.

How does one use a non-serial console device on the KIM? The only choice is to do without the KIM's monitor and replace it with one of your own that can accommodate your console I/O. Since all the KIM monitor does is inspect and change memory, giving it up is not a great loss. For other systems with somewhat more extensive operating systems, replacing the operating system with one of your own is no small project.

SYM Avoids The Problem

The author of the SYM operating system recognized this potential problem and avoided it. This was done by "vectoring" the console I/O. When the SYM is reset, it initializes a block of operating system RAM. Among the locations initialized are an input vector (INVEC) and an output vector (OUTVEC). These occupy three RAM locations each. The first contains the JMP op code, $4C; the next two locations are the low and high address of the specified routine. Thus, JSR INVEC will cause the program to run the routine whose address is found at INVEC, and return to the instruction following the JSR.

MTU Introduces The Complete Desktop Computer

The MTU-130 ™ computer is THE COMPLETE 6502 system. This desktop system is designed for people who need to maximize their computing and minimize their learning time. It gives you the features you need to perform your applications.

A desktop computer should have clean expansion beyond the standard system. The MTU-130 is designed with an 18 bit address bus for up to 256K memory (80K standard) and includes an internal card cage for expansion boards or your own custom boards when needed. Of course, the power supply and fan have sufficient capacity for expansion. We even have provided rear panel cutouts for custom connectors if you need them for that special task you have to perform.

The human interface features of this system include: a 96 key keyboard with programmable function keys and displayed soft legends, a bit mapped display with 480 x 256 pixel resolution graphics, 80 column text (gray scale also), an 8 bit audio port for speech, music and sounds, and a high speed (60 points/sec) fiber optic light pen. Other standard I/O includes 2 parallel ports with handshaking and a serial port with software selectable 50-19.2K baud-rates. Of course connectors are provided on the rear panel.

You interact with the MTU-130 through our field proven Channel Oriented Disk Operating System (CODOS) which permits you to easily customize your system. Using CODOS ™, any file is transferred from disk to anywhere in memory at a sustained speed of 19.6K bytes/second (not burst speeds!). Files are handled automatically, freeing you to perform at your peak. Auto-execution of "jobs" when power is turned on can turn the MTU-130 into a dedicated-function system. A monitor with 32 commands and 19 utilities is standard. Text or data can be easily transferred to or from other systems on IBM or CP/M* (or others) format disks with our optional DISKEX ™ program.

Our standard full screen EDITOR allows you to edit text or program files with rapid positioning anywhere in the file. It edits any file size that fits on the disk (not just in memory) and will edit a file in place or save a backup copy. The concept "what you see on the display is what exists in the file" has been employed which significantly reduces your learning and interacting time. This is a very powerful tool usable by anyone.

If your needs include software development, you will find our optional MOS Technology compatible ASSEMBLER and DISASSEMBLER extremely fast, significantly reducing your development time. For example, a 210K byte source program with 6300 lines and 800 symbols can be assembled in less than 4 minutes. This includes generating the object file and the listing with sorted symbol table and cross reference map on disk. This can be accomplished on a standard 1-drive MTU-130-1S.

If you prefer to program in high level languages, keep in mind that the MTU-130 is RAM-based, not ROM-based, giving you the maximum memory possible for the use with any language. Our version of MICROSOFT BASIC is standard with MTU-130 systems. It allows libraries of commands to be added when needed such as our Virtual (floating point) Graphics. PASCAL and FORTH are planned.

The base standard MTU-130-1S system comes with one single-sided, double-density 8″ floppy disk, a 12″ green phosphor CRT, and MTU-BASIC for $3995. The 3 other models contain 1 or 2 single or double sided drives priced up to $4995 for 2 Megabytes of storage. You can choose an MTU-130 without disk drives, languages or CRT for $2640. 4 Megabyte systems available on request.

We obviously cannot describe fully all of the details of the MTU-130 in this advertisement. If you want to know more about this complete desktop computer, call or write for our complete 28 page descriptive literature. International requests include $5.00 U.S.

COME TO MTU - for excellence in microcomputing systems.

*CP/M is a trademark of Digital Research.

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

The SYM initializes INVEC and OUTVEC to point to the routines that service the on-board keyboard and display. However, should the serial input bit become active before a key on the keyboard is pressed, the SYM will switch the vectors to point to the serial I/O routines. At any time after reset, the user is free to change either the input or output vectors to point to his own routines. For example, suppose you wished to talk to the SYM using a parallel ASCII keyboard, but wanted to retain the on-board display. You would write a routine to service the ASCII keyboard and put the address of your program in INVEC. Now when the SYM looked for input, it would get it from the ASCII keyboard via INVEC.

The lack of vectoring for console I/O is most evident in the AIM system.

Recently, as an experiment in a course I was giving, I wrote a routine to service an ASCII keyboard, attached to one of the SYM's ports. The ASCII data went to the low-seven bits; the keyboard strobe went to the high, or sign bit. I changed the input vector to point at my program. When I attempted to use the SYM, however, something strange happened. As I entered the monitor command, nothing happened until I hit the carriage return required to execute a command. That is, I did not see my command being entered on the SYM's display. This problem illustrated that there are two distinct kinds of input routine.

In a pure input routine, the program waits for an input, returning with the value (in the accumulator) when the input occurs. The SYM, however, expects an input routine with echo. Such a routine, before returning and giving up the character, causes the character to be sent to the output device. Thus, you are able to see the character as it is entered. Inputs are generally echoed, but there appears to be no agreement as to whether the echoing should take place as part of the input routine, or that the routine calling for the input should echo the character before processing it. Examples of both are common.

If you are writing a routine to service an input device, you should include both styles. Given a pure input routine, INPUT, an input with echo routine is just two instructions:

```
INWITHECHO  JSR INPUT
            JMP OUTPUT ; (OR OUTVEC)
```

Another approach is to write a common input routine for both styles, and have the routine determine whether echo is desired or not with a flag. This is the method used by the SYM for its serial input routine. A RAM location called TECHO determines whether echo is desired. Instead of first inputting a character and then echoing it out, the SYM just causes the output bit to follow the input bit as the input character is being received.

It should be noted that in the INWITHECHO routine above, the OUTPUT routine must not destroy the character being output. This is a very important property that all output routines should have.

When I wrote my operating system for my KIM to accommodate a parallel ASCII keyboard and parallel video display, I did not know about vectoring. I then wrote some action games for the video display which used the video output routine which I had in EPROM. A problem arose when I upgraded my I/O routines. The locations of the video output routine changed and, when I tried to load and run a game, it would bomb since it was pointing to a non-existent output routine. This problem could have been avoided if my operating system used vectored I/O. The game program then, would always point to the output vector. Even if the location of the actual output program changed, the vector could be changed to point to the new output routine. That is, I would not have to make modifications to the game program every time the operating system was changed.

AIM Software: A Curious Mix

The lack of vectoring for console I/O is most evident in the AIM system. The AIM software is a curious mixture of very clever programming and serious oversights. Like the KIM, the AIM has two choices for console input, the built-in keyboard and display/printer, and a serial (TTY or CRT) interface. The choice is made by the slide switch. The switch affects both input and output simultaneously. (It should be understood that the vectors UIN and UOUT on the AIM have to do with mass I/O and do not affect console I/O.) For example, suppose you had a serial video device which you wanted to use for output, but you wished to use the AIM's ASCII style keyboard for input. If you put the switch in the TTY position to get serial output, the AIM would now look for input from the serial channel and you could not use the keyboard. The switch should have been used to initialize the I/O vectors. Then, after the fact, the user could change the input vector, the output vector, or both, to accomodate any special console I/O.

In all fairness, the console output is vectored in a fashion. A vector called DILINKS was included so that output could be echoed to a video display. However, a carriage return appears as $8D and not $0D. A backspace is echoed as a space. Thus, any video device will not be able to respond properly to a backspace or delete. Instead of backing up one, it will go ahead one. The reason for this is that the AIM processes the delete by backing up the display pointer and overwriting the previously written character with a space. Incredibly, the delete is processed in the input routine. An input routine should be responsible for returning characters, period. It should not make value judgements on the characters or play around with output, except for straight echoing.

Although the AIM keyboard resembles that on a CRT, complete with Shift and Control keys, it can be used only as a TTY style (uppercase only) keyboard. While it would not be difficult to write a new input routine to produce lowercase characters and provide for "Caps lock" when appropriate since console input is not vectored, there's no way to tell the AIM that it should use your keyboard routine.

The lack of vectored I/O is evident in the AIM's software listing. In many places in the program, changes were made by jumping to a "patch" area near the end of the listing and then jumping back onto the program. Why didn't the authors just insert the necessary changes and reassemble? Apparently, the I/O addresses from an early version were used when making the BASIC or assembler ROMs, making those addresses inviolate. Thus, changes to the monitor, however necessary, could be made only if they did not affect the addresses of the I/O routines. Had vectors been used, the monitor could be updated and improved at any time, without affecting compatibility with ROMed accessories like BASIC.

Console input/output is an essential element in any general-purpose computer. The ability to customize and personalize a computer system's console will depend upon whether or not console I/O is vectored. Non-vectored console I/O places serious restraints on the system and on the user.

In a later installment, I plan to show how vectored I/O can be taken advantage of to "massage" canned I/O routines and overcome objections to ROMed software accessories.

MTU-130: A New 6502 Microcomputer

Micro Technology Unlimited of Raleigh, North Carolina has announced the development of a new "top-of-the-line," general purpose microcomputer. The first production shipments were announced for November for this 6502-based machine which will retail for $3995 (with single-sided disk drive, 500,000 bytes storage). Other packages are offered, which increase disk storage, up to a unit with two double-sided drives, two million bytes, for $4995.

These prices include the MTU-130 computer with 80K RAM, a 12" green phosphor CRT module, the selected floppy drive(s), all necessary cables, the operating system CODOS, an Editor, four-voice, digital, synthesized music, and a demo disk.

Novel Features

"MTU believes that the user should receive a system powerful enough to perform all necessary functions without having to add memory expansion, graphic expansion, etc...." the designers remarked. The result is a computer which is fully, one might say luxuriously, implemented.

The unit features a 1MHz 6502 with 18 bit addressing for up to 256K clear address space. Three video display operating modes: 1. bit-mapped black and white high resolution graphics 480 wide by 256 high; 2. 25 lines by 80 characters, mixable with graphics; and 3. bit-mapped graphics with four levels of gray in 240 wide by 256 high.

All the software is in RAM permitting easy upgrading or personalizing. It includes a CODOS disk operating system, printer drivers (see the high-resolution possible on a definable dot-matrix

Figure 1.

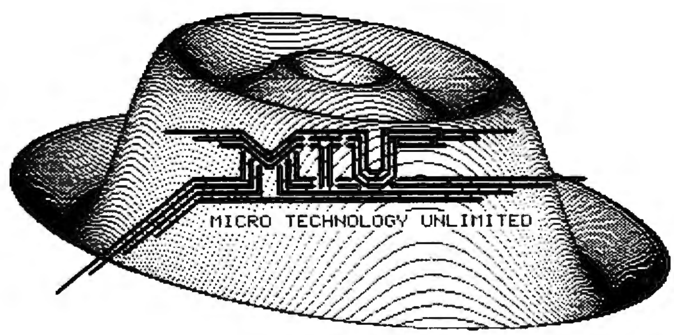
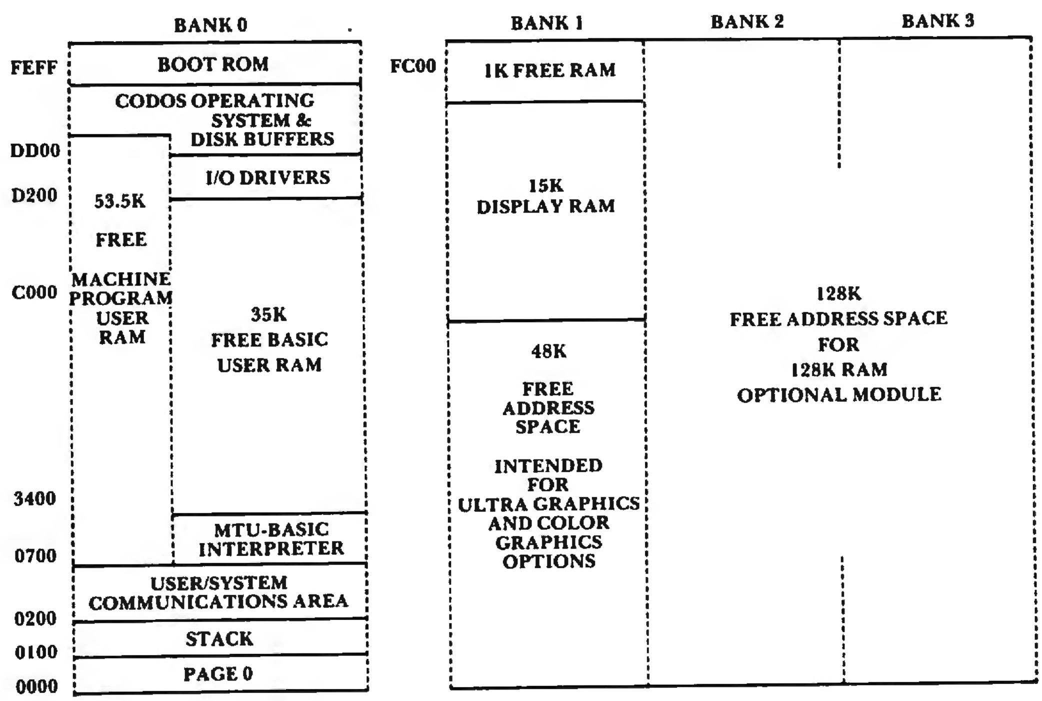


Figure 2.



NOTE: I/O addresses occupy 0BE00-0BFFF when enabled under software control.

printer in the photo), two eight-bit parallel ports and one RS-232 serial port with software select of baud rate, an eight-bit D/A port with filter and amp (for speech, sound, and music), and an interface for a 50K Baud, interrupt driven, network option.

Additionally, the MTU-130 contains four EPROM sockets which are software controlled, a high resolution light pen, separate cursor keys, and a bank of eight user-defined function keys.

A unique approach to bank switching — using indirect addressing on the 6502 — allows one 64K section of memory to contain a program while the data resides above in its own 64K zone.

Digitized Sound

The optional MTU-BASIC 1.0 with graphics and disk library extensions is an enhanced Microsoft BASIC. Currently, bank switching is not available to BASIC directly, but the system permits relatively easy user enhancements. Also, when the computer is turned on, it *says*, "MTU model 130. Please enter today's date." The "voice" is entirely digital and sounds remarably human (except that high frequency is muted — the cutoff is around 4 KHz). This provision for digital storage of sound is exciting, but, like high resolution graphics, it is a byte-eater. A two second message uses *16K* on the disk. This space can be reduced, though, and plans are in the works to make the storage more efficient. The manufacturer also expects to provide an optional A/D microphone peripheral which will permit owners to digitize their own messages.

The eight inch floppy drive spins all the time, but the head remains out of contact with the disk until necessary. And it is fast. A 14K high-resolution picture can load to screen in about two seconds. Transfer rate is over 19 thousand bytes per second, sustained.

Future Options

MTU is currently working on additional software for the 130. Expected in early 1982 are FORTH, PASCAL, cassette I/O, PET/Apple BASIC translator utilities, and a word processor. Planned hardware includes a 128K memory expansion board, the A/D microphone system, a high fidelity sound synthesis and analysis package, a network operating system, and a rigid disk controller.

A prototype board for construction of custom circuits and a banker board are available now as options.

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

THE apple® GAZETTE

Animating Applesoft Graphics

Leslie M. Grimm
Mt. View, CA

Animating graphics can add a special plus to your BASIC program. A previous article (**COMPUTE!** #14) described how to animate low-resolution graphics in Integer BASIC. A method is described here to do animation of either high-resolution or low-resolution graphics in Applesoft BASIC.

Before beginning, however, a few words comparing the two BASICs for this purpose are in order. Integer BASIC is much faster than Applesoft. This is because the Applesoft interpreter must perform time-consuming manipulations of floating point arithmetic, whereas the Integer BASIC interpreter ignores everything to the right of a decimal point. The effect of all this is that Applesoft graphics routines run about half as fast as Integer routines. This can be crucial in animation.

In general, if the object to be animated is very large (bigger than ¼ of the low-resolution screen area or bigger than about 20 x 20 dots in high-resolution) you will get better results in Integer. However, choice of Applesoft may be a matter of necessity for a variety of reasons. By keeping animated objects small and simple, and observing other speed-increasing tips mentioned below, you can get very nice effects.

Designing The Figure

For the low-resolution example listing below, the figure of a flying bird was chosen. The high-resolution example uses a simple shape (square) for the sake of brevity in this article, but you could modify the bird or make any shape you desire for high-res.

Whatever shape you choose, your first step is to draw the figure in various states of motion. Use graph paper, and number the squares as shown in Figure 1. (This applies to low- or high-res shapes.). Note that, for the flying bird, three different positions simulate the action of flying.

Because the figure will be moving about on the screen, you need to use relocatable coordinates in your plotting routine. Consider the square in the upper-left-hand corner as X=0, Y=0. Then specify all other points relative to that point. For example, a point five squares to the right and three squares down would be called X+5,Y+3.

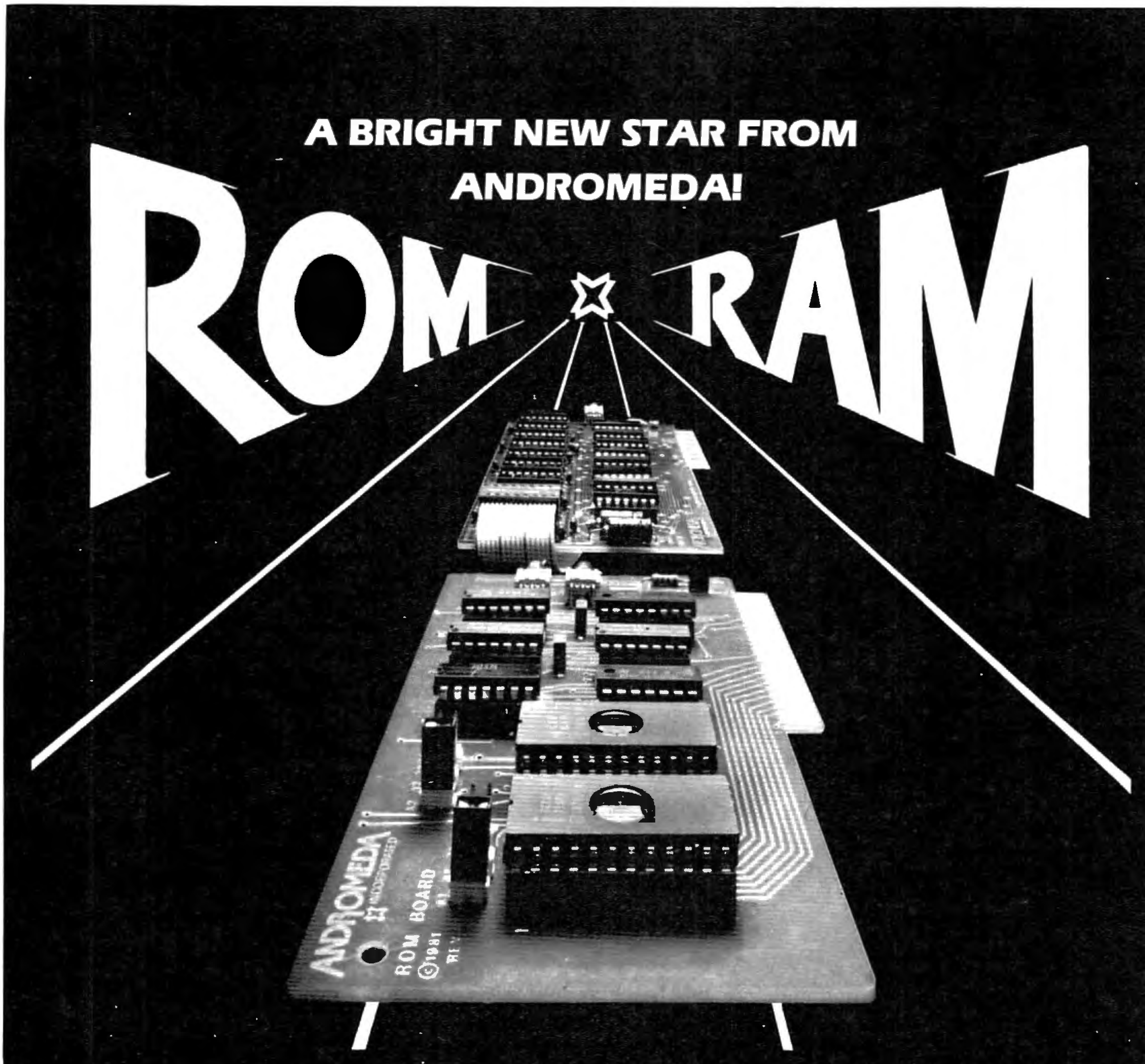
You should also think about the most economical way to draw the figure. In the case of the bird, you can see that the body is the same for all three drawings. One subroutine was made for it, and another for the wing in its upward position, and still another for the wing in its downward position. To draw the bird with its wing up, the program does a GOSUB to the body routine (at 100) followed by setting hue to 2 (blue) and issuing a GOSUB to the wing-up routine (at 110). Note that the subroutines for wing up and wing down use a variable (hue) for color. This way the same subroutine can be used to draw (hue = 2) or erase (hue = 0) the wing.

In writing the code it is important to keep speed of execution in mind. As much as possible you should put many statements on a single line, separated by colons. Use HLIN and VLIN commands instead of a lot of HPLOTs. Locate your graphics subroutines at low line numbers.

Smooth Animation

The basic technique in animation is to draw the figure at a certain location on the screen, then erase it and redraw it at a new location. (An alternative method is to draw the figure at location one, redraw it at location two, and erase the parts that are left over from location one. If you know that your figure will always move exactly the same number of spaces each time it is redrawn the latter method is preferable. It could work reasonably well without page flipping also, but, because it is not the most general case, it is not demonstrated here.)

For the flying bird, the erase procedure was done with two routines. Line 150 draws the body in color =0 (black), and then hue is set to zero and the appropriate wing routine is used. Note that if you wanted to use a colored background the erase routine could use the color of the background



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.

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

VISA
master charge

Price for Andromeda 16K RAM expansion board now only $120.00. Please add $5 for shipping and handling. North Carolina residents add 4% sales tax.

*DEALER INQUIRIES WELCOME.

rather than zero.

If that were all you did, though, you would probably be disappointed in the results. This is because you would be watching the figure being drawn and erased on the screen. This is distracting and can be avoided by the technique of "flipping pages." Pages can be flipped for either low-resolution or high-resolution graphics, and the methods to do this are described separately below.

The technique for flipping pages is similar for low- and hi-resolution graphics. There are two graphics screen pages for low-resolution graphics (beginning at $400 and $800 respectively) and two screen pages for high-resolution graphics (beginning at $2000 and $4000 respectively). Your program will display one page to the user while erasing and drawing "behind the scenes" on the other page.

In low-resolution graphics it is not possible to draw directly to the second screen page. Drawings can only be placed on screen page two by first making them on page one and then calling a routine in the Apple monitor to move the contents of page one onto page two.

You will need a short assembly language routine to do the move for you. The subroutine beginning at line 10000 pokes this assembly language routine in memory. All you need to do is CALL the routine when you need it.

(A description of how the routine works follows, but you don't need to know how it works to use it. Just skip on to the next paragraph if you wish.) The LDA $C054 at line $C00 causes the Apple to display page one. The lines from $C03 to $C15 specify that the contents of memory locations $400 through $7FF (graphics page one) are to be moved to the region from $800 to $BFF (graphics page two.) Line $C17 sets a counter (Y register) to zero, and the next line does a Jump to SUBRoutine (JSR) at $FE2C — the move routine in the Apple monitor. The move routine transfers the contents of page one to page two very quickly. Line $C1C causes page two to be displayed, and the last line ReTurnS you to your BASIC program.

Bird In Flight

Line 10 sets text mode (in case a previous program had left the machine set to graphics mode) and clears the screen Line 20 POKEs the assembly language routine in via the subroutine beginning at line 10000. Line 40 branches around the graphics subroutines to the start of the animation program. (The graphics subroutines were intentionally placed at low line numbers for speed of execution.)

The animating program first clears the screen (page one), sets initial values for X and Y, and calls the move routine (CALL 3072). The user will now be looking at page two, which is blank. Next, line 1010 draws the figure in its initial position (wing down) behind the scenes on page one. It then calls the move routine. Remember that the move routine displays page one while it is copying page one onto page two, and then flips to page two. The user only sees the finished drawing, first on page one and then on page two. The flip between pages doesn't show.

While that drawing is being displayed the original figure on page one is erased (line 1020). The value for X is changed and the figure is redrawn in a new position (wing up) and a new location (line 1030). Once again the move routine is called to put the new drawing on page two and show it to the viewer.

Line 1040 erases the wing-down bird, moves the bird over and up, and draws just the body. Then it performs the move and flip. In line 1050 the body is erased, and the bird is drawn with wing down in its next location. The move and flip is called again. This process is repeated several times in a FOR ... NEXT loop.

The last lines of the routine restore the display to graphics page one. The cursor is VTABbed to line 21 so that it will be visible when the program ends. The POKEs instruct the computer to locate the next Applesoft program at the normal location ($800). (See below)

In entering and debugging a program that flips pages you may occasionally get "stuck" on page two due to a programming error. When this happens you will hear the beep that accompanies an error message, but no message will show and there will be no cursor. Just type "POKE 16300,0 to restore the display to page one and see your error message.

Relocating

There is one more step required before you can actually run this program. Page two of low-res graphics occupies the same place in memory that your Applesoft program normally occupies. Your only alternative is to relocate your Applesoft program. To do this, before you load your program you must change the values of the "program start" pointers to a new value. This will cause your program to be loaded in at a different place than usual.

The Applesoft program could be relocated to many possible places in memory. In this example it was located at the end of the assembly language subroutine. The assembly language subroutine was placed just above the second low-res graphics page. Alternatively, one could put the assembly language routine at $300 (decimal 768), but since this area is often needed for music routines, it was left free here.

There are several ways to relocate the program. One way is to type the following commands before running your program:

POKE 103,33
POKE 104,12
POKE 3104,0

The first two POKEs place the starting address of the program in memory. The third POKE sets the first byte of the program location to zero, which must be done in order for the Apple to find the program's beginning.

Alternatively, you can write a short program to do the POKEs for you. A sample listing is Program 1.

(A third method, which incorporates the relocating program as a subroutine of the main program, will not be explained here for the sake of brevity.)

Whichever method is used to relocate the program, it is a good idea to restore the pointers to their usual values at the end of your program. The next Applesoft program will then load into the normal area of memory. This is shown at the end of the example program.

Flipping between high-resolution pages is easier than flipping in low-resolution graphics because it is possible to draw directly on either page. Also, it is not necessary to relocate your Applesoft program. However, only very small drawings can be animated in BASIC, due to speed limitations. Program 3 moves a very small, simple shape (square) diagonally across the screen, flipping pages between each move.

Line 10 clears both hi-res pages and sets the screen to full-screen graphics. Full screen is necessary to prevent text "garbage" from appearing at the bottom of screen page two.

The subroutine at line 100 draws or erases the square, depending on the value given to hue. A value of 5 sets the color to orange, and 0 is black. Line 1000 sets up the original values for X and Y, and causes page two to be displayed (POKE–16299,0).

The value POKEd into location 230 determines whether your program draws on hi-res graphics page one or two. To draw on page one this value must be 32 ($20). To draw on page two, set it to 64 ($40). Note that you could also simply specify HGR for page one or HGR2 for page two, but these commands include an implicit "clear screen" which would erase the whole screen and take far too long.

As in the low-res animation process, the program displays only finished drawings to the viewer while it erases and redraws figures on the undisplayed pages. Line 1002 directs the drawing process to page one, but it will not be seen since page two is being displayed.

Again, as in the low-res animation routine, a FOR ... NEXT loop is used. Line 1010 sets the color to orange and the GOSUB 100 draws it on page one. The POKE–16300,0 flips the display to page one when the drawing is finished. To the viewer, the drawing seems to pop onto the screen.

Line 1020 first resets X and Y to the previous location so that the last square on page two can be erased. Location 230 is set to 64 so that drawing will be done on page two. X and Y are then advanced to the new location, color is set to orange again, and the new square is drawn. Finally, the display is flipped back to page two. The viewer sees the square slide to a new location.

Line 1030 sets drawing to occur on page one again, erases the square there, and sets X and Y to the location for the next square. When the NEXT J instruction in Line 1040 is encountered, the program will jump back to line 1010, which will actually draw the square.

Line 2000 restores the display to page one, and ends. One could add the command TEXT before END to restore the viewer to text mode.

This method for high-resolution animation is not as satisfactory as an assembly language routine would be, but could be useful in many simple applications. Another possibility for a simpler way to use this method would be to have two pictures, (one on each page) showing different positions of the same figure. For example, one could have a Jumping Jack with arms up in one and arms down in the other picture. These could be large, elaborate drawings. By flipping between the two pages (POKE –16299,0, then POKE –16300,0) many times the Jumping Jack would appear to swing its arms up and down. In practice, it would probably be necessary to have a short delay between successive flips for this application.

Many other techniques of animation can be employed, but these methods should provide a starting point for the beginning or intermediate level Applesoft BASIC programmer.

Figure 1. Sketch of flying bird.

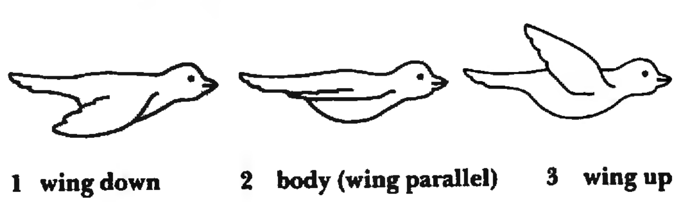
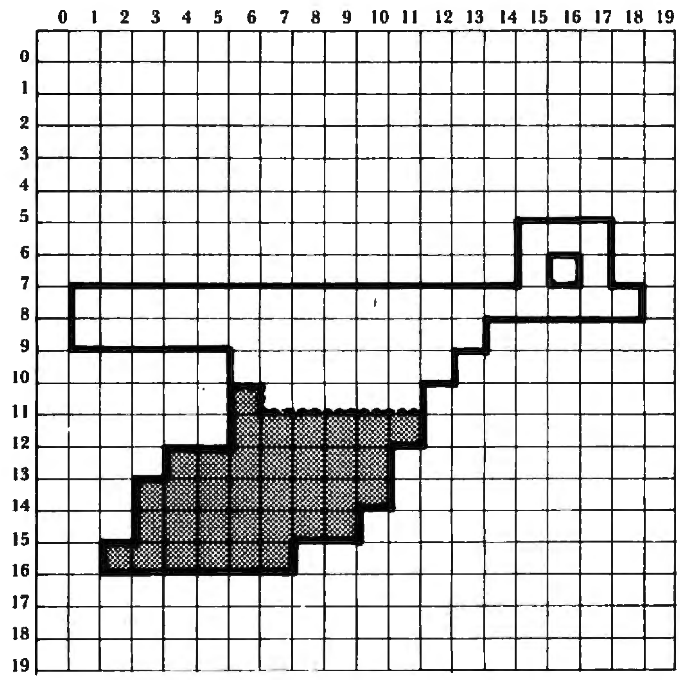
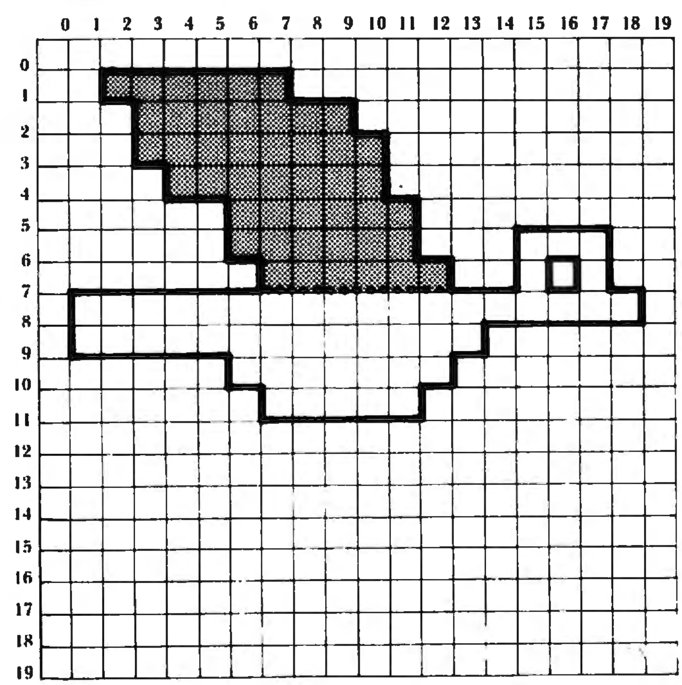


Figure 2. Wing down and body.



.... = bottom of body

Figure 3. Wing up and body.



... = top of body

Program 1. (50)

```
5   REM  BIRD LOADER PROGRAM
10  TEXT : HOME : VTAB 10
20  FLASH : HTAB 17: PRINT "LOADING": NORMAL
30  PRINT : PRINT : HTAB 13: PRINT "BIRD IN FLIGHT"
40  POKE 103,33: POKE 104,12: POKE 3104,0: REM  RELOCATES NEXT APPLESOFT
     PROGRAM TO LOAD AT $C20
50 D$ =  CHR$ (4): PRINT D$;"RUN BIRD IN FLIGHT"
```

Program 2. (10010)

```
3   REM  BIRD IN FLIGHT
5   REM   POKE 103,33, POKE 104,12, POKE 3104,0 TO RELOCATE PROGRAM BEFORE
     RUNNING
10  TEXT : HOME
20  GOSUB 10000: REM  POKE IN MOVE AND FLIP ROUTINE
40  GOTO 1000
99  REM  ** GRAPHICS SUBROUTINES **
100  COLOR= 2: HLIN X + 1,X + 18 AT Y + 8: HLIN X + 1,X + 13 AT Y + 9: HLIN
     X + 6,X + 12 AT Y + 10: HLIN X + 7,X + 11 AT Y + 11
102  HLIN X + 15,X + 17 AT Y + 6: HLIN X + 15,X + 17 AT Y + 7: COLOR= 0: PLOT
     X + 16,Y + 7: COLOR= 1: HLIN X + 17,X + 18 AT Y + 8: RETURN : REM BO
     DY
110  COLOR= HUE: HLIN X + 2,X + 7 AT Y + 1: HLIN X + 3,X + 9 AT Y + 2: HLIN
     X + 3,X + 10 AT Y + 3: HLIN X + 4,X + 10 AT Y + 4
112  HLIN X + 6,X + 11 AT Y + 5: HLIN X + 6,X + 11 AT Y + 6: HLIN X + 7,X
      + 12 AT Y + 7: RETURN : REM  WING UP
```

```
120  COLOR= HUE: PLOT X + 6,Y + 11: HLIN X + 6,X + 11 AT Y + 12: HLIN X +
     4,X + 10 AT Y + 13: HLIN X + 3,X + 10 AT Y + 14
122  HLIN X + 3,X + 9 AT Y + 15: HLIN X + 2,X + 7 AT Y + 16: RETURN : REM
     WING DOWN
150  COLOR= 0: HLIN X + 1,X + 18 AT Y + 8: HLIN X + 1,X + 13 AT Y + 9: HLIN
     X + 6,X + 12 AT Y + 10: HLIN X + 7,X + 11 AT Y + 11: HLIN X + 15,X +
     17 AT Y + 6: HLIN X + 15,X + 17 AT Y + 7: HLIN X + 17,X + 18 AT Y +
     8: RETURN : REM ERASE BODY
999  REM  ** ANIMATION DRIVER **
1000  GR : HOME :X = 0:Y = 20: CALL 3072
1010  GOSUB 100:HUE = 2: GOSUB 120: CALL 3072
1020  FOR FLY = 1 TO 4: GOSUB 150:HUE = 0: GOSUB 120:X = X + 2
1030  GOSUB 100:HUE = 2: GOSUB 110: CALL 3072
1040  GOSUB 150:HUE = 0: GOSUB 110:X = X + 1:Y = Y - 1: GOSUB 100: CALL 3
     072
1050  GOSUB 150:X = X + 2:Y = Y - 1: GOSUB 100:HUE = 2: GOSUB 120: CALL 3
     072
1060  NEXT FLY: POKE  - 16300,0
1200  VTAB 21: POKE 103,1: POKE 104,8: POKE 2048,0: END : REM  RESET PROG
     RAM START POINTERS TO NORMAL VALUE
9990  REM  ** ASSEMBLY LANGUAGE ROUTINE
9992  REM  COPIES LO-RES GRAPHICS PAGE ONE
9994  REM  TO PAGE TWO WITH PAGE FLIPPING
9996  REM  LOCATED AT $C00 (3072)
10000  FOR I = 3072 TO 3103: READ CODE: POKE I,CODE: NEXT I: RETURN
10010  DATA   173,84,192,160,0,132,60,169,4,133,61,169,255,133,62,169,7,
     133,63,169,8,133,67,132,66,32,44,254,173,85,192,96
```

Program 3. (2000)

```
5  REM  ANIMATED SQUARE
6  REM  HI-RES ANIMATION DEMO
10  HOME : HGR2 : HGR : POKE  - 16302,0: REM   FULL SCREEN
20  GOTO 1000
99  REM  ** DRAW SQUARE **
100  HCOLOR= HUE: FOR I = Y TO Y + 10: HPLOT X,I TO X + 10,I: NEXT I: RETURN
999  REM  ** ANIMATION DRIVER **
1000 X = 50:Y = 50: POKE  - 16299,0: REM  DISPLAY PAGE TWO
1002  POKE 230,32: REM   DRAW ON PAGE ONE
1005  FOR J = 1 TO 20
1010 HUE = 5: GOSUB 100: POKE  - 16300,0: REM  DISPLAY PAGE ONE
1020 X = X - 2:Y = Y - 2:HUE = 0: POKE 230,64: GOSUB 100:X = X + 4:Y = Y +
     4::HUE = 5: GOSUB 100: POKE  - 16299,0
1030  POKE 230,32:X = X - 2:Y = Y - 2:HUE = 0: GOSUB 100:X = X + 4:Y = Y + 4
1040  NEXT J
1050  POKE  - 16301,0: REM  RESTORE MIXED TEXT AND GRAPHICS MODE
2000  POKE  - 16300,0: VTAB 22: END
```

Program 4. (0C1F) Assembly Language Routine Flip And Move

```
0C00-   AD 54 C0    LDA   $C054
0C03-   A0 00       LDY   #$00
0C05-   84 3C       STY   $3C
0C07-   A9 04       LDA   #$04
0C09-   85 3D       STA   $3D
0C0B-   A9 FF       LDA   #$FF
0C0D-   85 3E       STA   $3E
0C0F-   A9 07       LDA   #$07
0C11-   85 3F       STA   $3F
0C13-   A9 08       LDA   #$08
0C15-   85 43       STA   $43
0C17-   84 42       STY   $42
0C19-   20 2C FE    JSR   $FE2C
0C1C-   AD 55 C0    LDA   $C055
0C1F-   60          RTS
```

Apple Authors:

COMPUTE! is looking for good applications articles on Apple.

Paddle-Adapple

I/O ADAPTOR FOR YOUR APPLE

- COMPATIBLE WITH ALL JOYSTICKS AND PADDLES
- COMPATIBLE WITH ALL SOFTWARE
- GIVES YOU **FOUR** PUSH-BUTTON INPUTS
- UNIQUE "JUMPERS" SOCKET ALLOWS YOU TO CONFIGURE THE **PADDLE-ADAPPLE** TO MEET YOUR SPECIFIC NEEDS
- SELECT ONE OF TWO DEVICES PLUGGED IN or
- USE FOUR PADDLES SIMULTANEOUSLY
- **BPI**™ USERS CAN HAVE **BPI** DEVICE AND PADDLES PLUGGED IN SIMULTANEOUSLY
- EXCHANGE X AND Y AXES IN JOYSTICKS
- ALL STROBES, ANNUNCIATORS, AND POWER LINES AVAILABLE ON BOTH CONNECTORS
- SUPPLIED WITH CABLE (ONE FOOT)
- ADHESIVE FOAM TAPE SUPPLIED
- SUPPORTS SHIFT KEY MODIFICATION

APPLE is a trademark of Apple Computer, Inc
BPI is a trademark of BPI Systems, Inc
PADDLE-ADAPPLE is a trademark of Southern California Research Group

AVAILABLE FROM YOUR COMPUTER DEALER
or
SOUTHERN CALIFORNIA RESEARCH GROUP
POST OFFICE BOX 2231-C
GOLETA, CA 93118
(805) 685-1931

$29.95

Add $1 50 for shipping and handling
CA residents add tax

MASTERCARD AND VISA ACCEPTED
DEALER INQUIRIES INVITED

---

utility city

21 of our most-asked-for Apple Utilities
by Bert Kersey 48K Applesoft ROM

For YOUR Big Apple—Our 21 most-asked-for Apple Utilities on one big disk—**List Formatter** makes properly spaced & indented listings with printer page breaks. Each program statement is on a new line with If-Then's & Loops called out, a great de-bugger! **Catalog in any number of columns** & any page-width to CRT or printer. Automatically post the Run-Number & last-used Date in your programs. **Make any command invisible** in your listings; Access program lines in memory for garbage repair & "illegal" alteration; Quickly sort & store info on disk; Run any Applesoft file while another stays intact; Move lines within your programs, Renumber to 65535; Save inverse, INVISIBLE & trick file names; Convert dec to hex & binary, or INT to FP; Append programs; dump the text screen to ANY printer . . . More too: **21 Programs Total!**

PLUS . . . APPLE TIP BOOK NUMBER THREE!
40 PAGES of new tips, tricks & articles— "DOS Trickery," "Copy Stoppers," "Programming the Reset Key" & much more. Plus understandable explanations of how each U City Program works— One of the best Apple learning tools on the market today!

ALL 21 PROGRAMS **$29.50** +$1 50 Shipping

- ✓ **Utility City** on Applesoft Disk
- ✓ 40-page Beagle Bros Apple Tip Book #3
- ✓ Apple PEEKS, POKES & POINTERS Chart

alpha plot

Hi-Res Apple Graphics/Text Utility
by Bert Kersey & Jack Cassidy 48K REQUIRED

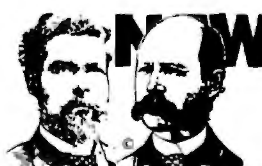
HI-RES DRAWING: Create hi-res pictures & charts, **appendable to your programs.** Keyboard or Paddle control; Optional Xdraw Cursor (see lines before you draw!); Any color mix or REVERSE (opposite of background); Circles, Boxes & Ellipses, filled or not. Bonus Programs too—**SCRUNCHER** stores hi-res in as little as 1/3 normal disk space. **SHIFTER** transfers any portion of the hi-res screen. Also superimpose hi-res images and convert Hi-Res to Lo-Res & back for fascinating abstracts!

HI-RES TEXT: Beautiful upper & lower case with Descenders; color or reverse; Positionable anywhere (NOT restricted by Htabs & Vtabs). Professional looking **PROPORTIONAL SPACING!** Adjustable Type Size, Leading (line spacing) & Kerning (letter spacing). Multi-directional typing; up, down, even backwards!

PLUS . . . APPLE TIP BOOK NUMBER FOUR!
40 PAGES of tips for making the most of your Apple's advanced graphics capabilities. Alpha instructions plus shape table tips, new titling & color tricks and fascinating animation experiments!

$39.50 +$1 50 Shipping

- ✓ **Alpha Plot** on Applesoft Disk
- ✓ 40-page Beagle Bros Apple Tip Book #4
- ✓ Apple PEEKS, POKES & POINTERS Chart



NEW® Apple Utility Disks!

With Free PEEKS & POKES Wall Chart!

Beagle Bros MICRO SOFTWARE

4315 Sierra Vista
San Diego,
CA 92103
714-296-6400

WITH EACH ORDER, YOU WILL RECEIVE . . .

- ☐ **Dos Boss, Utility City or Alpha Plot on Applesoft Disk**
- ☐ **A 40-page Apple Tip Book (each entirely different)**
- ☐ **An 11x17 Apple PEEKS, POKES & POINTERS Chart**

BONUS WITH EACH DISK!
POKE your Apple all night long with this handy 11 × 17 reference poster! The most useable PEEKS, POKES, POINTERS & CALLS scrounged up from every source imaginable! A great companion to our original Apple Command Chart

dos boss

DISK COMMAND EDITOR

by Bert Kersey & Jack Cassidy

A classic utility you will ENJOY! **Rename Commands:** "Catalog" can be "Cat," etc. **Save-Protect your programs;** Unauthorized copy attempts produce "Not Copyable" message. 1-Key Program Selection from catalog (extremely handy!). Catalog Customizer—Change Disk Volume message to your title; Omit or alter file codes; Catalog by file type, etc. Rewrite Error Messages—"Syntax Error" can be "Oops!" or anything you want!

All of Dos Boss's change feaures may be appended to your programs, so that anyone using your disks on any Apple (booted or not) will be formatting DOS the way you designed it!

PLUS . . . APPLE TIP BOOK NUMBER TWO!
With valuable DOS information "Poking Around in DOS," "Making Programs Unlistable," "Two Sided Disk Tips," "Care of Your Drives," "Creative Peeking & Poking," "3 2 vs 3 3," etc

ONLY $24. +$1 50 Shipping

- ✓ **Dos Boss** on Applesoft Disk
- ✓ 40-page Beagle Bros Apple Tip Book #2
- ✓ Apple PEEKS, POKES & POINTERS Chart

MINIMUM 32K REQUIRED

Ask around about Beagle Bros Software.
Our programs are practical, friendly & user-oriented, and always come with an amazing amount of documentation & juicy Apple information. If your dealer doesn't have our stuff, get on his case about it, or order directly from us. We ship all disks & books WITHIN 48 HOURS

24 hour TOLL FREE Order Desk:
MasterCard, VISA or COD orders, call:

Nationwide:	**800-854-2003** ext. 827
California:	**800-522-1500** ext. 827
Alaska/Hawaii:	**800-854-2622** ext. 827

(Orders only please. Questions, phone 714-296-6400.) COD orders, add $3.)

Or order by mail (use this coupon or separate sheet)

(Use this coupon or separate sheet!)

RUSH! The disks checked below
Plus The Tip Book and PEEKS & POKES Chart!

☐ **UTILITY CITY** ($31) ☐ **ALPHA PLOT** ($41) ☐ **DOS BOSS** ($25.50)

(Prices include First Class Shipping; Calif residents add 6%.)

Specify:
NAME$= ______________ ☐ **3.2** ☐ **3.3**
ADDR$= ______________
CITY$= ______________ **ZIP=** ______

VISA **VISA/MASTERCARD: Include Account No. & Expiration Date.** master charge

Mail U.S. check, money order or Visa/MC Numbers to: **BEAGLE BROS, Dept. S**
4315 Sierra Vista,
San Diego, CA 92103
Telephone (714) 296-6400.

Please add $4.50 for shipping outside US or Canada.
COD orders, add $3. California residents add 6%.

Programming The RESET Key The Easy Way

Richard Cornelius, Wichita, KS

On the Apple Computer the RESET key, to most users, is a magical key that provides an instant means to get out of any program. Usually a person begins to modify the RESET function only after learning machine language. Here is a method of making the RESET key do anything (well, almost anything) that you want it to do on an Apple II Plus, and you don't need *any* knowledge of machine language.

First let's examine what the RESET key does. When the RESET key is pressed the currently running program is interrupted; the screen display is set to text page 1; output to the screen is set to NORMAL; the text window is set to the complete screen; the cursor is moved to the bottom of the page; a beep sounds; accessory I/O is shut down; and then the computer looks in locations 1010 and 1011 in memory to see where it should go next for instructions. When the computer is turned on, the contents of these two locations are automatically set such that when RESET is pressed the computer is returned to immediate mode in BASIC. Changing these locations to make the computer go to different places for instructions involves only POKEs to positions 1010 and 1011 and a CALL-1169.

Where should the computer be sent? Starting at position 768 there is some room that is reserved for short machine language programs, and that is where we shall send it. (Don't worry — you don't need to know any machine language.) POKEs to seven bytes are used to make the RESET key run a BASIC program starting at the *second* line of code. When the first line of the program makes the program jump around the second line, then the second line will *only* be executed when RESET is pressed.

The program will help you understand how the RESET key can be used to execute any BASIC statements that can be put into a program. When the program is RUN, statements 110 through 190 are jumped over so that lines 200 through 260 are the first statements in the program that perform any tasks. These lines fix the RESET key so that the computer will go to line 110 when the RESET key is pressed. The length of the very first statement is critical. As long as it has a three-digit number after the GOTO, the RESET key will operate as desired. Changes in the length of the statement will likely mean that the RESET key will send the computer to some nonsense location. Placing a REM statement (or any other statement) before line 100 will have the same effect. Modifying the DATA statement in line 230 to accommodate changes in the length of that first statement is not difficult, but, unless you understand what to do, you had better not make any changes.

Lines 270 through 310 constitute a dummy BASIC program to show that the program is being RUN. Statements 110 through 190 tell the computer what to do when the RESET key is pressed. Lines 140 through 190 can be changed to make the RESET key do whatever you want it to do. In this example, the program is simply rerun from the beginning, but you can make lines 140-190 do whatever you wish. Lines 120 and 130 should not be changed since they fix up some things that are undone by the short machine language program that is POKEd in, but omit line 130 if you don't have a disk drive. If you should want to "turn off" the changes to the RESET key so that it behaves normally, simply POKE 1010,3: POKE 1011,244: POKE 1012,69 if you have no disk drive or POKE 1010,191: POKE 1011,157: POKE 1012,56 if you do have a disk drive.

For those who don't wish to stray from BASIC, this short program contains all that is needed to make the RESET key do almost anything. Take an existing program and add it starting at line 280 to the program. In lines 140-190, put statements that you wish to be executed when the RESET key is pressed. You can thus program the RESET key in BASIC without knowing any machine language. For those who are interested in straying just a little from BASIC, the final paragraphs explain the details of what is happening.

Positions 1010 and 1011 (hex 3F2 and 3F3) contain the low and high bytes of the location that the RESET key makes the computer jump to after it performs a fixed set of operations. The POKEs in statement 210 change this location from 40383 (hex 9DBF) to 768 (hex 300). Before the computer performs this jump, it looks at the "power-up" byte, position 1012 (hex 3F4), to see whether the value at this location equals an exclusive OR of the value in position 1011 (hex 3F3) with the constant 165. If the values correspond properly, the computer believes that it has *not* just been turned on and it executes a jump to the specified location. If the values do not properly correspond, the computer thinks that it has just been turned on, and it will attempt to reboot the disk if a disk controller card is present. The CALL-1169 in statement 210 properly sets this power-up byte.

The DATA statement in line 230, coupled with the POKE statement in the FOR...NEXT loop in lines 240 through 260, puts a very short machine language program into memory. This program is shown below:

```
0300-  A9 0A      LDA  #$0A
0302-  85 67      STA  $67
0304-  4C 66  D5  JMP  $D566
```

The first two statements in this program place the value 10 (hex 0A) into location 103 (hex 67). Position 103 is the low byte (and position 104 is the high byte) of the starting location of the current BASIC program. The first statement in the program is "100 GOTO 200" and occupies 9 bytes: 2 bytes for the location of the next line, 2 bytes for the statement number, 1 byte for the GOTO token, 3 bytes for the digits of the number 200, and 1 byte for a terminating 0. Normally location 103 would contain the value 1, so adding 9 to this value makes the computer think that the BASIC program begins at the second line. To see that this works, enter the BASIC program and then POKE 103,10. If you list the program after this POKE, the list will begin with line 20. POKEing 103 with the value 1 will restore the program to begin with statement 10.

The final line in this machine language program jumps to 54630 (hex D566) where the RUN routine in firmware Applesoft BASIC resides. Since the value in location 103 (hex 67) has been changed, the RUN command executes the BASIC program starting at line 110. Once the program is running, the section that can only be accessed by the RESET key, the value in location 103 is changed back to its standard value so that the RUN command in line 190 will RUN the program starting with the first line of the BASIC program.

Many variations on this general scheme are possible. By making the RESET key RUN statements of BASIC code, changing the RESET key function becomes an easy adaptation to add to any BASIC program.

```
100 GOTO 200
110 REM **HERE IS WHERE THE RESET KEY
          SENDS THE COMPUTER**
120 POKE 103,1
130 CALL 1002
140 HOME
150 VTAB 3
160 PRINT "YOU HAVE PRESSED THE RESET
    KEY."
170 PRINT: PRINT "I WILL NOW RERUN THE
    PROGRAM."
180 FOR PAUSE=1 TO 2000:NEXT
190 RUN
200 REM **MAKE THE RESET KEY GOTO
          SECOND STATEMENT**
210 POKE 1010,0: POKE 1011,3:
    CALL -1169
220 REM -THE ABOVE STATEMENT RESETS
      "JUMP TO" LOCATION FOR RESET
230 DATA 169,10,133,103,76,102,213
240 FOR SPOT = 768 TO 774
250 READ CODE: POKE SPOT,CODE
260 NEXT
270 REM **PLACE BODY OF PROGRAM HERE**
280 HOME
290 VTAB 3
300 PRINT "THE PROGRAM IS NOW RUNNING."
310 GOTO 310
```

A Simple Printer Interface For The Apple II

Marvin L. DeJong
Dept. of Math-Physics
The School of the Ozarks
Pt. Lookout, MO

In the January 1981 issue (**COMPUTE!** #8) I described a simple circuit that could be used with an Apple II to perform the experiments in my book[1]. The circuit provided one eight-bit output port. These two ports can also be used to interface the Apple II to a parallel port printer.

The Circuit

For the unfortunates who do not have a copy of **COMPUTE!** #8, I have included the circuit diagram of the peripheral I/O card in this article. It is shown in Figure 1. My circuit was wire wrapped on a Vector Electronic #4609 plugboard which fits into the peripheral card connectors on the Apple II. For the purpose of this application, the eight LEDs and the DIP switch (with pull-up resistors) are not needed. They are only used if you wish to use the peripheral I/O board in conjunction with the experiments in my book. You may also wish to experiment with the possibility of omitting the 74LS242 bus transceivers and the associated logic, simplifying the circuit further. This would leave only the 74LS138, an inverter, the two 74LS75s, and the 81LS97. Since only one bit of the input port is used to interface to the printer, you may wish to replace the 81LS97 with a 74LS125. I used the circuit as it is shown in Figure 1, with the DIP switch removed from the socket.

My printer (which was not the one used to make the listings in this article) is a MICROTEK MT-80P which I normally use to interface to one of my TRS-80 machines. It claims to have a "Centronics-compatible interface," so perhaps the circuit and software we describe here may also be used with Centronics printers. The printer has eight data lines and several handshaking lines. The eighth bit is not used by the printer: it uses seven-bit ASCII. So seven bits of the output port on our peripheral I/O card are used to send the character to be printed to the MT-80P printer.

Two handshaking lines are used, DATA STROBE and BUSY. The microcomputer must supply a logic-zero pulse (strobe) of at least one microsecond in duration when the character on the data lines is to be read by the printer. Thus, the DATA STROBE line is controlled by the Apple II peripheral I/O card. In particular, I used bit zero (PA0 in Figure 1) to control the DATA STROBE line, while the seven-bit ASCII character appears on bits one to seven (PA1 - PA7). When the DATA STROBE pulse is sent, the printer responds by bringing the BUSY line to logic one. It stays at logic one until the character is read. This will only take about 40 microseconds unless the buffer is full. The BUSY line will stay high until there is room in the buffer. Thus, the BUSY line is connected to bit seven of the input port on the peripheral I/O card where it may be watched with a BMI instruction. Figure 2 shows the connections to the printer, and Figure 3 illustrates the handshaking sequence.

The Software

The machine language software driver routine is shown in Program 1. It is used with DOS 3.3, but other versions should work equally well. The machine language program consists of two parts. The first part starts with line six in the listing and ends with line 19 (locations $02C0 – $02DB). It has two functions:

1. It sets up the Apple II output registers ($36 – $37) to point to the printer routine at $02E0, and it jumps to a DOS routine to fix the DOS output register. (See pages 103-104 in the APPLE II DOS MANUAL.)

2. It loads a form-feed character, $0C, into the printer and pulses the DATA STROBE line.

The second part of the machine language routine is the actual print routine. It puts an ASCII character on the data lines to the printer and then it pulses the DATA STROBE line, but it does not do this unless the BUSY line is at logic zero, indicating that the printer is not busy. Finally, it jumps the monitor COUT routine that prints the character on the video monitor screen, before returning to the DOS program.

In Program 2 I show a greeting program that is the INITilization program on the slave diskette for our DOS 3.3. It gives the user the chance to call PRINTS, the object code file that is also stored on the slave diskette. This completes the description of the software for this system. Refer to the comments for more details regarding the software.

If you are not running a disk system, then to operate the printer load the machine language programs in Program 1 with a single modification. Replace the JMP DOSSYS instruction with a BRK.

Execute the program from the monitor, starting at $02C0. You can then either stay in the monitor or return to BASIC with a control B.

Notice that the software is located in page two of memory. If you type in a very long sentence you may wipe out your program, since it is part of the input buffer for the Apple II. Ideally, you would PROM the software. (We should add that the software as shown assumes that the peripheral I/O card is in *slot one* on the Apple II. The software, assuming the peripheral I/O card is in slot one, would be loaded into locations $C100 upward, *starting with the instructions at $02E0 in Program 1*.

To initialize the printer you would still want to execute instructions from $02C8 through $02D8, with a BRK replacing the JMP DOSSYS at location $02D9. Thereafter a PR#1 command would produce an active printer, and a PR#0 would disable it. I should add that I have *not* tried to run the system with the program in EPROM, but I think that I understand my Apple II enough to make the instructions just given. I would very much like to hear from someone who might try this approach.

Obviously this interface circuit will work with almost any microcomputer system and any parallel printer. Even the software requires little modification to work with any 6502 based system. The card to mount the components is the most expensive item, $23.25. Note that the card I used has another edge connector not used to plug into the Apple II, and I used that connector to attach to our printer cable. It accepts a standard 20/40 edge connector, but my printer used a 19/36 edge connector, so I sawed and filed to fit. It has a big price advantage over the usual parallel interfaces found in your catalogs.

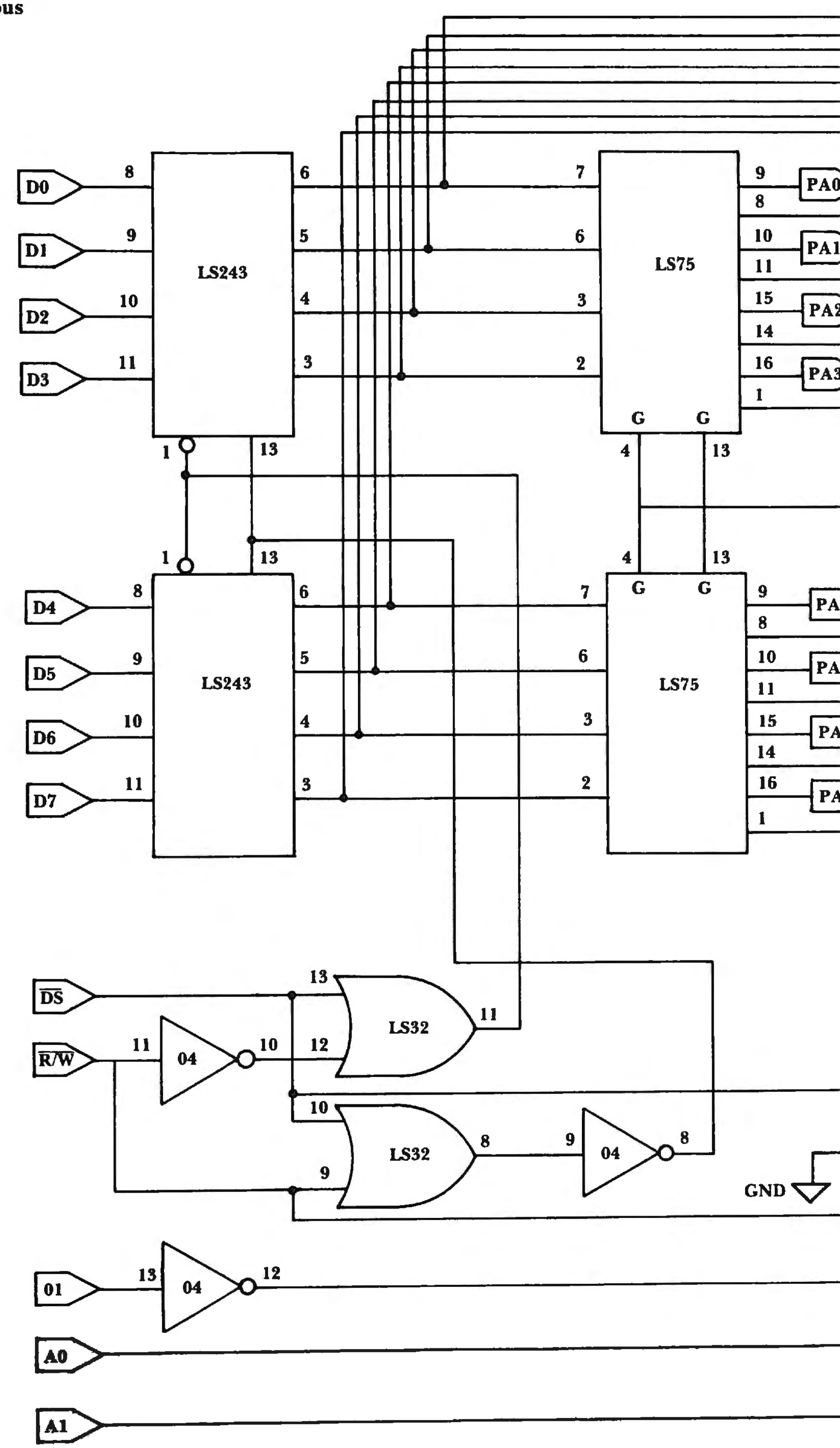
Reference

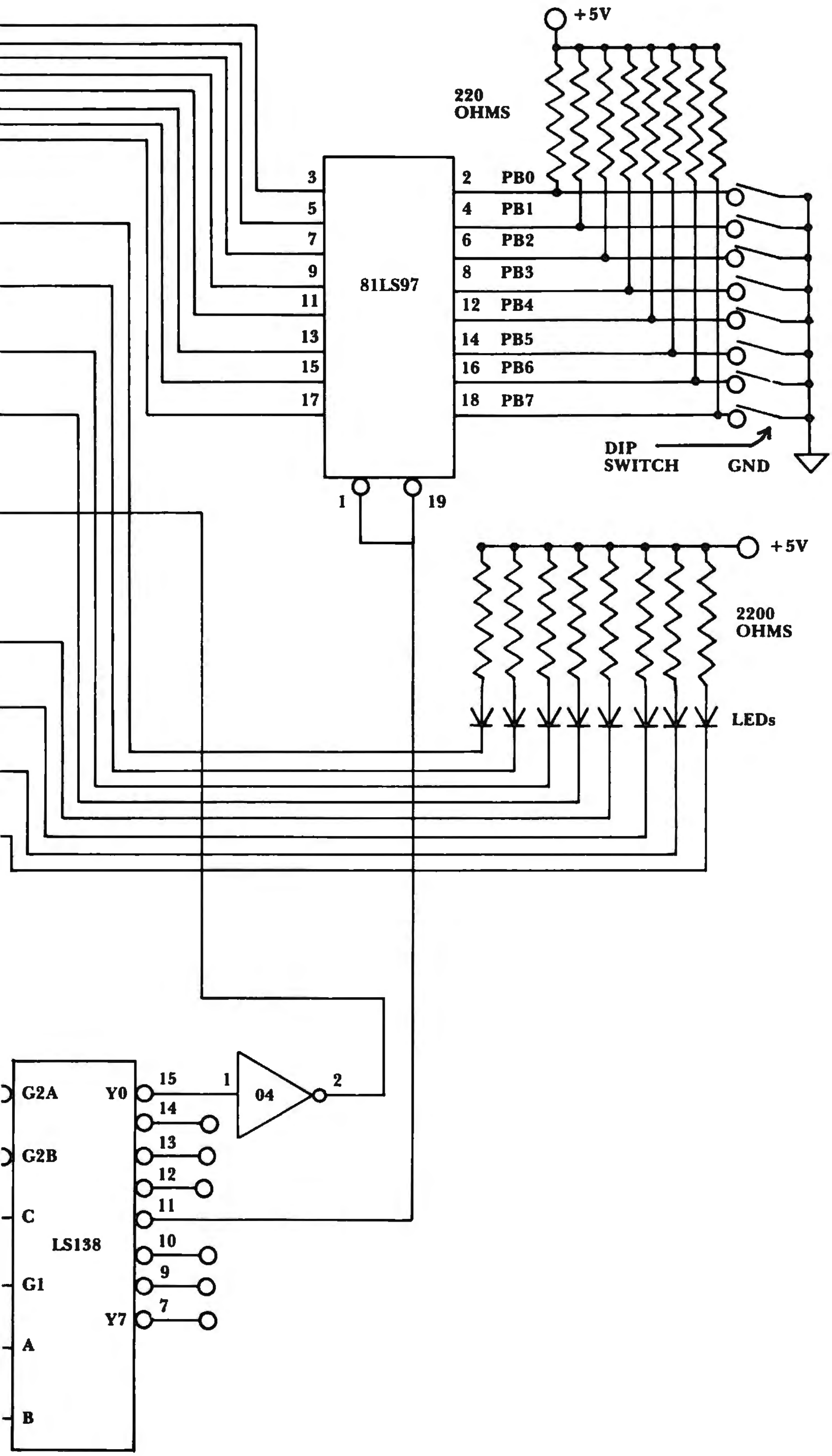
[1]M. L. De Jong, *PROGRAMMING & INTERFACING THE 6502, WITH EXPERIMENTS*, Howard W. Sams & Co., Inc. 4300 West 62nd St., Indianapolis, Indiana 46268, 1980. $15.95

Program 1.

```
SOURCE FILE: PRINTS
0036:                1 APREGLO EQU   $0036
0037:                2 APREGHI EQU   $0037
C090:                3 OUTPORT EQU   $C090
C094:                4 INPORT  EQU   $C094
03EA:                5 DOSSYS  EQU   $03EA
----- NEXT OBJECT FILE NAME IS PRINTS.OBJ0
02C0:                6         ORG   $02C0
02C0:A9 E0           7 INITIAL LDA   #$E0       ;SET UP APPLE OUTPUT REGISTERS
02C2:85 36           8         STA   APREGLO    ;TO POINT TO PRINTER ROUTINE
02C4:A9 02           9         LDA   #$02
02C6:85 37          10         STA   APREGHI
02C8:A9 0C          11         LDA   #$0C       ;SEND FORM FEED TO PRINTER
02CA:38             12         SEC
02CB:2A             13         ROL   A          PUT CHARACTER IN HIGH ORDER 7 BITS
02CC:8D 90 C0       14         STA   OUTPORT    ;OUTPUT THE CHARACTER WITH BIT ZERO AT
 LOGIC ONE
02CF:29 FE          15         AND   #$FE       ;NEXT BRING BIT ZERO TO LOGIC ZERO TO
SEND DATA PULSE
02D1:8D 90 C0       16         STA   OUTPORT
02D4:09 01          17         ORA   #$01       ;BRING BIT ZERO TO LOGIC ONE AGAIN
02D6:8D 90 C0       18         STA   OUTPORT
02D9:4C EA 03       19         JMP   DOSSYS     ;JUMP TO DISK ROUTINE TO EXCHANGE OUTP
UT REGISTERS
02DC:EA             20         NOP
02DD:EA             21         NOP
02DE:EA             22         NOP
02DF:EA             23         NOP
02E0:48             24         PHA               ;SAVE CHARACTER
02E1:AD 94 C0       25 BUSY    LDA   INPORT     IS PRINTER STILL BUSY?
02E4:30 FB          26         BMI   BUSY       ;YES, THEN DONT BOTHER IT
02E6:68             27         PLA               GET CHARACTER BACK
02E7:48             28         PHA               AND SAVE IT AGAIN
02E8:38             29         SEC               SET CARRY TO ROTATE A ONE INTO BIT ZE
RO
02E9:2A             30         ROL   A          MOVE CHARACTER UP ONE BIT
02EA:8D 90 C0       31         STA   OUTPORT    OUTPUT IT
```

Figure 1. Peripheral Card Data bus




+5V
220
OHMS
81LS97
3
5
7
9
11
13
15
17
2 PB0
4 PB1
6 PB2
8 PB3
12 PB4
14 PB5
16 PB6
18 PB7
DIP
SWITCH
GND
1
19
+5V
2200
OHMS
LEDs
G2A
Y0
G2B
C
LS138
G1
Y7
A
B
15
14
13
12
11
10
9
7
1
04
2

```
02ED:29 FE          AND  #$FE     ;PULSE DATA LINE
02EF:8D 90 C0       STA  OUTPORT
02F2:09 01          ORA  #$01
02F4:8D 90 C0       STA  OUTPORT
02F7:68             PLA            GET CHARACTER BACK FOR SCREEN OUTPUT
02F8:4C F0 FD       JMP  $FDF0    ;JUMP TO COUT ROUTINE IN THE MONITOR

*** SUCCESSFUL ASSEMBLY: NO ERRORS
```

Program 2.

```
]LISTLIST
?SYNTAX ERROR
]LIST

5   REM  GREETING PROGRAM
10   PRINT "SLAVE DISKETTE CREATED ON
     32K SYSTEM"
15   PRINT : PRINT : PRINT : PRINT :
     PRINT
21   INPUT "DO YOU WANT THE PRINTER ON?
     (TYPE Y OR N.) ";A$
22   IF A$ = "Y" THEN 30
23   GOTO 50
30 D$ =  CHR$ (4)
40   PRINT D$;"BRUN PRINTS"
50   END
```

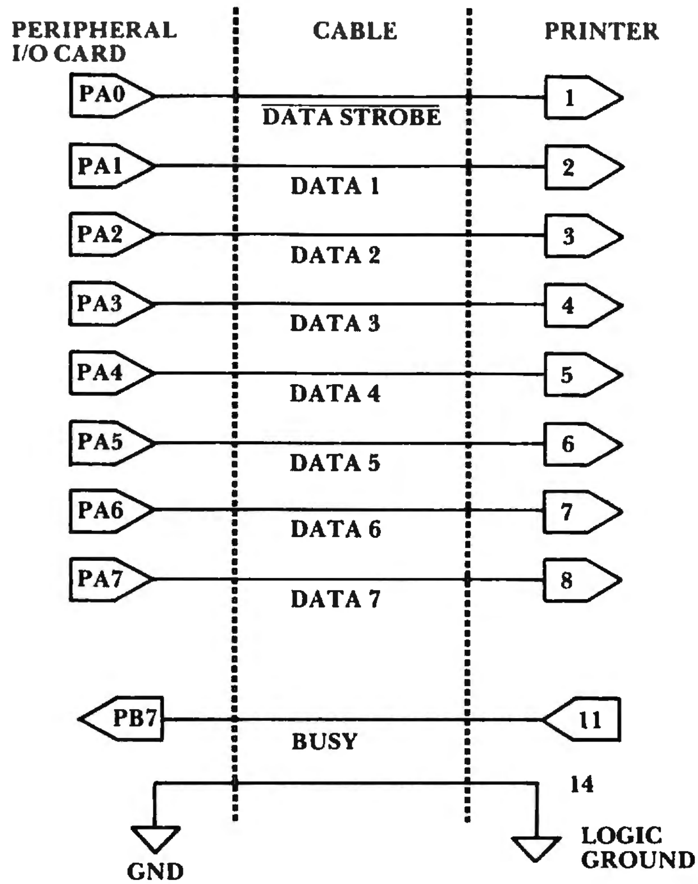


Figure 2. Interface between the peripheral I/O circuit and the printer.

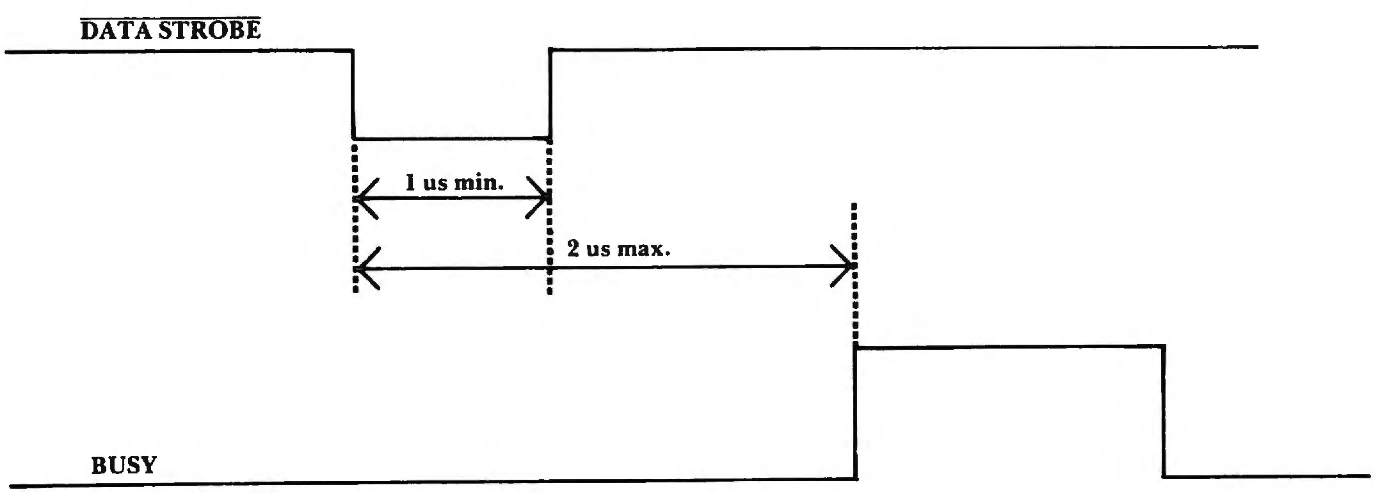


Figure 3. Microcomputer-Printer handshaking sequence.

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
Empty / 32K / 16K	48K RAM 40K With BASIC Cartridge	48K RAM 40K With BASIC Cartridge
Empty / 32K / 8K	40K RAM	Danger! This Configuration Can Damage Computer
Empty / 16K / 32K	48K RAM 40K With BASIC Cartridge	Danger! This Configuration Can Damage Computer
Empty / 8K / 32K	40K RAM	Danger! This Configuration Can Damage Computer
8K / 32K / 8K	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

INSIGHT: Atari

Bill Wilkinson
Optimized Systems Software
Cupertino, CA

Last month, we tackled some of the fundamentals of I/O under Atari's OS. This month we will look at the extended disk operations available and will try our hand at writing a useful program in assembly language.

There simply isn't space to repeat the charts given in last month's article, so you will have to open to those pages: we will be referring to them often.

Atari I/O, Part 2: Disk File Manager

Notice that the title of this section is *not* "ATARI DOS." There is a simple reason, which I expounded on before: Atari does not have a DOS. (But please don't tell them I said so; they think they have to call it "DOS," because that's what everybody else calls it.) Atari *has* an "OS"; actually a much more powerful system than what is normally called "DOS" on microcomputers. And please recall from last month that the Atari OS understands *named* devices, such as "P:" and "E:". The Disk File Manager (DFM) is actually simply a device driver for the disk ("D:") device. It was written completely separately from Atari OS and interfaces to OS the same way any other driver does. In fact, there is nothing magic about the DFM. In theory, by the end of next month's article you should know enough about Atari OS and the DFM to implement your own File Manager and to replace the one that Atari supplies you. (*In theory*. In practice, you had better know the principles of disk space allocation, I/O blocking and deblocking, and much more, before tackling such a job.) Even if you aren't quite that ambitious, we hope that this series will give you some "insight" into how such things as BASIC's I/O are implemented.

Extended Disk Operations

We should first note that most of the extended disk operations are documented in the Atari *Basic Reference Manual* in the section about the XIO command. There are two exceptions, NOTE and POINT, which were given special BASIC commands (and we will see why soon). Naturally, the Atari *Disk Operating System II Reference Manual* is pertinent, but it doesn't really give more information about the internal workings of Atari's OS than does the BASIC manual. Before delving into assembly language, let's examine each of the extended disk operations in a little detail:

ERASE, PROTECT, UNPROTECT — Also known as Delete, Lock, and Unlock, these three commands simply provide OS with a channel number (i.e., the X-register contains IOCB number times 16), a command number (ICCOM), and a filename (via ICBAL/ICBAH). When OS passes control to the DFM, an attempt is made to satisfy the request. Note that the filename may include "wild cards," as in "D:*.??S" (which will affect all files on disk drive one which have an 'S' as the last letter of their filename extension).

RENAME — Very similar to ERASE, et al, in usage. The only difference is in the form of the filename. Proper form is:

"Dn:oldname.ext,newname.ext"

Note that the disk device specifier is not and *cannot* be given twice.

NOTE, POINT — Other than OPEN, these are the only commands we have encountered so far (including last month) which use any of the AUXilliary bytes of the IOCB. For these commands, one specifies the channel number and command number and then receives or passes file pointer information via three of the AUX bytes. ICAX3/ICAX4 are used as a conventional 6502 LSB/MSB 16-bit integer: they specify the current (NOTE) or the-to-be-made-current (POINT) sector within an already OPENed disk file. ICAX5 is similarly the current (NOTE) or to-be-made-current (POINT) byte within that sector. These are complex commands to use, but their operation from BASIC is adequately covered in the *Atari DOS II Manual* so it will not be covered here.

OPEN — Open is not truly an extended operation, but for disk I/O we need to know that the DFM allows two additional "modes" beyond the fundamental OS modes (which are 4, 8, and 12 for read, write, and update). If ICAX1 contains a 6 when DFM is called for OPEN, then the disk DIRECTORY is opened (instead of a file) for read-only access. The filename now specifies the file (or files, if wild cards are used) to be listed as part of a directory listing. Note that DFM expects this type of OPEN to be followed by a succession of GETREC (get text line) OS calls (and we present an example of this below). If ICAX1 contains a 9, the specified file is opened as a write-only file, but the file pointer is set to the current end-of-file. Caution: DFM only appends on sector boundaries (normally this is transparent to the user, but *caveat artificer*).

Error Handling

This may not be the best place to introduce this topic, but the information is needed for examples which follow. Space doesn't permit a listing of all the I/O error codes, so we must refer you again to the BASIC and/or DOS II reference manuals. There are four fundamental kinds of errors that can occur with Atari OS:

HARDWARE ERRORS — Such as attempting to read a bad disk, write a read-only disk, etc.

SERIAL BUS ERRORS — Errors which occur when data is transferred between the computer and a peripheral device. Examples include Device Timeout, Device NAK, Framing Error, etc.

DEVICE DRIVER ERRORS — Found by the driver for the given device, as in (for the DFM) File Not Found, File Locked, Invalid Drive Number, etc.

OS ERRORS — Usually fundamental usage problems, such as Bad Channel Number, Bad Command, etc.

On return from any OS call, the Y-register contains the completion code of the requested operation. A code of one (1) indicates "normal status, everything is okay." (I know, why not zero, which is easier to check for? Remember, I said Atari was good, not perfect.) By convention, codes from $02 to $7F (2 through 127 decimal) are presumed to be "warnings." Those from $80 to $FF (128 through 255 decimal) are "hard" errors. These choices facilitate the following assembly language sequence:

```
JSR CIOV     ; call the OS
TYA          ; check completion code
BMI OOPS     ; if $80-$FF, it must be an error
```

In theory, Atari's OS always returns to the user with condition codes set such that the TYA is unnecessary. In practice, that's probably true; but a little paranoia is often conducive to longer life of both humans and their programs.

A Real, Live Example

Believe it or not, you now have all the information you need to do from assembly language any and all I/O done by Atari BASIC and/or BASIC A+ (excepting graphics, but that's coming...hold your breath). In an attempt to make you believe that statement, we will write a program in both BASIC and assembly language.

The BASIC Program

```
100 DIM BUFFER$(40)
200 OPEN #1,6,0,"D:*.*"
300 TRAP 700
400 INPUT #1,BUFFER$
500 PRINT BUFFER$
600 GOTO 400
700 CLOSE #1
```

This program will list all files on disk drive one (D1:) to the screen. This is exactly equivalent to using the "A" option of Atari's menu "DOS" (and then hitting RETURN for the filename) or to using "DIR" from OS/A+. Admittedly, this program is easily improved. For example, replace line 200 with:

```
200 INPUT BUFFER$ : OPEN #1,6,0,BUFFER$
```

and now you can choose to list only some files. You might also wish to send the listing to the printer (change PRINT to LPRINT). However, we will leave such changes as an exercise to the reader and discuss only our simplified version.

Please now refer to the listing in Program 1. Since it follows the scheme of the above BASIC listing, it is almost self-explanatory. A few words are in order, though. The equates at the beginning have been kept to a minimum; I refer you to the "SHOOT" listing in **COMPUTE!** #16 if you want a comprehensive list. (The mnemonics used are not all identical to those in the "SHOOT" listing; those shown are from our standard equates file.)

The program is intended to be called from BASIC via the USR function. However, no check is performed to see if the BASIC program were coded as (for example) PRINT USR(1600,0) instead of just PRINT USR(1600). (Note that 1600 decimal = 640 hex, the starting address.) If you would like to test this program with the BUG debug monitor, you should replace the RTS at the end of the program with a BRK before saying 'G641' (641 to avoid the PLA).

All errors, including an error on the OPEN DIRECTORY call to OS, are treated as end-of-file. A better program would verify the error status and print a message or some such. As an example of a

LOOK WHAT YOUR ATARI WANTS FOR CHRISTMAS.

Before this Christmas is past, give a Cybersoft™ present.

This Christmas, treat the ATARI in your house to the very best in software. Your ATARI leads the way in micro-computer excellence. So should your software.

We have the "best of the best" programs from the Grand Masters of micro-computer software: Automated Simulations, Crystal, Swifty Software, Adventure International, Datasoft, Personal Software, and Dynacomp. All sold by mail. And, with the CYBERSOFT ironclad guarantee*.

Space Chase *(Swifty Software) 16K* Nifty eye-hand coordination tests... If you clear one galaxy of "planets" and avoid destruction at the hands of an alien ship you are rewarded by an attack by two aliens and then three. Game keeps score and displays high score. Colorful graphics and good playing. Cassette $14.95

Fastgammon™ *(Quality Software) 8K* The best of the computer backgammon games. You can repeat games with the same dice rolls to try different strategies. You roll dice or let computer roll for you. This machine language program challenges experienced and beginners alike. Underutilizes ATARI® graphics but playing quality more than makes up. Cassette $19.95

(NEW!) Temple at Apshai (Automated Simulations) 32K Just released for ATARI, the program many believe to be the best of the role-fantasy games now available. You will encounter as many as 30 monsters in more than 200 rooms. Select from 18 different actions: fighting to evesdropping. Cassette $39.95

(NEW!) Crush, Crumble and Chomp (Automated Simulations) 32K Tired of destroying countless monsters? This one is for you. You are the monster. Pick from six. You may be the giant amphibian, Goshilla. Knock down the Golden Gate bridge, or tip over the Empire State Building. But look out for the National Guard. You pick the goal and see if you can make it. Five objectives. Cassette $29.95

Bridge 2.0 *(Dynacomp) 24K* Now you can practice your bridge at home before you have to embarrass yourself at the neighbors. Bridge 2.0 bids (Goren) and plays both contract and duplicate bridge. It even doubles if you get carried away with your bidding. No graphics, but clearly arranged format makes playing easy. Cassette $17.95

Nominoes Jigsaw *(Dynacomp) 24K* A 60 piece jigsaw puzzle you complete on your screen. Three levels of difficulty to confound you. Scoring is a function of how many tries and how difficult the puzzle. Cassette $17.95

Star Trek 3.5 *(Adventure International) 24K* Lots of color, lots of action as you hyperwarp through the quadrants in your search for Klingons. This program will satisfy the "trekkie" in us all. Cassette $14.95

Adventureland *(Adventure International) 24K* Designed for the novice at adventuring and a great way to get hooked on the world of adventure. As you make your way through an enchanted world you will encounter wild animals, spell-ridden critters and more. Your task is to avoid the perils as you try to collect the 13 lost treasures. Cassette $14.95

Mission Impossible *(Adventure International) 24K* Another adventure from the great Scott Adams. Beginning "Good morning, your mission is...", your task is to save a nuclear reactor from destruction. This adventure game is not easy. Cassette $14.95

(For Your Business.)

File Manager 800 (Synapse) 40K The premier data base management system available for ATARI. Not a translation. File Manager fully utilizes the special keyboard of ATARI. It creates files, sorts data, prints lables to your specifications. The full documentation is clear and easily understood. We think this one is fully competitive with data base systems at several times the price. Disk, 825 Printer $94.95

VisiCalc™ *(Personal Software) 32K* If you use your ATARI for any business purpose or for your personal finances, you will want VisiCalc. It probably sold more computers than any program other than Star Raiders. VisiCalc creates a very large accountants ledger pad upon which you write with your ATARI. The fundamental difference is, if you change any figure(s) the balance of the figures are automatically adjusted. Extensive documentation outlines the full features of this fantastic business tool. Diskette $199.95

Text Wizard™ *(Datasoft) 32K* ATARI owners have waited a long time for a high quality, full blown word processor. This easy-to-use and comparatively inexpensive program has received raves from users across the country. This program fully utilizes the unique ATARI keyboard and ease of cursor movement. Justifies right, left and center. Pagination. Word search and substitute. Diskette and ATARI 825, Centronics 737 or Epson MX-80 printer. $99.95

*CYBERSOFT Ironclad Guarantee.

We fully guarantee the software we sell. If for any reason, you are not satisfied with your purchase, return the program within 30 days in its original packaging. We'll refund your money, no questions asked.

Our order system is as accurate and fast as our software. Just dial

1-800-247-2000 EXT. 505

(In Iowa 1-800-362-2133, extension 505). Or, you can write to CYBERSOFT at P.O. Box 505 Bellevue, Washington 98009. VISA and MasterCard gladly accepted.

Include $2.00 handling charge per order. Washington residents add 5.4% sales tax.

cybersoft

minor improvement, at LINE700 one could save the Y-register (status) value in FR0 and zero in FR0+1 ($D4 and $D5), thus returning the error code to the calling BASIC program.

Notice that values stored into the IOCB for FILE0 (the console screen output) were stored directly into ICCOM, etc., without an X-register offset. This is perfectly valid, so long as the X-register contains the proper value on calling CIO. In fact, we could have stored the values for FILE1 (the directory) by coding (for example) STA ICCOM+FILE1. Obviously, this technique only works when one uses a constant channel number; but most BASIC programs and many language programs can use predefined channel numbers.

There isn't really much more to say other than, "Try it!" It really does work. And, even if you don't understand the concepts on first reading, actually entering the program and following the program flow and remarks might give you a painless introduction to I/O from assembly language.

The Easiest Way Of Making Room?

With an ATARI 400 or 800, there are many ways and places to find "safe" hunks of memory, places to put assembly language routines, player/missile graphics, character sets, etc. Many of the programs that I have seen involved techniques that I consider risky. For example, moving BASIC's top of memory down requires that one do so only after issuing a GRAPHICS, command for the most memory-consuming graphics mode used in the program.

Other programs use machine language subroutines; but such subroutines must themselves have a place to stay. The best of such routines, however, approach the "official" Atari method. The approved method is normally used (by Atari) to add device drivers to the OS; in fact, the drivers for both DOS and the RS-232 ports follow these rules:

1. Inspect the system LOMEM pointers.
2. Load your routine (or reserve your buffer) at the current LOMEM.
3. Add the size of the memory you used to LOMEM and
4. Store the resultant value back into LOMEM.

If each routine, driver, etc., followed these rules, one could reserve more and more of memory without disturbing any following routine. (In fact, Atari drivers presume that LOMEM will never grow beyond 16K, $4000, or even less; but the principle holds.) Actually, there's a hole in the above method: if the SYSTEM RESET button is pushed, OS goes through and resets all its tables, including the value in LOMEM. A "good" device driver can even take this into account, but we are going to make a few presumptions that are generally valid.

By now, you should realize that all of BASIC's fundamental I/O commands are simply implementations of OS calls. PRINT becomes PUT TEXT RECORD; INPUT becomes GET TEXT RECORD; OPEN and CLOSE are essentially unchanged. In fact, the only BASIC commands that are not obvious clones of their assembly language counterparts are GET and PUT. Suffice it to say that these are actually simply special case implementations of GET BINARY RECORD and PUT BINARY RECORD (commands 7 and 11) where the buffer length is set to one byte.

Next month, we tackle the task of understanding how device drivers work, and we actually write a new and useful one that talks to a device built into *all* Atari machines (but one that Atari didn't provide a driver for). And we haven't forgotten the promise to show how graphics routines (such as PLOT and DRAWTO) are actually I/O routines.

The trick: BASIC always, repeat always, LOADs new programs at what it perceives LOMEM to be! Unfortunately, BASIC keeps its own MEMLOW pointer, which is loaded from LOMEM only on execution of a NEW, not on execution of LOAD or RUN and (significant!!!) not even in the case of SYSTEM reset. However, when there's a will...

```
— ATARI BASIC —
10 LOMEM = 743 : MEMLOW = 128
20 ADDR = PEEK(LOMEM) + 256 * PEEK
   (LOMEM+1)
30 ADDR = ADDR + SIZE
40 HADDR = INT( ADDR / 256 ) : LADDR = ADDR
   - 256 * HADDR
50 POKE LOMEM,LADDR : POKE LOMEM+1,
   HADDR
60 POKE MEMLOW,LADDR : POKE MEMLOW+1,
   HADDR : RUN "D:PROGRAM2"
— BASIC A+ —
10 lomem = 743 : memlow = 128
20 addr = dpeek(lomem) : dpoke lomem, addr+size
30 dpoke memlow, addr+size : run "D:PROGRAM2"
```

The above listing is Program A, whose only purpose in life is to set up memory for the real program, Program B. "SIZE" is the amount of memory to be reserved. The program changes both the system and BASIC bottom-of-usable-memory pointers so that either NEW or RUN "..." will recognize the reserved memory. The beginning lines of PROGRAMB follow:

```
— ATARI BASIC —
10 LOMEM = 743 : MEMLOW = 128
20 POKE LOMEM, PEEK(MEMLOW) : POKE
   LOMEM+1, PEEK(MEMLOW+1)
— BASIC A+ —
10 dpoke 743, dpeek(128)
```

The only reason for these lines in PROGRAMB is in case of SYSTEM RESET. If the user types RUN after the reset, BASIC will copy its MEMLOW (the value which includes the reserved space!) into the system's LOMEM, just so they agree with each other. A caution: I don't know what will happen if you hit SYSTEM RESET as BASIC is in the process of loading PROGRAMB.

As far as I can tell, the only real problem that could occur would be if SYSTEM RESET were followed by a "DOS" command from BASIC. The OS would then get control, thinking that LOMEM had not been changed. In a normal running program environment, though, this is, at worst, unlikely, so this method seems more than adequate.

Columnar Output

A problem inherent in Atari BASIC is that the default tabbing (when using 'PRINT exp,exp') is ten columns while the screen is 38 columns wide. This produces an output something like this:

```
PRINT 1,2,3,4,5,6,7,8,9,10

1         2         3         4
 5         6         7         8
  9         10
```

Not too pretty. POKE 82,0 will change the left margin of the screen to zero (default is column 2), thus producing a 40 column screen and thus making 10 column tabbing an excellent choice. Unfortunately, many TV sets have too much overscan to handle a true 40 column screen. Fortunately, Atari BASIC allows one to change the number of columns used in tabbing via a POKE 201, <tabwidth>. But the only factors of 38 are 19 and 2, meaning you can have 19 columns of 2 characters each or 2 columns of 19 characters each. Not much improvement so far.

Consider, though, the table of factors shown in Figure 1. As an example, if we have a screen 36 characters wide, we can have 2,3,4,6,9,12, or 18 columns. And to get a screen 36 characters wide is easy: just POKE 83,37 (presuming that location 82 still contains a 2). So look at the list of factors, choose a screen width of N, and you can use a tab width equal to any factor. NOTE: a tabwidth of two will not print numerics in only two columns.

Finally, consider the flexibility available by judiciously choosing your tabwidth setting:

```
20 POKE 201,4 : PRINT 1,2,
30 POKE 201,7 : PRINT 3,
40 POKE 201,10 : PRINT 4,5
```

Printing various values in a loop with this method can actually produce some quite readable columnar listings.

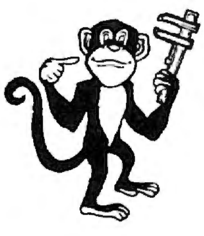
N	Factors of N
40	2,4,5,8,10,20
39	3,13
38	2,19
37	none
36	2,3,4,6,9,12,18
35	5,7
34	2,17
33	3,11
32	2,4,8,16

Figure 1.

```
0000        1000        .TITLE "DEMONSTRATION FOR DECEMBER COMPUTE"


DEMONSTRATION FOR DECEMBER COMPUTE
SYSTEM EQUATES

0000        1010        .PAGE "SYSTEM EQUATES"
            1020 ;
0342        1030 ICCOM  =    $342      ; 'COMMAND', IN IOCB
0344        1040 ICBADR =    $344      ; 'BUFFER ADDRESS'
0348        1050 ICBLEN =    $348      ; 'BUFFER LENGTH'
034A        1060 ICAUX1 =    $34A      ; 'AUX BYTE 1' (OPEN MODE)
            1070 ;
0003        1080 COPN   =    3         ; 'OPEN' COMMAND VALUE
0005        1090 CGTXTR =    5         ; 'GET TEXT RECORD'
0009        1100 CPTXTR =    9         ; 'PUT TEXT RECORD'
000C        1110 CCLOSE =    12        ; 'CLOSE'
            1120 ;
0006        1130 OPDIR  =    6         ; 'OPEN DIRECTORY' SUB-COMMAND
            1140 ;
E456        1150 CIO    =    $E456     ; WHERE TO CALL ATARI OS
            1160 ;
            1170 ; NOTE: OS/A+ users may omit lines 1010 thru 1160
```

```
             1180 ; if they use .INCLUDE #D:SYSEQU.ASM
             1190 ;
0000         1200 FILE0   =    $00        ; IOCB NUMBER * 16
0010         1210 FILE1   =    $10        ; IOCB NUMBER * 16
00FF         1220 LOW     =    $FF        ; MASK FOR LSB OF ADDR
0100         1230 HIGH    =    $100       ; DIVISOR FOR MSB


DEMONSTRATION FOR DECEMBER COMPUTE
BEGIN ACTUAL PROGRAM

0000         1240         .PAGE "BEGIN ACTUAL PROGRAM"
             1250 ;
             1260 ; HOUSEKEEPING:
             1270 ;
0000         1280         *=   $640       ; PUT ALL THIS IN SAFE PLACE
0640         1290         .OPT OBJ        ; WE DO WANT OBJECT CODE
             1300 ;
             1310 ; This program will list the
             1320 ; directory of disk D1: to the
             1330 ; E: device.
             1340 ;
             1350 ; Throughout, reference is made
             1360 ; to the BASIC demo program
             1370 ; which performs the same
             1380 ; functions.
             1390 ;
             1400 DIR
             1410 ; !!!! CAUTION !!!!
             1420 ; If this routine is to be used
             1430 ; from BASIC, the form MUST be
             1440 ; xxx=USR(addr)  as this routine
             1450 ; makes no check on number of
             1460 ; parameter bytes !!!
             1470 ;
0640 68      1480         PLA             ; PULL OFF # OF BYTES
0641 4C7206  1490         JMP  START
             1500 ;
             1510 ; We jump around the buffer.
             1520 ; Normally, the buffer would
             1530 ; be at the end; but we simulate
             1540 ; the BASIC program as closely
             1550 ; as possible
             1560 ;
             1570 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;
             1580 ;
             1590 ; 100 DIM BUFFER$(40)
             1600 ;
0028         1610 BUFLEN =   40
0644         1620 BUFFER *=  *+BUFLEN    ; RESERVER 40 BYTES OF SPACE
             1630 ;
             1640 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;
             1650 ;
             1660 ; 200 OPEN #1,6,0,"D:*.*"
             1670 ;
066C 44      1680 NAME   .BYTE "D:*.*",0
066D 3A
066E 2A
066F 2E
0670 2A
0671 00
             1690 ;  just a place to put filename
```

7 ATARI PRODUCTS



THE MONKEY WRENCH

The Monkey Wrench is a machine language ROM cartridge which extends the operating capability of the ATARI 800 computer. The Monkey Wrench provides 9 new BASIC commands. They are:

Auto Line Numbering	— Provides new line numbers when entering BASIC program lines.
Delete Line Numbers	— Removes a range of BASIC line numbers.
Renumber	— Renumbers BASIC's line numbers including internal references.
Cursor Exchange	— Allows usage of the cursor keys without holding down the CTRL key.
Change Margins	— Provides the capability to easily change the screen margins.
Memory Test	— Provides the capability to test RAM memory.
Hex Conversion	— Converts a hexidecimal number to a decimal number.
Decimal Conversion	— Converts a decimal number to a hexidecimal number.
Monitor	— Enter the machine language monitor. In addition to the BASIC commands, the Monkey Wrench also contains a machine language monitor with 15 commands used to interact with the powerful features of the 6502 microprocessor

Cartridge and Manual — $49.95

TYPING EXERCISE FOR ATARI

Typing Exercise is a great educational program for those who wish to improve their typing skills. Typing Exercise consits of two programs. TYPING 1 contains 13 typing drills, 9 drills progress thru alphabet and 4 thru numerics. TYPING 2 is a timed typing test. Time and words per minute are calculated for you. . . . . 810 Diskette — $12.95

EPROM CARTRIDGE

The EPROM cartridge is a specially designed printed circuit board which will allow the user to install his or her own EPROM software. Uses 2716, 2532, 2732, type EPROMs

Cartridge and Manual — $19.95

OTHER NICE STUFF

Atari 400 16K Memory	$345.00
Atari 800 16K Memory	$779.00
Commodore UIC-20 Computer	$265.00
Syncom or Memorex 5¼" disks	10 for $30.00
Mini-Flex disk File Case (holds 60 5¼" disks)	$24.95
Ribbon Cartridge For Starwriter, Diablo, etc.	Mylar —$5.00, Cloth — $6.00
Starwriter 25 cps printer with tractors Parallel Interface	$1645.00

MAE (Macro Assembler Editor)

MAE contains the most powerful 6502 assembler and test editor currently on the market. If you are looking for a professional development tool that can greatly increase the productivity of your programming staff, then MAE may be the answer. The following are just some of MAE's features — Write for detailed spec sheet:

- MAE was written entirely in machine language — not in Basic like some assemblers we know of. Thus you get very fast and accurate assemblies.
- Contains a machine language monitor with numerous commands for debugging machine code.
- 38 error codes, 27 commands, 26 pseudo ops, and 5 conditional assembly operators.
- Contains a word processor, example files, and learning aid.
- Requires at least 32K of memory.
- All commands oriented for disk operation with ATARI 810 disk drive.
- Macro, Conditional Assembly, and Interactive Asembly capability.
- Sorted Symbol Table.
- Optionally creates executable object code in memory or relocatable object code on disk.
- 50 page manual. 810 Diskette and Manual — $169.95

(requires license agreement)

MACRO ASSEMBLER AND TEXT EDITOR (ASSM/TED)

ASSM/TED is a high powered Macro assembler and text editor for use with ATARI 800 computers with at least 40K of memory.

- Written entirely in Machine Language — Not in Basic like some we know of. Thus you get very fast and accurate assemblies.
- 36 Error Codes, 26 Commands, 22 Pseudo ops.
- Macro and Conditional Assembly Capability.
- Input/Output of source files to cassette deck.
- Multiple source files on cassette may be assembled.
- Built-in machine language monitor. Cassette and Manual — $49.95

810 Diskette and Manual — $53.95

MACHINE LANGUAGE MONITOR FOR ATARI

The Machine Language Monitor for ATARI provides 21 commands which allows the user the ability to interact with the 6502 microprocessor. It is compatible with ATARI BASIC and (once loaded) is ready for your use at anytime. The monitor comes on cassette or on diskette for the ATARI 810 disk.

Cassette version — $24.95 Diskette version — $29.95

MEMORY TEST FOR ATARI

When you purchase a new ATARI or add on new RAM modules, you need to be sure that the memory is working properly. Remember, you only have a short guarantee on your memory. Memory Test performs the most extensive memory check available. . . . . Cassette and Manual — $6.95

EASTERN HOUSE SOFTWARE
3239 Linda Drive
Winston-Salem, N.C. 27106 U.S.A.

Call Orders: (919) 924-2889

We Pay Postage on Prepaid USA/Canada Orders. Add $7.50 to Cover Shipment Elsewhere

---

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

ATARI™ GRAPHICS MADE EASY? YES?

FINALLY — SOMEONE HAS WRITTEN A SERIES OF PROGRAMS THAT DO ALL THE DIFFICULT CALCULATIONS FOR YOU WHILE DEMONSTRATING THE SPECIAL "TRICKS" THAT THE ATARI IS CAPABLE OF . . . .

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 calculations!

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

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

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

#5: PLAYER MISSILE GRAPHICS — This complex subject will be 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. Ready Nov 1981. Requires 32 k disk or tape and costs 29.95 (extensive manual included)

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!

MASTER MEMORY MAP(tm) — 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**

THE GRAPHICS MACHINE!! — Turn your computer into an incredible graphics tool with advanced comands 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 need 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 retrival. The amount of text held depends on memory. Requires disk and 32k for **$14.95**

BOB'S BUSINESS — 14 small business type programs for home of office, all chosen from a nice menu Supports printed output. 169 sectors of programs require 32k, tape or 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 or 24k disk and cost **$14.95** each.

* * * * * * * * * * * * * * *

See your local Dealer, or send check to:

SANTA CRUZ EDUCATIONAL SOFTWARE
5425 Jigger Dr., Soquel, CA 95073
C.O.D. orders call (408) 476-4901
***Include $2.00 postage ($0.75 for Memory Map)**
***In Calif. include 6% tax**

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

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

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

```
DEMONSTRATION FOR DECEMBER COMPUTE
BEGIN ACTUAL PROGRAM

                1700 ;
                1710 START
                1720 ;  begin actual program
                1730 ;
0672 A210       1740        LDX  #FILE1
0674 A903       1750        LDA  #COPN       ; THE OPEN COMMAND
0676 9D4203     1760        STA  ICCOM,X     ; IS SET UP
0679 A906       1770        LDA  #OPDIR      ; MODE 6, DIR OPEN
067B 9D4A03     1780        STA  ICAUX1,X    ; THUS THE MODE
067E A96C       1790        LDA  #NAME&LOW
0680 9D4403     1800        STA  ICBADR,X    ; LSB OF ADDR
0683 A906       1810        LDA  #NAME/HIGH  ; AND MSB OF ADDR
0685 9D4503     1820        STA  ICBADR+1,X  ; ...OF FLNM
0688 2056E4     1830        JSR  CIO         ; CALL ATARI OS
068B 98         1840        TYA              ; CHECK STATUS
068C 3035       1850        BMI  LINE700     ; HUH??
                1860 ;
                1870 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                1880 ;
                1890 ; 300 TRAP 700
                1900 ;  SEE THE 'BMI' JUST ABOVE
                1910 ;
                1920 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                1930 ;
                1940 ; 400 INPUT #1,BUFFER$
                1950 ;
                1960 LINE400
068E A210       1970        LDX  #FILE1
0690 A905       1980        LDA  #CGTXTR
0692 9D4203     1990        STA  ICCOM,X     ; 'INPUT' A LINE
0695 A944       2000        LDA  #BUFFER&LOW
0697 9D4403     2010        STA  ICBADR,X    ; LSB OF ADDR
069A 8D4403     2020        STA  ICBADR      ; OF WHERE LINE GOES
069D A906       2030        LDA  #BUFFER/HIGH
069F 9D4503     2040        STA  ICBADR+1,X  ; AND MSB
06A2 8D4503     2050        STA  ICBADR+1    ; (WE ALSO SET UP ADDR FOR FILE #0)
06A5 A928       2060        LDA  #BUFLEN
06A7 9D4803     2070        STA  ICBLEN,X    ; BUFFER LEN
06AA 8D4803     2080        STA  ICBLEN      ; IS MAX WE USE
06AD 2056E4     2090        JSR  CIO         ; AND GO GET A LINE
06B0 98         2100        TYA
06B1 3010       2110        BMI  LINE700     ; "TRAP 700"
                2120 ;
                2130 ;;;;;;;;;;;;;;;;;;;;;;;;;;;
                2140 ;
                2150 ; 500 PRINT BUFFER$
                2160 ;
                2170 ; note that PRINT automatically
                2180 ; uses file #0, so we will do
                2190 ; so also !!
                2200 ;
                2210 ; also note that we saved a few


DEMONSTRATION FOR DECEMBER COMPUTE
BEGIN ACTUAL PROGRAM

                2220 ; bytes by setting up the buffer
                2230 ; address and length in 'LINE400'
                2240 ;
```

```
06B3 A909   2250         LDA  #CPTXTR
06B5 8D4203 2260         STA  ICCOM      ; PUT A LINE IS CMD
06B8 A200   2270         LDX  #FILE0     ; THE CONSOLE IS #0
06BA 2056E4 2280         JSR  CIO        ; TO THE I/O
06BD 98     2290         TYA
06BE 3003   2300         BMI  LINE700    ; OOPS?? HOW ???
            2310 ;
            2320 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
            2330 ;
            2340 ; 600 GOTO 400
            2350 ;
06C0 4C8E06 2360         JMP  LINE400    ; SELF EXPLANATORY
            2370 ;
            2380 ;
            2390 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
            2400 ;
            2410 ; 700 CLOSE #1
            2420 ;
            2430 LINE700
06C3 A210   2440         LDX  #FILE1
06C5 A90C   2450         LDA  #CCLOSE
06C7 9D4203 2460         STA  ICCOM,X    ; COMMAND IS 'CLOSE'
06CA 2056E4 2470         JSR  CIO        ; GO CLOSE THE FILE
06CD 60     2480         RTS             ; END OF ROUTINE
            2490 ;
            2500 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
            2510 ;
06CE        2520         .END
```

Discovering Atari's "Hidden" Graphics

Gregory L. Kopp
Indianapolis, IN

If you were a stumbling, beginning BASIC programmer like I was, you probably tried to enter a few "improper" graphics commands which resulted in curious and unexpected displays on your television screen. Before I understood the function and proper use of POKE 756 (which allows one to display Atari lower case letters and special graphics characters in text modes 1 and 2, I stubbornly tried to put CONTROL characters on-screen *without* the requisite POKE, which produced only seemingly random keyboard characters and frustration instead.

Much later, the thought nevertheless occurred to me that I might have actually accidently discovered some "hidden" (or at least undocumented) graphics capability of my Atari. In the experimental binge to which owners of microcomputers are sometimes given, I used the PRINT #6; command to enter each keyboard character while pressing CTRL at the same time. Discovery! Although the Atari special graphics characters appeared in the PRINT #6; statement, the actual screen display consisted of *keyboard* characters, but *not* the characters for the keys I entered.

Dutifully noting the results (first chart below), I pondered the apparent micro-fluke, these "hidden" characters, then asked myself the inevitable scientific question: "So what?" Two uses came fairly quickly to mind — the first purely cosmetic, the second functional.

```
10 GR. 2+16
20 X=0
30 FOR L=1 TO 50
40 RC=INT(15*RND(0)):RS=(255*RND(0))
50 SETCOLOR 0,RC,6
60 SOUND 0,RS,10,4
70 POSITION 5,4
80 PRINT #6; "1⌐▯⊟" (use CTRL=Q)
90 FOR W=1 TO 25:NEXT W
100 X=X+1:IF X=4 THEN X=0
110 NEXT L
120 SOUND 0,0,0,0
130 GR. 2+16
140 POSITION 5,4
150 PRINT #6; "1⌐▯⊟"
160 FOR W=1 TO 500:NEXT W
170 POSITION 2,7
180 PRINT #6; "HIDDEN GRAPHICS!"
190 FOR W=1 TO 1000:NEXT W
```

If one could change these hidden characters from "default green" to other colors, one could eliminate the irksome problem encountered in modes 1 and 2 of having punctuation and numbers displayed in different colors than the text lettering. The INVERSE key! Sure enough, PRINTing the graphics characters in inverse changed my hidden green characters to red. Now I could choose from normal character (yellow), inverse normal (blue), CTRL character (green) and inverse CTRL (red). Experimenting further, I discovered one could achieve *any* Atari color by use of a SETCOLOR 0 to 3 or POKE 709 to 711 command to change each respective character. No more would I have to sheepishly explain to those not-of-the-computer-persuasion why my apostrophe or my "1" was blue while my text was red!

... one may use "hidden graphics" to redefine the number set ...

So much for cosmetics. If you are not bothered by the inconsistent color text problem, then use the last two paragraphs as speed-reading exercises. However, if you have purchased software such as Iridis 2 or Datasoft's Character Generator, you may already have thought of the second application. Instead of redefining one's lower case (and thereby "losing" it) to achieve new characters, one may use "hidden graphics" to redefine the number set, selected punctuation marks, or arithmetic signs. While this could be done normally, using "hidden graphics" allows one to display numbers, punctuation, or signs in *four* colors instead of only two! (If you have not run the above program yet, try it. Then try to produce four different color 1's the conventional way.)

A Second Approach

Now enter and run the following program:

```
10 X=0: Y=0: Z=65
20 GR. 2+16
30 FOR AZ=0 TO 25
40 SOUND 0,255-AZ*10, AZ+8,8
50 COLOR Z
60 PLOT X,Y:IF X=18 THEN X=0: Y=Y+1
```

Announcing

ATARI

software™

from the authors of An Invitation to Programming

exciting games and educational programs for kids, teenagers and adults featuring sound and color graphics.

available on guaranteed-to-load cassettes at fine computer dealers in your area or, write us directly for descriptive materials

Program Design, Inc.
Department CA
11 Idar Court
Greenwich, CT 06830

203-661-8799

SOFTWARE for the ATARI 800* and ATARI 400* from QUALITY SOFTWARE

STARBASE HYPERION™
By Don Ursem

Become absorbed in this intriguing, original space simulation of war in the far future. Use strategy to defend a front line Star Fortress against invasion forces of an alien empire. You create, deploy, and command a fleet of various classes of space ships, while managing limited resources including power generators, shields and probes. Real time responses are sometimes required to take advantage of special tactical opportunities. Use of color, sound, and special graphics add to the enjoyment of this program. At least 24K of RAM is required.

On Cassette — $19.95 On Diskette — $22.95

NAME THAT SONG
By Jerry White

Here is great entertainment for everyone! Two players listen while the Atari starts playing a tune. As soon as a player thinks he knows the name of the song, he presses his assigned key or joystick button. There are two ways to play. The first way requires you to type in the name of the song. Optionally, you can play multiple choice, where the computer asks you to select the title from four possibilities. The standard version requires 24K of RAM (32K on diskette) and has over 150 songs on it. You also get a 16K version that has more than 85 songs. The instructions explain how you can add songs to the program, if you wish. Written in BASIC.

On Cassette — $14.95 On Diskette — $17.95

QS FORTH
By James Albanese

Want to go beyond BASIC? The remarkably efficient FORTH programming language may be just for you. We have taken the popular fig-FORTH model from the FORTH Interest Group and expanded it for use with the Atari Personal Computer. Best of all we have written substantial documentation, packaged in a three ring binder, that includes a tutorial introduction to FORTH and numerous examples. QS FORTH is a disk based system that requires at least 24K of RAM and at least one disk drive. Five modules that may be loaded separately from disk are the fig-FORTH kernel, extensions to standard fig-FORTH, an on-screen editor, an I/O module that accesses Atari's operating system, and a FORTH assembler.

Diskette and Manual — $79.95 Manual Only — $39.95

FOR OUR COMPLETE LINE OF ATARI SOFTWARE
PLEASE WRITE FOR OUR CATALOG

QS QUALITY SOFTWARE
6660 Reseda Blvd., Suite 105, Reseda, CA 91335
(213) 344-6599

ASK FOR QUALITY SOFTWARE products at your favorite computer store. If necessary you may order directly from us. MasterCard and Visa cardholders may place orders by calling us at (213) 344-6599. Or mail your check or bankcard number to the address above. California residents add 6% sales tax. *Shipping Charges:* Within North America orders must include $1.50 for shipping and handling. Outside North America the charge for airmail shipping and handling is $5.00. Pay in U.S. currency.

*Indicates trademarks of Atari

```
70 X=X+1:Z=Z+1
80 FOR W=1 TO 50:NEXT W
90 NEXT AZ
100 IF Z=91 THEN Z=193:X=0:Y=2:GOTO 30
110 IF Z=219 THEN Z=225:X=0:Y=4:GOTO 30
120 IF Z=219 THEN Z=225:X=0:Y=6:GOTO 30
130 FOR W=1 TO 500:NEXT W
140 POSITION 2,9:PRINT #6; "COLOR
    STATEMENT"
150 PRINT #6; " GRAPHICS!"
160 FOR W=1 TO 1000:NEXT W
```

If you are trying to figure out how we got all those alphabet characters using PLOT and COLOR statements, read on.

As any intermediate programmer can tell you, one cannot plot points in modes 1 and 2. you get absolutely nothing displayed if you try it. Of course the stumbling beginner might think the reason one gets nothing is that one did not enter a COLOR statement. Sandwiching COLOR 1 between the lines and trying again, he discovers that he has plotted an "!" instead of a point. "Pixel-head!" he chides himself. "You can't use PLOT in modes 1 and 2!" He notes this in his reference manual and ranks himself a step closer to intermediate programmer, missing the opportunity to discover more hidden graphics.

The Atari will plot a *character* in modes 1 and 2 at whatever position the programmer commands. The nature and color of that character are determined by a single COLOR statement. Using the COLOR Statement Graphics Chart, you can display any Atari keyboard character (POKE 756 for lower case) by using the associated COLOR statement, then plotting X,Y coordinates to place it at the desired position on the screen.

Once again, SETCOLOR 0 to 3 or POKE 709 to 711 can be used to color each individual character, *including* lower case characters which are normally limited to only two colors. (Note: these SETCOLOR's and POKE's work only when using GR. 1 or 2 *+ 16*.) Again, redefined characters may be used and this time manipulated arithmetically. Game-writers, rejoice!

While the PRINT #6; approach displays numbers, punctuation and arithmetic signs, the COLOR/PLOT technique allows access to upper and lower case letters as well. Preference for one method over the other will vary from user to user and application to application, as you will see once you have tried them a few times.

Table A. "HIDDEN GRAPHICS" CHART

To Get Character	Press CTRL + Key	Character	(Default)	SE.	POKE
0	P	normal	yellow	0	708
1	Q	"hidden"	green	1	709
2	R	inverse,			
3	S	normal	blue	2	710
4	T	inverse,			
5	U	"hidden"	red	3	711
56	V				
7	W				
8	X				
9	Y				
:	Z				
!	A				
"	B				
#	C				
$	D				
%	E				
&	F				
'	G				
(	H				
)	I				
*	J				
+	K				
,	L				
–	M				
.	N				
/	O				
[	;				
@	.				

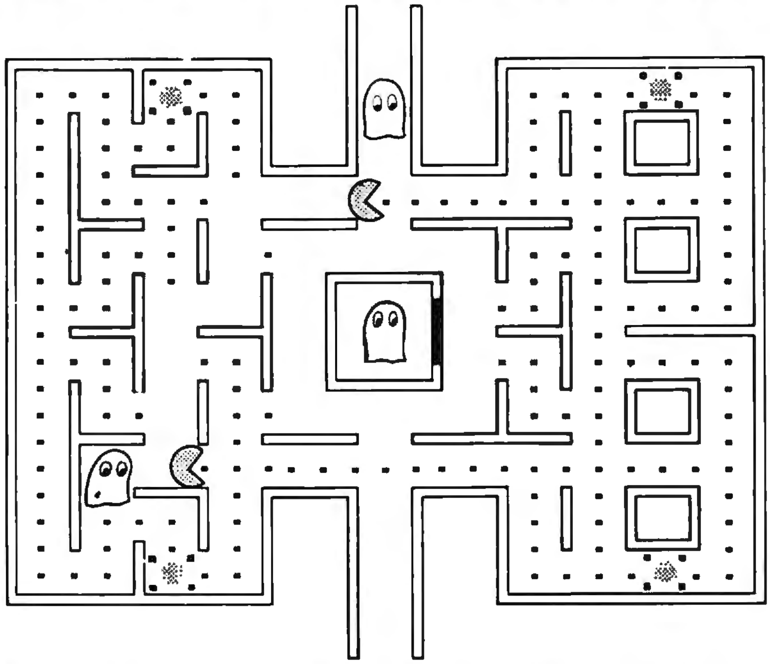
Other Color*		Press Keys
^	green	ESC then DELETE · BACK S
	red	ESC then CTRL + DELETE · BACK S
<	green	ESC then CTRL + |–
	red	ESC then SHIFT + DELETE · BACK S
>	green	ESC then CTRL + —
	red	ESC then CLR · SET · TAB
=	green	ESC then CTRL + |≐
	red	ESC then SHIFT + INSERT ·>
?	green	ESC then CTRL + —*
	red	ESC then SHIFT + CLR · SET · TAB
–	green	ESC then CLR · SET · TAB
	red	ESC then CTRL + INSERT ·>
;	green	ESC then ESC
]	red	ESC then CTRL + ".2

* greens manipulated by SE.1 and POKE 709
reds manipulated by SE.3 and POKE 711

Table B. COLOR STATEMENT GRAPHICS CHART

		Use Color Number			
To Get	SE.	0	1	2	3
Character	POKE	708	709	710	711
		green	yellow	red	blue
!		1	33	129	161
"·		2	34	130	162
#		3	35	131	163
!		4	36	132	164
%		5	37	133	165
&		6	38	134	166
'		7	39	135	167
(		8	40	136	168
)		9	41	137	169
*		10	42	138	170
+		11	43	139	171
,		12	44	140	172
–		13	45	141	173
.		14	46	142	174
/		15	47	143	175
0		16	48	144	176

GHOST HUNTER™



THE ULTIMATE GAME OF HIDE AND SEEK.

In the beginning... there was Star Raiders.™

Then... came Missile Command.™

Now... Arcade PLUS introduces Ghost Hunter,™ the first in a new generation of software for the Atari® 400/800™ personal computers.

Your mission in Ghost Hunter™ is simple—rid the mansion on Huckleberry Hill of ghosts... before they get you!

Ghost Hunter™ begins where most computer games end!

- 51 game variations!
- One or two player game... you can play alone or head-to-head against another player!
- Choose from 16 different "floor plans" or let the computer randomly choose for you!
- Special features allow you to "Build-A-Hunter" and "Hide-Instantly"!
- Fast paced, 10 color action with 4-channel music and simulated sound effects!

Ghost Hunter™ is available on cassette or disk for Atari® 400/800™ computers with 16K minimum memory from your local Atari computer dealer. Or send $29.95 (cassette)/$34.95 (disk) + $2.50 postage and handling (CA residents please add 6% sales tax).

arcade PLUS

5276 Hollister Avenue Suite 208 Santa Barbara, CA 93111 (805) 683-2305

1	17	49	145	177
2	18	50	146	178
3	19	51	147	179
4	20	52	148	180
5	21	53	149	181
6	22	54	150	182
7	23	55	151	183
8	24	56	152	184
9	25	57	153	185
:	26	58	154	186
;	27	59	*	187
<	28	60	*	188
=	29	61	157	189
>	30	62	158	190
?	31	63	159	191
@	96	224	*	192
[	91	123	219	251
]	93	*	221	253
\	92	124	220	252
^	94	126	222	254
–	95	127	223	255
A	97	65	225	193
B	98	66	226	194
C	99	67	227	195
D	100	68	228	196
E	101	69	229	197

Table C. COLOR STATEMENT GRAPHICS CHART (Cont.)

Character	Color Number			
	green	yellow	red	blue
F	102	70	230	198
G	103	71	231	199
H	104	72	232	200
I	105	73	233	201
J	106	74	234	202
K	107	75	235	203
L	108	76	236	204
M	109	77	237	205
N	110	78	238	206
O	111	79	239	207
P	112	80	240	208
Q	113	81	241	209
R	114	82	242	210
S	115	83	243	211
T	116	84	244	212
U	117	85	245	213
V	118	86	246	214
W	119	87	247	215
X	120	88	248	216
Y	121	89	249	217
Z	122	90	250	218

*Writing color statements that would logically appear in these positions displays nothing on the screen.

Atari Program Listings

COMPUTE! is starting a new, standardized Atari program listing format. All the editing and cursor-control characters are spelled out (e.g., CLEAR for clear screen) and surrounded by brackets.

Other characters, such as CTRL-T, the "ball" character, will be listed as the "normal" character within brackets: {T}. A series of identical control characters will be indicated by a number within the brackets, e.g. 5 DOWN for five cursor downs and 12 R for twelve CTRL-R's. Two control characters, { =} and {–} should be shifted. Any reverse-field text will be enclosed in vertical lines, | like this |. (Press the Atari logo key (Ⓐ) for each vertical line.) We expect that this convention will permit easy, unambiguous program typing.

PREMIUM SOFTWARE FOR YOUR ATARI

DEDICATED TO ATARI®

ARCADE GAMES 24K disk; 16K cassette; Joystick required

Add these HIGH RESOLUTION, REAL-TIME, ANIMATED games to your software arsenal. Get FAST ACTION and FULL SOUND GRAPHICS that take advantage of the unique features of your ATARI. Enjoy challenge that requires strategy and skill.

SPACE CHASE Fly against intelligent invader clones. Arm Yourself with Nuclear Defense Charges and play with or without Defense Shields. Enjoy this action-packed multicolor space odyssey. Displays top score, number of planets saved and number of galaxies conquered. $14.95 cassette; $19.95 disk

TIMEBOMB Meet the challenge of this fast moving animated race against time, enemy aircraft and enemy bombs as you attempt to disarm timebombs set to explode ammunition depots. Avoid aircraft of varying sizes and speeds — and their bombs. Choose one of ten Day or Night Missions. Use from one to four Joysticks. Any number can play; top players listed on scoreboard. $14.95 cassette; $19.95 disk

DATA MANAGEMENT

FILE-IT With this **startup database system** you can file and manage personal information and data. Use this database system to create, sort, store, and manipulate information such as appointment calendars, address or telephone data, credit or charge card records, stock investments, medical or prescription information, hobby, coupon or other types of collection informaton...and more. With a printer you get 1 or 2 across mailing labels, disk jacket inventory covers and neatly written copy of all your data files. Comes with well documented instruction manual explaining basics of computer filing. Fast and easy to use. Holds 100+ records in 24K and over 300 in 40K. Requires minimum of 24K and one disk drive. Pinter optional. $34.95 (Disk Only)

FILE-IT 2 An **expanded database system** which extends FILE-IT, provides the following additional capabilities. User controlled data selection for creating subfiles from main data files. Random access file updating for label and financial data files. Financial entry and report generator programs provide data selection by code(s) and/or date(s). Monthly bar graph program generates visual pictures of selected data on screen/printer. Requires 1 disk drive, minimum of 24K RAM, and an 80 column printer. Supports single or multiple disk drives. Includes detailed documentation, users manual and utility programs. $49.95

EDUCATIONAL/ENTERTAINMENT

MY FIRST ALPHABET Will give your youngster an unparalleled learning experience. Complete with melodies and thirty-six professional drawings, children see pictures and hear tunes with letters and numbers of their own choosing or the ones you choose to show. Package includes a GRAPHICS EDITOR for creating, editing and VIEWING your own custom drawings in real time. Use drawings with any basic program or as part of MY FIRST ALPHABET. Instruction manual included. Minimum of 24K and disk required. $29.95

WORDGAMES This package is jam-packed with hours of fun and challenge. Wordgames contains GUESSIT, WORDJUMBLE and POSSIBLE. GUESSIT, a deductive alphabetic reasoning game for 1 or 2 players can also be used for teaching or learning dictionary look-up skills. Comes with 60 word vocabulary. WORDJUMBLE is a multiple word descrambling puzzle with play-on-word hints and mystery answers. Comes with 20 puzzles. Instructions show how to substitute your own words and clues. Use POSSIBLE as a word game tool to assist creating or playing word or letter scrambling games by showing all combinations of letters you supply. If you like word games you will love this package. 16K cassette $14.95; 24Kdisk $19.95

UTILITIES

PROGRAMMING AIDS PACKAGE I Is four utility programs to help increase programming efficiency and learn more about your computer. RENUMBER handles references and even variables. Generates Diagnostic Tables for programming error detection. PROGRAM DECODER, DECIMAL TO BCD and BCD TO DECIMAL programs give you a practical way of studying internal program representation and ATARI number-conversion procedures. Comes with comprehensive users manual. 16K cassette $14.95; 24K disk $19.95

SWIFTY UTILITIES Make programming time more efficient; increase programming productivity. Includes all of PROGRAMMING AIDS I plus has REM Remover, Variable Lister, Disklist, DOS Caller (access DOS Utilities with program in core), MENU/SELECTOR (to run programs in either saved or listed format) and Custom Print (for preparing condensed and indented program listings on either EPSON MX-80's or ATARI 825 printers). (Listings skip page perforations and title and number pages.) Minimum of 24K disk system required. $29.95

DISKETTE INVENTORY SYSTEM Use this system to gain control of your expanding disk/program inventory. Quickly get locations of single or multiple copies of your programs and all your valuable files. An invaluable tool, this system is easy and convenient to use and to update. 24K disk system required. $24.95 Printer suggested.

GRAPHICS EDITOR (refer to MY FIRST ALPHABET.)

SWIFTY DATALINK High Quality Smart Terminal Communications program. Easy to use Multi-Option, Menu Driven. Full performance uploading/downloading. Printer dump, screen dump and disk search options. Use as remote terminal. Send/receive and store programs and data files. Save on connect time charges with commercial services.....$34.95
Requires: 24K RAM, 810 disk drive, 850 interface or equivalent, 830 or other 300 baud modem. (Printer Optional)

SEND CHECK OR MONEY ORDER TO:

SWIFTY SOFTWARE, Inc.
P.O. Box 641 Melville, New York 11747 (516) 549-8141
N.Y. Residents add 7¼ % sales tax
SEND FOR FREE CATALOGUE

Fantasy for your ATARI

Ali Baba and the forty thieves

By Stuart Smith

Guide your alter ego, Ali Baba, through the thieves' mountain den in an attempt to rescue the beautiful princess. Treasure, magic, and great danger await you! One or more human players can guide up to seventeen friendly characters through the many rooms, halls, and caves. Some characters wander around randomly, making each adventure a little different.

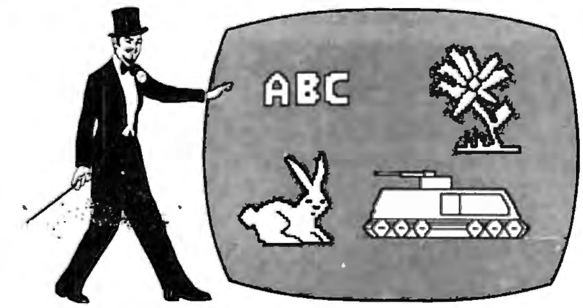
ALI BABA AND THE FORTY THIEVES is written in high resolution color graphics and includes music and sound effects. Adventures can be saved to disk and resumed at a later time. Requires 32K.

On Diskette Only — $32.95

Graphics for your ATARI

Character Magic

By Chris Hull



It's easy to create your own character sets and save them to diskette or cassette with CHARACTER MAGIC. But this is not just another character editor. CHARACTER MAGIC helps you use all the character types that the Atari is capable of, including descending characters (8 x 10 dots) and two types of five-color character graphics not supported by Atari's Operating System. Documentation includes examples of display lists that let you use these "secret" graphics modes. Requires 32K.

Cassette or Diskette - $29.95

FOR OUR COMPLETE LINE OF ATARI SOFTWARE PLEASE WRITE FOR OUR CATALOG

ASK FOR QUALITY SOFTWARE products at your favorite computer store. If necessary you may order directly from us. MasterCard and Visa cardholders may place orders by calling us at (213) 344-6599. Or mail your check or bankcard number to the address above. California residents add 6% sales tax. *Shipping Charges:* Within North America orders must include $1.50 for shipping and handling. Outside North America the charge for airmail shipping and handling is $5.00. Pay in U.S. currency.

QS QUALITY SOFTWARE

6660 Reseda Blvd., Suite 105, Reseda, Ca. 91335 (213) 344-6599

*Indicates trademarks of Atari

String Art

Craig Maiman
Spring Valley, NY

If you always wanted a program which generates beautiful mathematical patterns, now you have it. This program serves very well to impress friends (and yourself!) with the graphics capabilities of the Atari 400/800.

The program actually generates two lissajous figures that are TWISTed relative to each other, it then draws lines between them. The variables that determine the shape of the lissajous are FREQUENCY RATIO and PHASE. Since a lissajous pattern is generated by two signals perpendicular to each other, as on an oscilloscope, we can specify their frequency ratio to obtain many different figures. Changing the phase makes the pattern seem to rotate in 3-D space. An example would be a circle which has a frequency ratio of one-to-one and a phase of 0 degrees. If you now change the phase to 45 degrees it will look like a tilted ellipse. Another variable which can be controlled is DISPLACEMENT: this variable determines the vertical separation of the two lissajous patterns. It can be varied between 0 and 95.

Here are some numbers to generate nice patterns:

FREQUENCY RATIO: 1,1
PHASE: 40
TWIST: 135
DISPLACEMENT: 0
FREQUENCY RATIO: 2,1
PHASE: 0
TWIST: 60
DISPLACEMENT: 0
FREQUENCY RATIO: 1,2
PHASE: 45
TWIST: 120
DISPLACEMENT: 70

Hints

1. Good numbers for the FREQUENCY RATIO are various combinations of 1,2, and 3. Higher numbers tend to make messy pictures.

2. For PHASE, 0 and 90 are good numbers, but try numbers in between also.

3. The TWIST can be varied from –180 to 180. Try them all if you want.

4. DISPLACEMENT can vary between 0 and 95. When you get near 90 all the pictures start looking the same. For starters, I would recommend using 0 then try 20, 30, etc.

Now to reveal the secrets of the program:

Lines	Description
20-40	Screen initialization (Put String... in inverse video)
50-80	Prompts for input (Put these in inverse video)
90	Tests for illegal displacement
130-140	Initializes screen for GRAPHICS 8
160-170	generates first lissajous
180-190	generates second lissajous
200	plots and connects the 2 lissajous
210-220	tests for key touch

```
10 REM STRING ART ! BY CRAIG MAIMAN, JUL
Y, 1981
20 GRAPHICS 0:SETCOLOR 1,3,10:SETCOLOR 2
,3,0
30 SETCOLOR 4,6,0:H=95:POKE 752,1
40 ? "        | STRING ART PROGRAM |"
50 ? :? :? "|   INPUT FREQUENCY RATIO |"
;:INPUT A,B
60 ? :? :? "|   INPUT PHASE |";:INPUT PH

70 ? :? :? "|   INPUT TWIST |";:INPUT TW

80 ? :? :? "|   INPUT DISPLACEMENT |";:I
NPUT DI
90 IF DI>95 THEN 80
100 ? :? :? "|   WHEN PICTURE IS DONE HI
T |"
110 ? "|   ANY KEY TO CONTINUE      |"
120 FOR D=0 TO 680:NEXT D
130 DEG :GRAPHICS 24:SETCOLOR 1,3,10
140 SETCOLOR 2,3,0:SETCOLOR 4,6,0:COLOR
1
150 FOR ANG=0 TO 360 STEP 5
160 X1=H*SIN(A*ANG)+159
170 Y1=(H-DI)*COS(B*ANG+PH)+96-DI
180 X2=H*SIN(A*(ANG+TW))+159
190 Y2=(H-DI)*COS(B*(ANG+PH+TW))+96+DI
200 PLOT X1,Y1:DRAWTO X2,Y2:NEXT ANG
210 IF PEEK(764)<>255 THEN POKE 764,255:
GOTO 20
220 GOTO 210
```

©

WORDSEARCH by GO-TARI

ATARI* Games & Good Music!!

400/800 WITH 16 k & JOY STICKS

W	F	H	L
O	O	C	O
W	R	R	P
R	E	A	D
P	L	E	A
J	U	S	T
S	Y	O	U

Computer assisted 'BONGS' & 'THUDS'
Kids enter weekly spelling words!
Use names, places, or 'fun' words
Select the WORDSEARCH words
Key words change color & score
Hear latest release from AXBAR RECORDS
ONLY $15.95 FOR CASSETTE TAPE
(INQUIRE ABOUT OUR CARTRIDGES)

GO-TARI Enterprises
PO Box 33684
San Antonio, TX 78233-0684
(512) [illegible]

*ATARI 400/800—REG.T.M. of Atari, Inc.

ATARI

"QUALITY DESIGN WITH THE USER IN MIND"

DRUG BUST – Exciting new game that puts you in the midst of drug trafficking. Buy at rock bottom prices and resell for enormous profits. But, don't get caught holding! One or two players. Requires 16K.
Cassette **$19.95**
Diskette **$24.95**

DIRECTORY – Load this user friendly program and you can view diskette contents, select and run programs. Menu selection approach as found in many larger computers. Program is transferrable to any diskette. Requires 3K.
Diskette only **$18.95**

ROLL-EM – System generated dice roll. Excellent graphics for use in any game of chance requiring dice. Included as a bonus when both above programs are ordered. Requires 6K.
Cassette **$7.95**
Diskette **$11.95**

Catalog included in all above orders. For catalog only, send $1.00 to cover postage and handling to:

Esplanade Enterprises
2042-312 Peach Orchard Drive
Falls Church, VA 22043

Add 5% Virginia sales tax. Sorry, no C.O.D. orders please.

GALACTIC CHASE™

The aliens have swept undefeated across the galaxy. You are an enterprising star ship captain—the final defender of space.

As the aliens attack, you launch a deadly barrage of missiles. Flankers swoop down on your position. Maneuvering to avoid the counterattack, you disintegrate their ships with your magnetic repellers.

As your skill improves, the attackers increase their speed. And as a last resort, the aliens use their invisible ray to slow the speed of your missile launcher.

GALACTIC CHASE provides Atari owners with the most challenging one or two person game in the galaxy.

ATARI®

Atari 400/800 16k. Written in machine language. Requires joysticks.
Payment: Personal Checks—allow three weeks to clear.
American Express, Visa, & Master Charge—include all numbers on card. Please include phone number with all orders. 24.95 for cassette or 29.95 for disk plus 2.00 shipping. Michigan residents add 4%.

Check the dealer in your local galaxy. Dealer inquiries encouraged.

Galactic Chase © 1981 Stedek Software.

SPECTRUM COMPUTERS

Dept C.
26618 Southfield
Lathrup Village, MI. 48076
(313) 559-5252

Billiard Bounce

Kevin and Priscilla Laws
Carlisle, PA

This program was written initially to provide a graphic demonstration of two lessons on Billiard Ball Mathematics presented by Harold Jacobs in his delightful book entitled *Mathematics: A Human Endeavor* (Freeman, San Francisco, 1970). Once the program was entered, we discovered that we could spend hours watching wonderful patterns unfold before us. Floor designs, Navajo rugs, smooth and nubby fabrics can all be designed with a simple change of two inputs.

In the program, the path of a "billiard ball" is traced on a "table" with a horizontal length of 160 pixels and a vertical width of 96 pixels. The user inputs the horizontal and vertical distance the ball moves during each program step. (These inputs determine the angle at which the ball moves.) A background color and trace color are chosen at random during each run to prevent viewers from becoming tired of the color scheme.

When the program is run the Atari prints:

Angle Horizontal, Vertical?

The user then enters two numbers separated by a comma, such as 1,2 and presses the return key. Users will quickly discover that integers lead to fairly smooth patterns, large numbers to rapidly unfolding patterns, and decimal fractions to jagged lines. Some entries a novice user might like to try:

3.14159, 3.14159	**(leads eventually to a waffle iron)**
3.3, 7.7	**(looks like a woven curtain)**
6.2, 6.3	**(a folksy looking fabric)**
4.5, 6.3	**(an indian rug)**
3.4, 5.5	**(a greek design)**
2, 9	**(bedsprings!)**

If a particularly interesting pattern appears before the design is complete it can be studied by pressing the "CNTRL" key and the "1" key simultaneously. Hitting these two keys again will allow the design process to continue.

The use of the GRAPHICS 7 + 16 mode allows the program to fit easily in an 8K Atari.

```
10 PRINT "Angle Horizontal, Vertical":IN
PUT X1,Y1
20 GRAPHICS 7+16
30 COLR1=INT(RND(0)*15)+1:SETCOLOR 4,COL
R1,10
40 COLR2=INT(RND(0)*15)+1:SETCOLOR 0,COL
R2,5
50 IF COLR1=COLR2 THEN 40
60 X=X-X1:Y=Y-Y1
70 IF ABS(Y)>95 THEN Y=95
80 IF ABS(X)>159 THEN X=159:GOTO 100
90 GOTO 60
100 COLOR 1:PLOT ABS(X),ABS(Y)
110 DRAWTO ABS(X),ABS(Y)
120 X=X-X1:Y=Y-Y1
130 IF ABS(X)>159 THEN X=159
140 IF ABS(Y)>95 THEN Y=95
150 GOTO 110
```

Blinking Characters

Frank C. Jones
Silver Spring, MD

The inverse video (⁄⁄\ ) key on the Atari 400/800 computer allows messages to be displayed in inverse video for special emphasis or eye-catching effects. Another, sometimes even more dramatic, method of catching the viewer's eye is to have the message flash on and off, or blink. There is no simple command in Atari BASIC to produce this effect, but the key to producing it lies in the register, maintained by the operating system, called CHACT (Dec-755, Hex-2F3). If bit 1 in this register is set true, inverse video characters are displayed in inverse video; if it is set to zero they are displayed normally. However if bit 0 is set true, these characters are displayed as blank spaces (inverse video or normal blanks depending on bit 1).

Looking For A Faster Solution

With this information we can write a program that will produce blinking characters on the screen. (Program 1). The trouble with this approach is that our BASIC program is completely preoccupied with timing loops and toggling bit 0 of CHACT. If we try to incorporate this method in a program that does anything else, the timing problem gets very difficult, if not downright impossible. What we really want is a routine that will sit in the background and toggle bit 0 of CHACT on a regular basis without interfering with any BASIC program that might be running at the time. Fortunately the Atari has in it the resources we need to do just this.

The Atari operating system maintains five separate timers which are incremented or decremented during every vertical blank period (the period between successive TV picture frames during which the screen is dark). One of these, called CDTMV2 (Hex21A) is a two byte down counter that can be set to any value between 1 and 65535. Every 60th of a second, during vertical blank, the operating system reduces this number by one and, when it counts to zero, initiates a subroutine jump to the address that it finds in the two byte vector CDTMA2 and returns to the operating system, waiting for the next time the counter counts down to zero.

Program 2 achieves this result by poking a machine language program into memory starting at page 6 (Dec1536,Hex 0600) and transferring control to it via the USR function. Since the operation of this program is obvious, we will spend no time discussing it. Rather, we will turn our attention to the assembly language version of the program that does all of the work, Program 3.

Lines 20-30: Identifies the hex locations of the names times and registers.

Line 50: Starts the program assembly at location hex 0600 (decimal 1536).

Lines 60-130: Initialize jump vector and start timer.

Line 60: Pops one number off the stack. This is required by the USR function; the routine will work without this step, but you will get an Error-9 on return to BASIC.

Lines 70-100: Stores the address of the routine that begins on line 140 in the subroutine jump vector CDTMA2.

Line 110-120: Stores a number in the timer, CDTMV2, to get it going; the number itself is arbitrary.

Line 130: Return from initializing subroutine.

Line 140: Start of subroutine that does the blinking; load the register CHACT into the accumulator.

Line 150: AND the accumulator with the number one, turns off all bits except bit 0.

Line 160: EOR — exclusive OR the accumulator with the number one; reverses the state (toggles) of bit zero.

Line 170: Stores the result back into the register CHACT.

Line 180-190: Resets the timer CDTMV2 at 30 (½ sec.).

Line 200: Return from blink subroutine.

This program, in machine language, is contained in the string of numbers in the DATA statement, line 50, of the BASIC in Program 2. A few of the numbers are readily identified and can be changed by the user to obtain different effects. First of all, the last #30 in the list (the 29th number) is the number that is loaded into the countdown timer each cycle. It determines the delay time between each jump to the toggling routine and hence the blink frequency. Since the counter is decremented every 1/60 of a second, loading 30 in the timer causes the characters to be on for ½ second and blank for another ½ second. This

ALTERNATE REALITY SOFTWARE

Presents for the Atari

THE I CHING

THE ANCIENT CHINESE BOOK OF DIVINATION

- The complete text of the world's oldest book on disk
- 40K program
- 73 disk files (155,000 bytes)
- Occupying 1211 disk sectors
- High Resolution Graphics
- Animation
- Music and Sound Effects
- Instructional text material
- Calculates and Displays Hexagrams
- Displays Judgement, Image, Moving Lines for primary & secondary Hexagrams
- $44.95

order from:

Alternate Reality Software
2111 W. Arapahoe Drive
Littleton, Colorado 80120

Dealer inquiries invited
Atari is a T.M. of Atari, Inc.

VIEW FROM THE TOP
(TRS-80 16K Atari 24K)
A presidential simulation in which you as president must solve the whole array of problems now facing our country. $24.95

WORKERS COMPENSATION RETRO RATING
(TRS-80 8K Atari 8K)
This is an insurance utility to aid agents and brokers in the rating of retrospective rating programs. $19.95

MURDER-MURDER-MURDER
(TRS-80 16K Atari 24K)
In this simulation you as a well known detective must solve the mystery. Depending on how you proceed with your investigation, the solution may change. $21.95

SPACE EMPIRES
(TRS)-80 J6K Atari 16K)
With this fantasy you must colonize as many planets as you can in 20 years. You will face many problems and hazards but the rewards are great — your own empire. $15.95

The above programs are available on cassette only.

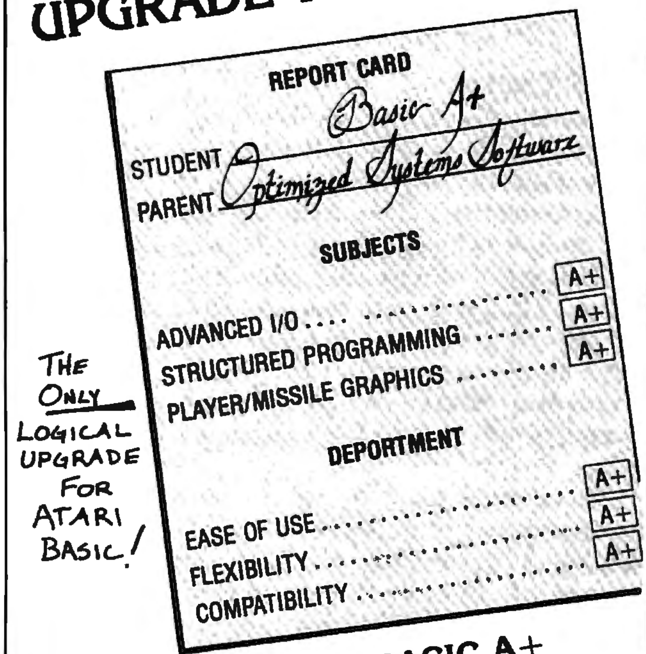
Send check, money order or see your local dealer.

C.E.L. PROGRAMS
4040 Winterpark Dr.
Liverpool, NY 13088

TRS-80 is a trademark of the Tandy Corporation
Atari is a trademark of Atari, Inc.

Atari is a reg. ™ of Atari, Inc.

UPGRADE YOUR ATARI®



GRADUATE TO BASIC A+ FROM THE AUTHORS OF ATARI BASIC!

BASIC A+

BASIC A+ will rate an A+ from any Atari user! Upward compatible with Atari Basic, it adds statements and features that enhance the Atari 800's real power, flexibility, and ease of use: Superior I/O features for business and other applications. Additional file manipulation commands. Significant help in program development and debug. Structured programming aids. And MORE! A partial list of the enhancements of BASIC A+ includes:

RPUT/RGET (record I/O) **BPUT/BGET** (binary I/O) **ERASE**
PRINT USING SET TAB INPUT"..." DIR PROTECT RENAME
TRACE WHILE...ENDWHILE IF...ELSE...ENDIF
SUPERB PLAYER/MISSILE GRAPHICS

BASIC A+ requires a disk and 32K bytes of RAM. Since no cartridge is used, BASIC A+ will take advantage of all the RAM (48K bytes) in a maximum Atari 800 system (recommended) $80

OS/A+

Completely compatible with Atari's DOS (version 2), but with an advanced, command-driven console processor. Simple. Flexible. Powerful. With an easy-to-use **BATCH** capability. OS/A+ **INCLUDES** all the following utilities (and more):

EASMD (Editor/ASseMbler/Debug) is our upgraded all-in-one assembly language development package for the 6502 microprocessor. The editor — with such features as FIND and REPLACE — can even edit BASIC A+ programs. The assembler can include multiple source files in a single assembly.

DUPDSK and FORMAT create master or slave disks. Make sector by sector copies of any OS/A+ disk.

COPY a simple, single file copy utility.

All of this power is included in our OS/A+ package $80

For the utmost in capability and flexibility, our combination system — **BASIC A+** and **OS/A+** — is available $150

SOFTWARE IN A CLASS BY ITSELF
AVAILABLE NOW THROUGH LOCAL DEALERS

Optimized Systems Software, Inc.
10379 Lansdale Ave., Cupertino, CA 95014,
(408) 446-3099

number may be changed by the user to be anything between 1 and 255 to make the blink frequency anything between 1/30 of a second to 8½ seconds.

See For Yourself

The two ones in the list, the 22nd and 24th numbers are the ones that are AND'ed and EOR'ed against the contents of CHACT. Changing the first one to 23 leaves inverse video on during the blanking. If both ones are changed to threes, inverse video is on when the characters are on, but the blanks are normal. Changing both ones to twos causes the characters to alternate between normal and inverse video. Finally putting fours in place of the ones produces an effect that you will just have to see for yourself; I still haven't figured out a use for this one.

Once the BASIC program has been entered and run, it may be erased by typing NEW, (RETURN), and entering a new program and the flashing will continue (the flashing will stop during I/O to or from a disk or cassette since stage two vertical blank processes are suspended during I/O operations). System Reset will stop the flashing and bring back inverse video; however, merely typing A = USR(1536) (RETURN) will reinstate the flashing characters without having to reload and run the BASIC program.

This program may be added on to almost any other BASIC program to produce the flashing characters and should add some new twists to your special effects.

Program 1.

```
10 CHACT=755
20 DELAY=200
30 PRINT "!HELLO!"
40 FOR I=1 TO DELAY:NEXT I
50 POKE CHACT,0
60 FOR I=1 TO DELAY:NEXT I
70 POKE CHACT,1
80 GOTO 40
90 END
```

Program 2.

```
10 FOR I=1536 TO 1536+32
20 READ B:POKE I,B:NEXT I
30 A=USR(1536)
40 END
50 DATA 104,169,17,141,40,2,169,6,141,41
,2,169,30,141,26,2,96,173,243,2,41,1,73,
1,141,243,2,169,30,141,26,2,96
```

Program 3.

```
                   ;CHARACTER BLINK ROUTINE
02F3               CHACT     =    $2F3
021A               CDTMV2    =    $21A
0228               CDTMA2    =    $228
0000                         *=   $0600
0600 68            INITIATE   PLA
0601 A911                     LDA #BLINK&$00
FF
0603 8D2802                   STA CDTMA2
0606 A906                     LDA #BLINK/256

0608 8D2902                   STA CDTMA2+1
060B A91E                     LDA #30
060D 8D1A02                   STA CDTMV2
0610 60                       RTS
0611 ADF302        BLINK      LDA CHACT
0614 2901                     AND #1
0616 4901                     EOR #1
0618 8DF302                   STA CHACT
061B A91E                     LDA #30
061D 8D1A02                   STA CDTMV2
0620 60                       RTS
```

Make Your Atari Keyboard A Little Friendlier

Ric Mears
San Francisco

If you've ever been typing on the Atari computer and hit the inverse video key instead of the shift key, cleared the screen inadvertently, or locked the keyboard into character graphics mode when you only meant to move the cursor right one space, then this article is for you.

Although I do use assembly language to tame the Atari down a bit, don't worry if you don't know a thing about programming in assembly because I've included a version of my code written in BASIC. And for those of you who are old pros at tickling the insides of your machines, you'll find a listing of the original source code with notes on how it works.

This code does two main things. First, it speeds up the initial delay you encounter while waiting for the auto-repeat to engage. Normally there is a pause of about 3/4 of a second after pressing a key before it starts to repeat. This seems a bit too long to me. It gets to be aggravating, especially when moving the cursor around the screen for editing. You can adjust this speed to suit yourself, or even shut the auto-repeat off altogether if you like.

The Right Pinkie's Burden

The second main thing this code does is to give you an audible signal when you press the inverse video key, the clearscreen key, caps/lower key, or the shift-delete key. The burden of finding all these keys plus a number of others is dumped completely on the right hand's little pinky, making it all too easy to make a mistake. For example, when typing a long mathematical formula, it has been quite frustrating to me when, instead of hitting the final right parenthesis, I'd accidentally press the clear-screen, leaving me to start all over. This happened to me so many times that I decided to do something about it. Now, instead of the screen going blank, I get the sound of five quick keyclicks letting me know my little pinky is off target. If I really want the screen cleared, I just hold the key down a bit longer than usual for the key to be activated. This short wait seems a small price to pay compared to the time that can be lost in correcting mistakes.

Whether you know assembly or not, if you have a disk drive and DOS 2, you can create an "AUTORUN" file so that, when you turn your computer on, the keyboard will behave itself automatically. If you're using DOS 1 or a cassette only system, you'll have to run the BASIC program each time you power up. Of course, those of you without DOS 2 running the assembler cartridge will have to load in the necessary object code after powering up.

It is possible to "tap the computer on the shoulder"...

Cassette Users

A special note to anyone using a cassette, with or without disk drives: If you want to use your cassette in addition to taming your keyboard down, you'll have to switch the two values at the beginning of the BASIC version of the program. I'll try to explain:

There are 256 bytes of memory (referred to as Page 6) available for use any way you see fit located at address 600 hex, or 1536 decimal. The computer's operating system doesn't use them, nor does any typical BASIC program. Still, many BASIC programs, with a need for high-speed assembly language subroutines, *do* use Page 6. In my own programming, I frequently make use of this area of memory and wanted to put the keyboard code someplace else which was safe from unfriendly programs. There are 131 bytes of memory similar to Page 6, located at address 3FD hex, 1021 decimal, unused for anything but conversations between the computer and the cassette. Since I rarely use my cassette, I appropriated this area of memory to hold the new keyboard code. Since this forfeits your ability to use the cassette, you may wish to forfeit use of Page 6 instead by switching the two previously mentioned values at the beginning of the program. The initialization code can pretty much go anywhere, since it's only used once when the program is run, but the keyboard code has to have a permanent place to reside. If neither Page 6 nor the cassette buffer will work for you, you'll have to find some other place for the code. But when you do, you'll only have to change the values at lines 190 and 200 to the new addresses where you want the code placed since it is position independent.

Setting Up

For those of you using a cassette only, or those of you with DOS 1, just run the BASIC program that follows. I suggest first saving a copy of the program before running it though, since, as with any assembly program, if you make a mistake, your computer may very well "lock-up," requiring you to turn it off and back on. Hopefully you will find this program useful and a timesaver. Once it has been run, it is no longer needed in memory and is transparent to normal operations of the computer.

This paragraph is for those of you using DOS 2. Run the BASIC program that follows and see how your keyboard acts afterwards. Then, call up the DOS menu and type "K" for the binary save option. If you already have an AUTORUN file on your disk, respond with "D1: AUTORUN,SYS/A, 3FD, 47F". If you don't already have an AUTORUN file on the disk leave the "/A" off the filename. This will save or append the cassette buffer area containing the new keyboard instructions. Again type "K", and now respond with "D1: AUTORUN-.SYS/A, 600, 60B, 601". Don't forget the "/A". This will append the initialization code along with the proper initialization address. That's it Now, whenever you turn the computer on with this AUTORUN file present, your keyboard will be ready to go automatically.

Changing The Auto-repeat Speed

As I mentioned before, the delay before the auto-repeat engages can be changed, and I've shortened it to suit my typing habits. Decreasing the second number in the data list at line 520, which is now 15, will decrease the length of time before the auto-repeat starts. I doubt you will want to decrease it, but increasing this number will slow the repeat down if you end up typing things like "LLISSTT." You can turn the auto-repeat off altogether by changing this number to a zero. If you should want to disable the new keyboard characteristics (to use the cassette for example) just press the System Reset key. It'll restore the keyboard to normal.

After careful study of the operating system source code, I regret to report that two desirable changes are not feasible. It does not look possible to change the speed of the auto-repeat once it has begun. Nor does it look feasible to shut off the keyclick with software. The keyclick got on my nerves when I first got my computer and, for what it's worth, I dicsovered that the bottom cover of the computer is easily removed (at the expense of voiding your warrenty) and the speaker can be unplugged. However, you lose use of the warning bell or buzzer, so I chose to solder a 200 ohm resister in series with the speaker to soften the volume of the keyclick without losing the bell.

How It Works

At the heart of the Atari computer is a 6502 microprocessor chip. It's always running some kind of program and always in the only language it understands, 6502 machine code. Whenever you use the BASIC cartridge, you're really running a machine language program called "BASIC," which waits for you to tell it to do something, such as to run a program you've typed into memory with its assistance. This program then "interprets" your instructions and sends the 6502 off to execute various machine code subroutines which collectively, and in the proper sequence, accomplish what you told the BASIC "interpreter" you wanted to have done. So, any time your computer is on, the 6502 is always executing machine code instructions.

Interruptions

It is possible to "tap the computer on the shoulder," asking it to stop what it's doing and take care of something else. This is called an *interrupt*. Regardless of whether the computer was busy drawing on the screen, trying to figure out a BASIC program, or whatever, it'll take note of what it was doing so that it can pick up again where it left off after it finishes taking care of the interruption.

There are all kinds of things that can cause an interrupt. One of them is pressing a key on the keyboard. Whenever a key is pressed, the 6502 gets "tapped on the shoulder," and it goes to a subroutine located in the operating system ROM. This subroutine does some checking around and finds out that a key was pressed. Then it looks at memory location 208 hex, which is called VKEYBD, short for Keyboard Vector. Now, *vector* is just a fancy word for *pointer* and normally, this memory location contains the starting address of another subroutine in ROM which takes care of saving information about the particular key that was pressed so that, at some time later on, yet another subroutine can do something with it (like print the letter on the screen). The initialization code at the end of the assembly language listing changes the value contained in the keyboard vector VKEYBD so that the 6502 will follow *our instructions* on what to do when a key is pressed. In fact, we'll tell the 6502 to figure out if the key pressed is one of those that our little pinky hits by mistake so often.

Most of the assembly code is fairly self-explanatory, but a few things are worth discussing. At line 500 of the assembly language listing a check is done first to make sure that the computer wasn't interrupted while it was in the middle of doing something where timing is critical, such as talking to the disk drive. Lines 610-650 take care of keybounce, caused by the vibration of the switch contacts inside the keys.

It is also worth noting that this whole routine is executed only in response to a key being pressed. Holding a key down for the auto-repeat does not generate any further interrupts, and repetition of the key is accomplished at a later time during another type of interrupt, the Vertical Blank period. This interrupt occurs sixty times a second during

May you never know the agony of delete (or clearscreen) again!

the time between pictures on your TV screen. The instructions executed during this interrupt are located in an area of the operating system that does not appear to be bypassable. This is also where the keyclicks are generated, making it impossible to shut them off or change the auto-repeat speed itself.

Although it is possible to intercept the vertical blank interrupt, it does not look feasible to replace the operating system ROM routines which take care of these things. If anyone has information to the contrary I'd be very interested in hearing from you. The reason the initial repeat speed can be altered is that the auto-repeat timer, SRTIMR, is initialized during the key-pressed interrupt.

At line 810 bits 6 and 7 of the key code are set to zero since these bits indicate the shift or control key was pressed which is no importance in the case of these two keys. The keycodes themselves have no relation to the ASCII or ATASCII codes. Only the engineers who designed the keyboard could tell you why they created it the way they did, (like why the little pinky got stuck with so many keys to deal with). Anyway, if you want to alter the program to warn you about your own problem keys, you can easily determine their codes with the following one-line BASIC program:

```
10 PRINT PEEK(53769): GOTO 10
```

At lines 1210-1260 you may notice a slightly curious bit of code. VCOUNT is a hardware register which can be read to determine the current scan line being drawn on the TV screen. This value will be zero, sixty times each second, and provides a quick and easy way to obtain a do-nothing loop without tying up any memory locations for counting. Of course, you might wonder why I didn't just use one of the system timers. There are two big reasons why not. One is that I wanted my routine to be as transparent to normal computer operations as possible and tying up a timer could conflict with another program. But a bigger reason is that the system timers are themselves maintained during an interrupt cycle, the old vertical blank period.

Thus, whenever pressing a key causes an interrupt, all other low priority interrupts are temporarily ignored, meaning that the system timers stop. They remain frozen during the time that this program code is being executed and resume counting afterwards. If having the timers lose accuracy is a problem for your application, I suggest adding a little more code which would make an appropriate correction to the timer values. I didn't do this because there is no way the extra code would fit into the cassette buffer. It is being packed to the very last byte already.

Finally, at lines 1400-1430 of the assembly listing, bit 3 of the memory-mapped hardware register named SKSTAT (Serial Port/Keyboard Status) is selected with a logical AND command. This bit indicates whether the last key pressed is still depressed, and is the determining factor as to whether one of those problem keys is actually wanted. May you never know the agony of delete (or clearscreen) again!

Program 1.

```
100 REM . MAKE YOUR KEYBOARD FRIENDLIER
110 REM .     BASIC Program Version
120 REM .
130 REM .        Ric Mears  8/15/81
140 REM
150 REM If you still want to use your
160 REM cassette, switch these two values:
170 REM (but you'll lose use of Page 6)
180 REM
190 CODE=1021
200 INIT=1536
210 REM
220 REM Poke the new code into place:
230 REM
240 FOR I=CODE TO I+130
250 READ OPCODE:POKE I,OPCODE
260 NEXT I
270 REM
280 REM Poke the init code into place:
290 REM
300 FOR I=INIT TO I+11
310 READ OPCODE:POKE I,OPCODE
320 NEXT I
330 REM
340 REM Fix the absolute address in case
350 REM the user has selected different
360 REM locations for the code.
370 REM
380 HIGH=INT(CODE/256)
390 LOW=CODE-HIGH*256
400 POKE INIT+2,LOW
410 POKE INIT+7,HIGH
```

```
420 REM
430 REM Now reset the the keyboard vector:
440 REM
450 INIT=USR(INIT)
460 PRINT :PRINT "ALL DONE"
470 REM
480 REM Here's the assembled program
490 REM as data:
500 REM
510 DATA 165, 66, 208, 81, 138, 72, 152, 72
520 DATA 160, 15, 173, 9, 210, 205, 242, 2
530 DATA 208, 5, 173, 241, 2, 208, 55, 173
540 DATA 9, 210, 201, 159, 208, 10, 173, 255
550 DATA 2, 73, 255, 141, 255, 2, 176, 38
560 DATA 201, 116, 240, 43, 201, 118, 240, 39
570 DATA 201, 182, 240, 35, 41, 63, 201, 39
580 DATA 240, 29, 201, 60, 240, 25, 173, 9
590 DATA 210, 141, 252, 2, 141, 242, 2, 169
600 DATA 3, 141, 241, 2, 133, 77, 140, 43
610 DATA 2, 104, 168, 104, 170, 104, 64, 160
620 DATA 5, 32, 216, 252, 162, 8, 142, 31
630 DATA 208, 162, 75, 173, 11, 212, 208, 251
640 DATA 202, 208, 248, 136, 208, 235, 160, 255
650 DATA 173, 11, 212, 208, 251, 136, 208, 248
660 DATA 160, 40, 173, 15, 210, 41, 4, 240
670 DATA 189, 208, 203
680 REM
690 REM Here's the initialization
700 REM code as data:
710 REM
720 DATA 104, 169, 253, 141, 8, 2, 169, 3
730 DATA 141, 9, 2, 96
740 END
750 REM End of BASIC program listing
```

Program 2.

```
0100 ; MAKE YOUR KEYBOARD FRIENDLIER
0110 ;      Assembler Source Code
0120 ;
0130 ;          Ric Mears  8/15/81
0140 ;
0150 ; EQUATES
0160 ;
0170 VKEYBD   = $208    ;Keyboard vector
0180 KBCODE   = $D209   ;Pokey register
0190 CONSOL   = $D01F   ;Speaker
0200 VCOUNT   = $D40B   ;Scan-line counter
0210 SKSTAT   = $D20F   ;Keyboard status
0220 CLICK    = $FCD8   ;OS keyclick routine
0230 CH       = $2FC    ;Current key pressed
0240 CH1      = $2F2    ;Last key pressed
0250 KEYDEL   = $2F1    ;Keybounce counter
0260 SSFLAG   = $2FF    ;Cntrl-1 start-stop
0270 ATTRACT  = $4D     ;Attract mode flag
0280 CRITIC   = $42     ;Critical code flag
0290 SRTIMR   = $22B    ;Auto-Repeat timer
0300 CNTRL1   = $9F
0310 CLEAR1   = 118     ;Internal
0320 CLEAR2   = 182     ; codes
0330 INVERSE  = 39      ;  for these
0340 CAPSLWR  = 60      ;   problem
0350 DELETE   = 116     ;    keys
0360 ;
0370 ;
0380 ; ENTRY POINT
0390 ;
0400 ; Whenever a key is pressed, an
0410 ; interrupt sends the 6502 here:
0420 ;
0430  *=$3FD            ;Or any safe place
0440 ;
```

```
0450 ; If the computer was executing
0460 ; critical code, then forget
0470 ; about the keyboard interrupt:
0480 ;
0490 NEWPROCEDURE
0500  LDA CRITIC
0510  BNE EXIT
0520 ;
0530  TXA                 ;OS has already
0540  PHA                 ; saved register A,
0550  TYA                 ;  must also
0560  PHA                 ;   save X & Y
0570 ;
0580 ;
0590  LDY #$F             ;Set new Auto-
0600 ;                    ; repeat speed
0610  LDA KBCODE
0620  CMP CH1             ;Same as last key?
0630  BNE NEWKEY
0640  LDA KEYDEL          ;If KEYDEL > 0
0650  BNE OUT             ; ignore as bounce
0660 NEWKEY
0670  LDA KBCODE
0680  CMP #CNTRL1         ;Take care of
0690  BNE NOTCTRL1        ; Control-1
0700  LDA SSFLAG          ;  stall flag
0710  EOR #$FF
0720  STA SSFLAG
0730  BCS OUT
0740 NOTCTRL1
0750  CMP #DELETE         ;Check for
0760  BEQ ALERT           ; aggravating
0770  CMP #CLEAR1         ;    keys
0780  BEQ ALERT
0790  CMP #CLEAR2
0800  BEQ ALERT
0810  AND #$3F            ;Strip off shift
0820  CMP #INVERSE        ; & cntl bits
0830  BEQ ALERT           ;  since these
0840  CMP #CAPSLWR        ;   keys are
0850  BEQ ALERT           ;    unique
0860 ;
0870 ;
0880 ; This point reached if a regular key
0890 ; or the typist wants the special key
0900 ;
0910 NOCHANGE
0920  LDA KBCODE
0930  STA CH              ;Pass the
0940  STA CH1             ; letter on
0950  LDA #3
0960  STA KEYDEL          ;Set debounce
0970  STA ATTRACT         ;Reset Attract flag
0980 OUT
0990  STY SRTIMR          ;Set auto-repeat
1000 ;                    ;      speed
1010  PLA
1020  TAY                 ;Restore
1030  PLA                 ; registers
1040  TAX
1050 EXIT
1060  PLA                 ;And return from
1070  RTI                 ; the interrupt.
1080 ;
1090 ; ALERT ROUTINE
1100 ;
1110 ; A problem key has been pressed
1120 ; so do the special signal.
1130 ;
1140 ALERT
1150  LDY #5              ;For 5 clicks
1160 LOOP5
1170  JSR CLICK
1180  LDX #8              ;Turn speaker
1190  STX CONSOL          ; back off
1200 ;
1210  LDX #75             ;For stall length
1220 WAIT
1230  LDA VCOUNT          ;Scan line count
1240  BNE WAIT            ; (extra delay)
1250  DEX
1260  BNE WAIT
1270  DEY
1280  BNE LOOP5           ;5 clicks yet?
1290 ;
1300  LDY #255
1310 WAITAGAIN
1320  LDA VCOUNT          ;Stall a moment
1330  BNE WAITAGAIN       ; after sounding
1340  DEY                 ;  the alert
1350  BNE WAITAGAIN
1360 ;
1370  LDY #40             ;Slower initial
1380 ;                    ; repeat for these
1390 ;
1400  LDA SKSTAT          ;Are they still
1410  AND #4              ; holding the
1420  BEQ NOCHANGE        ;  key down?
1430  BNE OUT
1440 ;
1450 ;
1460       *= $600
1470 ;
1480 ; INIT POINT
1490 ;
1500 ; Resets keyboard vector so that
1510 ; whenever a key-pressed interrupt
1520 ; occurs, the 6502 will go the new
1530 ; routine.
1540 ;
1550 INIT
1560  PLA
1570  LDA #NEWPROCEDURE & $00FF
1580  STA VKEYBD
1590  LDA #NEWPROCEDURE / 256
1600  STA VKEYBD+1
1610  RTS
1620 ;     End of Assembly Program Listing
```

Adding High Speed Vertical Positioning To P/M Graphics

David H. Markley
Reynoldsburg, OH

By now many of you have been playing the "Aliens from Outer Space" program I described in **COMPUTE!** #11, and have been able to experiment with an actual game program incorporating the advanced player/missile graphics of the Atari computer. As you may have observed, player images can be moved horizontally across the playfield quite easily just by placing the player's horizontal coordinate (0-120) into its associated horizontal position register. Vertical positioning with the P/M graphics however is somewhat more difficult. Since the player's vertical position on the playfield inversely corresponds to its position within the image buffer, it is necessary to relocate each byte of the image up or down within the buffer to produce vertical movement. For example, if we move the player's image to higher address locations within the image buffer, the player will appear to move downward on the playfield.

A BASIC routine can be written using PEEKs and POKEs to move the player within the image buffer, but for most applications this method is too slow. An alternative, however, is to use a small, general purpose vertical positioning routine written in 6502 assembly code which can be called by BASIC's USR instruction.

The vertical positioning routine shown in Program 1 is relatively simple, but provides the user with a flexible and easy method of handling P/M graphics within a BASIC program. This program not only provides a valuable tool to use with player missile graphics, but for those of you who have not used assembly language routines with BASIC, it will provide some insight into this area as well. The routine is called by a BASIC statement similar to below:

```
DUMMY = USR(VP,IMAGE,LAST LOCATION,
        NEW LOCATION)
```

The variable to the left of the equal sign called "DUMMY" is used by some machine language subroutines as a target for a value returned by the program. The vertical positioning routine however, does not return a usable value, but the DUMMY variable is still required to satisfy Atari's USR format requirements. Within the parentheses of the command are four arguments. The first argument, VP, is the transfer address to the VP routine which has been placed into a free area of memory. Loading of the VP routine into memory will be described later with a program application example. Following the transfer address argument (which, by

Before either step can be executed, the routine must first look at the image's data structure and get the image size parameter.

the way, is also required for any USR routine called by BASIC) are three arguments which are passed to the VP routine.

These arguments are the address of the image's data structure, the address of the image's current position in the P/M image buffers, and the address of its new position. Each image requires a small data structure which provides the VP routine with a pattern of the actual image which it will vertically reposition. An example of a typical image data structure is shown in Figure 1. The first byte of data provides the VP routine with the image's size in bytes. The second and following bytes are used to form a bit map pattern of the image as it would appear in the P/M image buffers.

The next two arguments contained in the USR command tell the VP routine the image's current and new positions. These arguments are actual memory addresses into the image buffers, therefore care must be taken to assure that they do not access another area of memory by mistake.

Routine Operation

The program operation begins with an initialization step in which the three arguments passed to it by the USR command are removed from the processor's stack and placed into an area in page zero where they can be more easily used. You may have noticed that a total of seven bytes are popped off the stack during this operation. This is because the USR command always places a one byte argument count onto the stack followed by the arguments themselves. The arguments are always two bytes in length.

Once the initialization task is complete, the routine is ready to begin its intended task of moving the player image. Basically the operation is performed in two steps. The image data is first removed from its current location and then copied to its new location. Before either step can be executed, the routine must first look at the image's data structure and get the image size parameter. This value tells the routine how large an image it must handle and thus determines the number of bytes it must remove and restore. To remove an image from its current location, the routine simply goes to the current location address and writes zeroes into an X number of memory locations indicated by the size parameter. Replacement of the image is done by copying from the image's data structure an X number of bytes, also determined by the size parameter, to the image buffer starting at the address specified by the new position argument.

In some cases it may not be desirable to have the VP routine perform both the delete and restore functions. One example would be if the player image is to be removed from the viewing field and not restored at a new location. This can be handled by using the following routine call:

```
DUMMY = USR(VP,IMAGE,CURRENT
            LOCATION,0)
```

The zero in the new location field tells the VP routine not to attempt to restore the image. Likewise, the delete function can be disabled by placing a zero in the current location field.

Let's Have Some Fun

Now that we have looked at the Player/Missile Vertical Positioner routine, let's put it to work. The following game will show you how to load the player images and VP routine into memory and how to use the routine in other ways besides vertical positioning.

This game which I call "Island Jumper" involves the cooperation of two characters named Crash Coleman and Deadeye Dan. Crash is the pilot of a reliable (but not so stable) airplane, the "Leaping Lucy." Crach has only had one flying lesson, but has courageously volunteered to make this flight so that you can see the VP routine in action. Although he has successfully managed to get the Leaping Lucy off the ground, he seems to be having some trouble keeping her in level flight. Our other daredevil of the sky, Deadeye Dan, will attempt, with your help, to jump out of Crash's airplane and land on Target Island. Since the ground seems to be a bit unstable from Dan's point of view, he is having difficulty figuring out when to jump and asks that you help him by pulling back on your joystick controller when you think he's on target.

Dan will make a total of five jumps each time you play the game. He will try to land on top of a sand dune located on the left side of the island. If he makes the jump on Crash's first pass over the island and lands on the dune with both feet, you get 30 points. If you don't give Dan the signal to jump during the first pass, Crash will continue to fly over the island until a jump is made. Each additional pass will deduct eight points from Dan's maximum obtainable score.

Dan can also land in the area between the sand dune and the palm tree, but you will only receive a maximum of 15 points for the jump. At the completion of the game, the computer will give you both a final score the last game played and the highest score for all games played since the last RUN command was entered. To play another game, press the button on the joystick controller.

The data for the VP routine and the player data structures is read from data statements and POKEd into memory by lines 110 thru 310 of the program. It is loaded into memory page 6 (starting at address 1536) which is a 256 byte area in memory that Atari has reserved for user binary data and machine language routines. Once the data structures and VP routine are loaded into memory, they are referenced in the BASIC program by variable names whose values have been set to the starting address of the data structure or VP routine they represent.

Program 1.

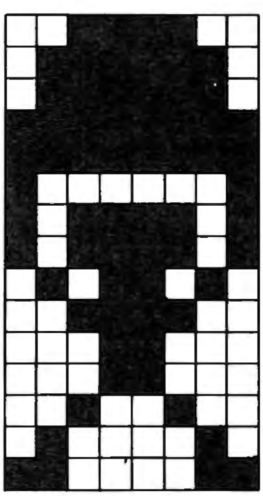
```
10 REM VERTICAL POSITIONER EXAMPLE
20 REM ISLAND JUMPER
30 REM COPYRIGHT (c) 1981
40 REM BY DAVID H. MARKLEY
50 GRAPHICS 2:POKE 752,1
60 SETCOLOR 4,9,4
70 ? #6:? #6:? #6:? #6;"      ISLAND"
80 ? #6:? #6;"      JUMPER"
90 ? ,"      BY"
100 ? :? ,"DAVID MARKLEY"
110 VP=1536
120 FOR G=0 TO 93
125 READ D
130 POKE VP+G,D
135 NEXT G
140 REM VERTICAL POSITIONER CODE
150 DATA 104,162,5,104,149,220,202,16,25
0,198,220,198,222,160,0,177,224,170
160 DATA 168,165,223,240,9,169,0,145,222
,136,208,249,138,168,165,221,240,7,177,2
24,145,220,136,208,249,96
170 REM AIRPLANE DATA
180 APIMG=VP+44
190 DATA 6,142,132,255,255,4,14
```

```
200 REM JUMPER DATA
210 JPIMG=APIMG+7
220 DATA 9,189,189,90,60,24,24,36,66,129

230 REM JUMPER AND CHUTE DATA
240 JSIMG=JPIMG+10
250 DATA 15,60,126,126,255,255,129,189,1
89,90,60,24,24,36,66,195
260 REM WAVING JUMPER
270 JWIMG=JSIMG+16
280 DATA 15,0,0,0,0,0,128,188,188,88,60,
26,25,37,66,195
290 REM DATA USED TO CLEAR MEMORY
300 CLEAR=JWIMG+16
310 DATA 255
320 FOR D=1 TO 300:NEXT D
330 GRAPHICS 5
340 SETCOLOR 2,9,2
350 SETCOLOR 4,8,6
360 I=PEEK(106)
365 X=I*256-1172
370 POKE X,112
371 POKE X+1,71
372 POKE X+2,96
373 POKE X+3,I-1
374 POKE X+4,112
375 POKE X+5,74
376 POKE X+6,160
377 POKE X+7,I-5
380 I=I-8
390 POKE 54279,I
400 J=I*256+513
410 POKE 559,46
420 POKE 53256,1
430 POKE 53277,3
440 POKE 704,56
450 POKE 705,12
460 D=USR(VP,CLEAR,J,0)
465 SLOPE=2
470 TOP=J+17
480 BOT=J+55
490 SETCOLOR 0,12,8
500 SETCOLOR 1,1,2
510 COLOR 2
520 PLOT 37,34:DRAWTO 42,34
530 PLOT 36,35:DRAWTO 49,35
540 PLOT 47,29:DRAWTO 47,34
550 COLOR 1
560 PLOT 43,30:DRAWTO 47,27
570 PLOT 51,30:DRAWTO 47,27
580 PLOT 47,27:DRAWTO 49,30
590 PLOT 47,27:DRAWTO 45,30
600 PLOT 46,27
610 HSCORE=0
620 LAPOS=0
630 APOS=J+70
640 I=-1
650 JUMP=5
660 SCORE=0
670 PNTS=30
680 JMP=0
690 SOUND 0,31,4,4
700 POKE 623,4
710 JSTOP=J+219
720 FOR G=20 TO 245 STEP 3
730 POKE 53248,G
740 D=USR(VP,APIMG,LAPOS,APOS)
750 IF JMP=0 AND G<180 AND STICK(0)<>15
THEN JMP=APOS+132:POKE 53249,G+4:IMG=JPI
MG:D=USR(VP,IMG,0,JMP)
760 LJMP=JMP
770 IF JMP=0 THEN 880
780 JMP=JMP+3
790 IF JMP<J+200 THEN HJMP=G+4:POKE 5324
9,HJMP:SOUND 1,G,10,8:GOTO 860
800 IMG=JSIMG
804 JMP=JMP-2
808 SOUND 1,0,0,0
810 IF HJMP>=122 AND HJMP<=126 THEN JSTO
P=J+208:GOTO 860
820 IF HJMP<120 OR HJMP>134 THEN 860
830 JSTOP=J+210
840 POKE 623,1
850 IF PNTS>15 THEN PNTS=15
860 IF JMP>JSTOP THEN 940
870 D=USR(VP,IMG,LJMP,JMP)
880 LAPOS=APOS
890 APOS=APOS+I
900 D=USR(VP,APIMG,LAPOS,APOS)
910 IF APOS>BOT THEN I=-SLOPE
920 IF APOS<TOP THEN I=SLOPE
930 NEXT G
940 IF JMP<J AND PNTS>9 THEN PNTS=PNTS-8
:GOTO 1220
950 IF JMP<J THEN 1220
970 IF HJMP<120 OR HJMP>134 THEN TONE=8:
GOTO 1010
980 SCORE=SCORE+PNTS
985 TONE=12
990 D=USR(VP,JWIMG,0,JMP-1)
1000 ? "SCORE ";SCORE:? :?
1010 FOR D=15 TO 0 STEP -1
1020 SOUND 1,12,TONE,D
1030 FOR I=1 TO 10:NEXT I
1040 NEXT D
1050 SOUND 0,0,0,0
1055 SOUND 1,0,0,0
1060 JUMP=JUMP-1
1070 IF JUMP<>0 THEN 1170
1080 IF SCORE>HSCORE THEN HSCORE=SCORE
1090 FOR I=1 TO 120
1100 IF I=1 THEN ? "HIGH SCORE ";HSCORE:
```

```
? :?
1110 IF I=60 THEN ? "FINAL SCORE ";SCORE
:? :?
1120 IF STRIG(0)=1 THEN 1150
1130 D=USR(VP,CLEAR,J,0)
1135 PRINT
1140 GOTO 630
1150 NEXT I
1160 GOTO 1090
1170 ? "JUMP ";6-JUMP:? :?
1180 FOR D=0 TO 250:NEXT D:?
1190 D=USR(VP,CLEAR,J,0)
1195 I=SLOPE
1200 IF RND(0)>0.5 THEN I=-SLOPE
1210 GOTO 670
1220 POKE 77,0
1225 GOTO 690
1230 END
```

Figure 1: Image Data Structure for the Player/Missile Vertical Positioner Routine.

Image Pattern



Byte Number	Byte Value
1	60
2	126
3	126
4	255
5	255
6	129
7	189
8	189
9	90
10	60
11	24
12	24
13	36
14	66
15	195

DATA 15,60,126,126,255,255,129,189,189,90,60,24,24, 36,66,195

Program 2. Assembly language representation of the P/M Vertical Positioner Routine

```
10  ; P/M VERTICAL POSITIONER
20  NEW=220
30  CURRENT=222
40  IMAGE=224
50  START    PLA              ; REMOVE ARGUMENT
                              BYTE COUNT
60           LDX #5           ; REMOVE 6 BYTES
70  LP1      PLA              ; AND PLACE IN PAGE
                              ZERO
80           STA NEW,X
90           DEX
100          BPL LP1
110          DEC NEW
120          DEC LAST
130          LDY #0
140          LDA (IMAGE),Y    ; GET IMAGE BYTE
                              COUNT
150          TAX
160          TAY
170          LDA LAST+1       ; IF ZERO DON'T DELETE
180          BEQ SKIPD
190 LP2      LDA #0           ; DELETE IMAGE
200          STA (LAST),Y
210 DEY
220          BNE LP2
230 SKIPD    TXA
240          TAY
250          LDA NEW+1        ;IF ZERO DON'T
                              RESTORE
260          BEQ SKIPR
270 LP3      LDA (IMAGE),Y    ; COPY IMAGE DATA TO
                              NEW
280          STA (NEW),Y      ; ADDRESS
290          DEY
300          BNE LP3
310 SKIPR    RTS
```

A Poor Programmer's Word Processor

Frank Roberts
Ft. Wayne, IN

A few days ago I walked into a local computer store for new software for my Atari 800. I was informed — just as expected — that there wasn't much available yet, "but a lot was expected real soon." Well, being impatient, low on cash and desperately wanting something besides Star Raiders to play with I decided to tackle one of those "soon-to-be-available" projects myself. After all, didn't I really buy my Atari to have fun with? The following program is a primitive (but workable) method of justifying left and right margins for letters and texts — sort of a poor man's word processor.

Program 1.

```
1 REM ***** PSEUDO WORD PROCESSOR (D:WRI
TE.LET)
5 GRAPHICS 0
10 DIM A$(100),B$(100)
20 REM
30 ? :? "HOW MUCH MARGIN ";:INPUT MARGIN

40 ? :? "ENTER TEXT (IN DOUBLE LINES) "
50 ? :? "WHEN FINISHED, ENTER '999'"
60 ? :? "TO BEGIN, HIT RETURN ";:INPUT A
$
70 WIDTH=INT(80-(2*MARGIN)):POKE 83,INT(
WIDTH)/2+2
80 POKE 201,MARGIN-1
90 INPUT A$:IF A$="999" THEN 140
95 IF LEN(A$)=0 THEN 120
100 IF LEN(A$)=WIDTH+1 THEN B$=A$:GOTO 1
20
110 GOSUB 1000
120 LPRINT " ",B$:B$="":PRINT
130 GOTO 90
140 END
1000 REM *****SUBROUTINE: JUSTIFY RIGHT
MARGIN
1010 C=0
1020 FOR J=1 TO LEN(A$)
1030 IF LEN(B$)=WIDTH+1 THEN 1070
1040 C=C+1:B$(C)=A$(J)
1050 IF A$(J,J)=" " THEN C=C+1:B$(C,C)="
"
1060 NEXT J
1070 RETURN
```

Program 2.

```
20 OPEN #1,8,0,"D:FILE.LET"
50 ? :? "DO YOU WANT PRINTOUNT NOW ";:IN
PUT A$
80 PRINT #1;MARGIN
120 PRINT #1;B$:B$="":PRINT
160 IF A$="NO" THEN END
170 RUN "D:TYPE.LET"
```

Program 3.

```
1 REM *** FETCH TEXT AND PRINT (D:TYPE.L
ET)
10 DIM A$(100)
20 ? :? "WHEN READY, HIT RETURN ";:INPUT
 A$
30 OPEN #1,4,0,"D:FILE.LET"
40 TRAP 80
50 INPUT #1,MARGIN:POKE 201,MARGIN
60 INPUT #1,A$:LPRINT " ",A$
70 GOTO 60
80 CLOSE #1
90 ? :? "DESTROY FILE NOW (Y-N) ";:INPUT
 A$
100 IF A$(1,1)="N" THEN END
110 XIO 33,#1,0,0,"D:FILE.LET"
```

How It Works

Line 70 calculates the parameters of the text string according to the MARGIN selected and POKEs the right margin of the screen monitor to one-half the width. The latter is necessary to accomodate one full line of typing for each A$ to be printed (the screen is only 40 columns wide). The user enters two lines for each single line of text and types as close to the right margin as possible without hyphenating the last word in the middle of a syllable. The subroutine beginning at LINE 1000 counts the characters within each line of text and adjusts the length of the text by inserting the proper number of spaces into the string.

The program is designed to print directly to an AXIOM II printer, but with the following modifications (Program 2) a temporary file (D:FILE.LET) can be made which will allow storage of the text for future printout or multiple copy.

There is also a file retrieval program (Program 3; LINE 110 automatically deletes the unwanted file without going to DOS).

COMPUTE! Magazine

When we first started **COMPUTE!** magazine in the Fall of 1979, we made a commitment to build a strong, ongoing users resource. Since that beginning, we've provided thousands of personal computer owners with breakthroughs, innovations, practical software, and in-depth reviews of new and significant products. In every issue of **COMPUTE!**, you'll find dozens of pages showing you exactly how to make better use of your computer; articles which tutor the beginner and challenge the advanced.

We're your specialists, whether your interests lie in home applications, game development, educational insights, or better business planning. Here are just a few facts to tell you more about **COMPUTE!**:

' In every issue, you'll find clearly, cleanly printed software you can type right into your computer and use...

Here's a comparison of average machine specific articles in the last three issues of **COMPUTE!**, matched against the average number of articles in the "Big Four" ...the personal computer magazines with circulations of 100,000 or more:

Computer	COMPUTE!		"Big Four"	
Type	Articles	Programs	Articles	Programs
Atari	10.67	13.3	1.08	1.42
PET/CBM	11.33	11.0	1.5	2.33
Apple	4.0	6.0	3.58	3.17
OSI	2.0	4.0	.5	.5

(Numbers are average number of articles or programs in **COMPUTE!** issues 16, 17, and 18 compared with the average number of articles or programs in each of the same three issues of the four largest personal computing magazines.)

Subscribe to **COMPUTE!** today! We're the Resource...

Announcing Home and Educational COMPUTING!, the newest publication from the publishers of COMPUTE!

We've expanded the scope of our new magazine and, beginning in January, you'll find resourceful coverage and helpful information for the **VIC-20, the Atari 400, the TI 99/4A, the Radio Shack Color Computer, the Sinclair ZX-81,** and more. We'll be covering personal applications, education, new thresholds, the pocket computers, and a great deal more.

COMPUTE! Books

Our **First Book of Atari** is now shipping (if you've already ordered one it's on the way). If you haven't, check with your dealer, or order directly from us by using the convenient toll-free numbers below.

In the continental US, Call TOLL FREE 800-345-8112. In PA call 800-662-2444.

SUBSCRIPTION RATES

	COMPUTE! (monthly) Twelve Issues	Home And Educational COMPUTING! (bimonthly) Six Issues
US:		
One Year	$20.00	$10.00
Two Years	$36.00	
Three years	$54.00	
Canada	$25.00	$12.00
Europe, Air Delivery	$38.00	$18.00
Elsewhere, Surface	$25.00	$12.00
Air:		
Middle East, Central America and North Africa	$48.00	$24.00
South America, South Africa, Far East and Australia	$88.00	$24.00

All subscriptions from outside the US must be prepaid in US funds drawn on a US bank or international money order. Visa/MasterCard also accepted.

Please enter my subscription to: ☐ **COMPUTE!**
☐ 1 year ☐ 2 years ☐ 3 years
☐ **Home and Educational COMPUTING!**

Please send me ______ copy(s) of **COMPUTE!'s First Book of ATARI** at $12.95 each + $2.00 shipping/handling.

Name ______________________

Address ______________________

____

For fastest service in the US call the toll free number above, or send this coupon to: **COMPUTE!**, 515 Abbott Drive, Broomall, PA 19008.

Bill me orders (US only) are charged a $1.00 billing fee.

COMPUTE!
515 Abbott Drive
Broomall, PA 19008

THE OSI GAZETTE

Super Cursor V1.3

Frank Cohen
Pacific Palisades, CA

My biggest complaint about Ohio Scientific's Superboard II has been about the awful video output. It's almost ironic noting all the good things the Superboard has going for it: a nice keyboard; a powerful Microsoft BASIC in ROM; a dependable cassette interface; 8K of RAM; and many other functions. The irony comes into play when you turn on the Superboard and take a look at the 24 by 24 video. And it gets worse as you start to use BASIC to list programs the effective display size becomes 23 by 20.

In reading through *The First Book of OSI*, from Elcomp Publishing, I found that a company names Silver Spur Electronics, in Chino, California, sells detailed instructions to double the display size by adding several jumpers and a couple of I.C.'s to the board. The modified display yields an effective display size of about 26 lines of 48 characters (which can be enlarged if you don't want a border around the display).

After making the modifications, though, the BASIC in ROM still thinks the memory map of the video display is the same, and so it only uses half the screen. Included with the modification instructions is a software patch which will allow BASIC to utilize the whole display. However, that, too, gives you only a *very* simple cursor. Using other computers I

```
;SUPER CURSOR V1.3
;Written by Frank Cohen
;
;Cursor Routine for OSI Superboard II
;to suppliment Microsoft's Basic-in-ROM
;cursor functions.
;
;Note: This program works with Steven
;Chalfin's video modifications and needs
;to be changed to work with a Superboard's
;normal 24 by 24 video. At the end of this
;listing are the changes for 24 by 24 video.
;
;
;This program loads into 1E40-1FE7 hex
;which is the top of memory on an 8K
;Superboard II. It may be reassembled for other
;addresses if desired.
;
;Directions: Once loaded the following must
;be done to start Super Cursor-
; 1. Set the Zero page locations
; 2. Cold start BASIC limiting the memory size
;     to 7624 (dec.).  MEM SIZ? 7624
; 3. Poke the following-
;        POKE 538,64:POKE539,30
;At this point a solid white cursor should
;appear at the home position (upper left corner)
;If this happens you have successfully loaded
;Super Cursor V1.3. If not, try it again.
;
;Options:
; To turn off the scrolling function-
;   POKE 7861,128:POKE 7862,30
; To turn on the scrolling function-
;   POKE 7861,105:POKE 7862,31
; To change the cursor symbol-
;   POKE 8033,X  (where x is a graphics number)
;
;HOME LOCATION = DOCC (hex)
;Horizontal Boundary = 44   (2C hex)
;Verticle Boundary = 26     (1A hex)
;
;BASIC Commands-
; Clear Screen = PRINT CHR$(1)
; Home Cursor = PRINT CHR$(2)
;
;Zero Page Usage
;>MR 1 80       >33 80 06
00E0 CC                 CURSLOC LOW;Cursor Location Low byte
00E1 D0                 CURSLOC HI ;Cursor Location High byte
00E2 20                 TEMP       ;Stores byte from cursor location
00E3 00                 HL         ;Horizontal Location of Cursor
00E4 00                 VL         ;Verticle Location of Cursor
00E5 00                 SCURS LOW  ;16 Bit scratch pad register
00E6 00                 SCURS HI   ;
                                   ;
                                   ;
1E40                               ;Start of Program
1E40 8D 02 02   Start   STA 0202        ; Save all register onto the
1E43 48                 PHA             ; the stack
1E44 8A                 TXA
1E45 48                 PHA
1E46 98                 TYA
1E47 48                 PHA
1E48 AD 02 02           LDA 0202
1E4B C9 5F      Menu    CMP $5F         ;Check key pressed for cursor
1E4D D0 03              BNE NDE         ;function
1E4F 20 BE 1F           JSR Left
1E52 C9 02      NDE     CMP $02
1E54 D0 03              BNE NHO
1E56 20 80 1E           JSR Home
1E59 C9 0D      NHO     CMP $0D
1E5B D0 03              BNE NCR
1E5D 20 95 1E           JSR CR
1E60 C9 0A      NCR     CMP $0A
1E62 D0 03              BNE NLF
1E64 20 AB 1E           JSR LF
1E67 C9 01      NLF     CMP $01
1E69 D0 03              BNE NCL
1E6B 20 C2 1E           JSR CLS
1E6E C9 00      NCL     CMP $00
1E70 F0 03              BEQ Exit
1E72 20 E8 1E           JSR Dispc
1E75 68         Exit    PLA             ;Restore all the resisters from
1E76 A8                 TAY             ;the stack
```

found that I really liked being able to Home, or Clear Screen, or Line Feed, or Backspace the cursor. All these are not possible with the cursor program in the ROM.

Super Cursor solved my needs for an advanced cursor program. In addition to the above functions, it can actually Backspace the cursor (the BASIC in ROM version prints another underline), you can define what the cursor looks like by picking any of the graphics characters available, you can also scroll at the bottom of the display or wrap around to the top. All these functions are available in BASIC or you can use Super Cursor from a machine language program.

If you have not installed the video modifications for the larger display size you will need to modify several locations in Super Cursor. These modifications can be found in the listing after Super Cursor's machine language listing.

In operating Super Cursor, some steps must be taken to tell BASIC to use Super Cursor rather than its old cursor. First load Super Cursor into memory. If you have an assembler, you can reassemble it to fit anywhere in memory. It occupies approximately 425 bytes of memory. If you don't have an assembler, I would not advise trying to move Super Cursor as almost everything uses subroutines which need absolute addresses (you would have to renumber everything). Super Cursor, as it is listed, fits into the top portion of an 8K Superboard II.

Once loaded, it is necessary to set up the page zero memory vectors. There are seven bytes in all which must be set as follows:

00E0 CC D0 20 00 00 00 00

After you have completed this, you can cold-start BASIC. Be sure to limit BASIC's memory size to only 7624 bytes or else you will wipe out Super Cursor. To limit BASIC's memory, enter:

```
1E77 68                 PLA
1E78 AA                 TAX
1E79 68                 PLA
1E7A 4C 6C FF           JMP FF6C.           ;Jump back to BASIC
1E7D EA EA EA           NOP                 ;For future expansion
                                            ;
1E80 20 53 1F   Home    JSR TC              ;Home routine
1E83 A9 D0              LDA $D0             ;Set Cursloc to D0CC
1E85 85 E1              STA Cursloc Hi
1E87 A9 CC              LDA $CC
1E89 85 E0              STA Cursloc Lo
1E8B A9 00              LDA $00             ;Set HL and VL to 00
1E8D 85 E4              STA VL
1E8F 85 E3              STA HL
1E91 20 60 1F           JSR SC
1E94 60                 RTS
                                            ;
1E95 20 53 1F   CR      JSR TC              ;Carrage Return
1E98 A9 00              LDA $00             ;Subtract HL from Cursloc
1E9A 85 E6              STA SCURS HI
1E9C A5 E3              LDA HL
1E9E 85 E5              STA SCURS LO
1EA0 20 B0 1F           JSR SBCC
1EA3 A9 00              LDA $00
1EA5 85 E3              STA HL
1EA7 20 5A 1F           JSR CT
1EAA 60                 RTS
                                            ;
1EAB A5 E4      LF      LDA VL              ;Line Feed
1EAD C9 19              CMP $19             ;Check for Scroll
1EAF D0 0A              BNE LFA
1EB1 20 53 1F           JSR TC              ;Carrage return and Scroll
1EB4 20 69 1F           JSR Scroll
1EB7 20 95 1E           JSR CR
1EBA 60                 RTS
1EBB 20 95 1E   LFA     JSR CR              ;No scroll
1EBE 20 27 1F           JSR DOWN
1EC1 60                 RTS
                                            ;
1EC2 A2 00      CLS     LDX $00             ;Clear Screen
1EC4 A9 20              LDA $20             ;Set up
1EC6 9D 00 D0   CLA     STA D000,X
1EC9 9D 00 D1           STA D100,X
1ECC 9D 00 D2           STA D200,X
1ECF 9D 00 D3           STA D300,X
1ED2 9D 00 D4           STA D400,X
1ED5 9D 00 D5           STA D500,X
1ED8 9D 00 D6           STA D600,X
1EDB 9D 00 D7           STA D700,X
1EDE CA                 DEX
1EDF F0 03              BEQ CLSE
1EE1 4C C6 1E           JMP CLA
1EE4 20 5A 1F   CLSE    JSR CT
1EE7 60                 RTS
                                            ;
1EE8 85 E2      DISPC   STA Tempreg         ;Display a character
1EEA A5 E3              LDA HL
1EEC C9 2C              CMP $2C
1EEE F0 04              BEQ DISA            ;Check for a line overflow
1EF0 20 FB 1E           JSR Right
1EF3 60                 RTS
1EF4 20 95 1E           JSR CR              ;Carrage return and line feed
1EF7 20 AB 1E           JSR LF
1EFA 60                 RTS
                                            ;
1EFB 20 53 1F   RIGHT   JSR TC              ;Cursor Right
1EFE A5 E3              LDA HL              ;Check for overflow
1F00 C9 2C              CMP $2C
1F02 F0 10              BEQ RA
1F04 E6 E3              INC HL              ;Increment Cursor
1F06 A9 00              LDA $00
1F08 85 E6              STA SCURS HI
1F0A A9 01              LDA $01
1F0C 85 E5              STA SCURS LO
1F0E 20 A2 1F           JSR ADDC
1F11 4C 23 1F           JMP FRI
1F14 A9 00      RA      LDA $00             ;Subtract 2C from Cursor
1F16 85 E3              STA HL
1F18 A9 00              LDA $00
1F1A 85 E6              STA SCURS HI
1F1C A9 2C              LDA $2C
1F1E 85 E5              STA SCURS LO
1F20 20 B0 1F           JSR SBCC
1F23 20 5A 1F   FRI     JSR CT
1F26 60
                                            ;
1F27 20 53 1F   DOWN    JSR TC              ;Cursor Down
1F2A A5 E4              LDA VL              ;Check for overflow
1F2C C9 1A              CMP $1A
1F2E F0 10              BEQ DDN
1F30 E6 E4              INC VL              ;Add 40 to Cursor
1F32 A9 00              LDA $00
1F34 85 E6              STA SCURS HI
1F36 A9 40              LDA $40
1F38 85 E5              STA SCURS LO
1F3A 20 A2 1F           JSR ADDC
```

7624, in response to the cold-start question: Mem Siz?

Now that you are running BASIC, all you have to do is to type POKE 538,64:POKE 539,30 and press ENTER. You should see the solid white cursor in the upper left (HOME) position of the display. If you hit the space bar, it should move. If it doesn't behave properly then go back into the Monitor Program and check to see if you entered Super Cursor correctly. It is quite easy to make a typing mistake with machine language programs.

If you don't want the cursor to scroll when it reaches the bottom of the screen, you can turn off the scroll function by typing: POKE 7861,128:POKE 7862,30. You can also turn on the scroll function by typing POKE 7861,105:POKE 7862,31. If you want to change the cursor symbol to some other graphics character, all you have to do is to type POKE 8033,x (where x is the graphics

```
1F3D 4C 4F 1F           JMP FDN
1F40 A9 00      DDN     LDA $00          ;Subtract 0640 from Cursor
1F42 85 E4              STA VL
1F44 A9 06              LDA $06
1F46 85 E6              STA SCURS HI
1F48 A9 40              LDA $40
1F4A 85 E5              STA SCURS LO
1F4C 20 B0 1F           JSR SBCC
1F4F 20 5A 1F   FDN     JSR CT
1F52 60                 RTS
                                         ;
1F53 A5 E2      TC      LDA TEMPREG      ;Temp reg. goes to Cursor location
1F55 A0 00              LDY $00
1F57 91 E0              STA (CURSLOC),Y
1F59 60                 RTS
                                         ;
1F5A A0 00      CT      LDY $00          ;Cursor location goes to Temp reg.
1F5C B1 E0              LDA (CURSLOC),Y
1F5E 85 E2              STA TEMPREG
                                         ;
1F60 A9 A1      SC      LDA $A1          ;Cursor symbol goes to Cursor location
1F62 A0 00              LDY $00
1F64 91 E0              STA (CURSLOC),Y
1F66 A9 00              LDA $00
1F68 60                 RTS
                                         ;
1F69 20 53 1F   SCROLL  JSR TC           ;Scroll display one
1F6C A9 20              LDA $20          ;Set up SCURS
1F6E 85 E2              STA TEMPREG
1E70 A9 00              LDA $00
1E72 85 E5              STA SCURS LO
1E74 A9 D0              LDA $D0
1E76 85 E6              STA SCURS HI
1F78 A2 00      SCRT    LDX $00          ;Scroll it
1F7A A0 40              LDY $40
1F7C B1 E5              LDA (SCURS),Y
1F7E 81 E5              STA (SCURS),X
1F80 A5 E5              LDA SCURS LO
1F82 18                 CLC
1F83 69 01              ADC $01
1F85 85 E5              STA SCURS LO
1F87 90 02              BCC SCAT
1F89 E6 E6              INC SCURS HI
1F8B A5 E6      SCAT    LDA SCURS HI
1F8D C9 D8              CMP $D8
1F8F F0 03              BEQ SCON
```

number). Normally, the cursor is equal to 161, which is a white box. If you want to Home the cursor type, PRINT CHR$(2). If you want to Clear the screen type PRINT CHR$(1).

Until I began to use the Home and Clear functions, I didn't realize what could be accomplished in a BASIC program. The following is a short program which tests the Random Number Generator of the Superboard's Microsoft BASIC. By running this program, you will see the screen being updated as though the program POKEs the display with the correct information. Actually, the use of the HOME function is all that is being utilized.

```
 10 REM RANDOM NUMBER
    GENERATOR TEST
        Remarks
 20 DIM A(9)
 30 PRINT CHR$(1),CHR$(2)
        Clear and Home
 40 POKE 8033,32
        Change the cursor to a space
 50 FOR I=1 TO 1000
 60 X=INT(RND(1)*10)
 70 A(X)=A(X)+1
 80 PRINT CHR$(2)
        Home the cursor
 90 FOR J=0 TO 9
100 PRINT J;"=";A(J)
110 NEXT J
120 PRINT"SAMPLE=";X
130 PRINT"I=";I
140 NEXT I
150 POKE 8033,161
        Restore cursor
160 END
```

As you can see by running this program, working with the Superboard II gets easier and easier with the help of an advanced cursor program like Super Cursor V1.3.


```
1F91 4C 78 1F              JMP SCRT
1F94 A2 40        SCON     LDY $40
1F96 A9 20        SCA      LDA $20          ;Blank bottom line
1F98 9D C0 D7              STA D7C0
1F9B CA                    DEY
1F9C D0 F8                 BNE SCA
1F9E 60                    RTS
1F9F EA EA EA              NOP              ;for future expansion
                                            ;
1FA2 A5 E0        ADDC     LDA CURSLOC LO   ;Add SCURS to CURSLOC
1FA4 18                    CLC
1FA5 65 E5                 ADC SCURS LO
1FA7 85 E0                 STA CURSLOC LO
1FA9 A5 E1                 LDA CURSLOC HI
1FAB 65 E6                 ADC SCURS HI
1FAD 85 E1                 STA CURSLOC HI
1FAF 60                    RTS
                                            ;
1FB0 A5 E0        SBCC     LDA CURSLOC LO   ;Subtract CURSLOC from SCURS
1FB2 38                    SEC
1FB3 E5 E5                 SBC SCURS LO
1FB5 85 E0                 STA CURSLOC LO
1FB7 A5 E1                 LDA CURSLOC HI
1FB9 E5 E6                 SBC SCURS HI
1FBB 85 E1                 STA CURSLOC HI
1FBD 60                    RTS
                                            ;
1FBE 20 53 1F     LEFT     JSR TC           ;Cursor Left
1FC1 A5 E3                 LDA HL           ;Check for overflow
1FC3 F0 10                 BEQ LLE
1FC5 C6 E3                 DEC HL           ;Add 01 to CURSLOC
1FC7 A9 00                 LDA $00
1FC9 85 E6                 STA SCURS HI
1FCB A9 01                 LDA $01
1FCD 85 E5                 STA SCURS LO
1FCF 20 B0 1F              JSR SBCC
1FD2 4C E4 1F              JMP LEFY
1FD5 A9 2C        LLE      LDA $2C          ;Add 2C to Cursor
1FD7 85 E3                 STA HL
1FD9 A9 00                 LDA $00
1FDB 85 E6                 STA SCURS HI
1FDD A9 2C                 LDA $2C
1FDF 85 E5                 STA SCURS LO
1FE1 20 A2 1F              JSR ADDC
1FE4 20 5A 1F     LEFY     JSR CT
1FE7 60                    RTS
                                            ;
                                            ;
                                            ;Routines
1E40              START                     ;Start of Program
1E80              HOME                      ;Home cursor
1EFB              RIGHT                     ;Cursor Right
1F27              DOWN                      ;Cursor Down
1E95              CR                        ;Carriage Return
1EAB              LF                        ;Line Feed
1EC2              CLS                       ;Clear Screen
1EE8              DISPC                     ;Display a character
1F53              TC                        ;Temp reg. goes to display
1F5A              CT                        ;Cursor char. goes to temp reg.
1F60              SC                        ;Cursor symbol goes to disply
1F69              SCROLL                    ;Scroll display one line up
1FA2              ADDC                      ;Add SCURS to CURSLOC
1FB0              SBCC                      ;Subtract SCURS from CURSLOC
1FBE              LEFT                      ;Cursor Left
                                            ;
                                            ;End
```

Modifications to Super Cursor V1.3 for 24 by 24 Video

Zero page locations must be changed as below:

00E0	**85**	**Cursloc LO**
00E1	**D0**	**Cursloc HI**

Make the following changes to the main program:

1E88	**85**	**LDA $85**
1EAE	**17**	**CMP $19**
1EED	**17**	**CMP $17**
1F01	**17**	**CMP $17**
1F1D	**17**	**CMP $17**
1F2D	**17**	**CMP $17**
1F45	**0E**	**LDA $0E**
1F49	**02**	**LDA $20**
1F7B	**20**	**LDY $20**
1F8E	**D4**	**CMP $D4**
1F95	**20**	**LDY $20**
1FD6	**17**	**LDA $17**
1FDE	**17**	**LDA $17**

©

Memory Recall Test

V. Nasser
Birmingham, England

If a human subject is exposed to a set of random numbers or letters for a short span of time, the number of items recalled is generally about 7 ± 2 items. Theories abound as to the capacity for immediate memory. Obviously words/letters/numbers that are meaningful will be more likely to be remembered than meaningless items. Also, if numbers can be organized in a meaningful way, then the probability of accurate recall is greater. It is possible, with training, to increase one's immediate memory span by a considerable amount. Let me give a typical example. Exposed to the numbers: 162536496481 (and providing one recognizes that each pair of digits is the square of consecutive natural numbers from 4 to 9) then one only has to remember six "chunks," but will nevertheless seem to remember 12 numbers.

However, if the numbers are random, obviously one may not always be able to organize the digits in a meaningful way. My program works in the following way: it asks the subject how many numbers he or she wants to recall. When the subject enters the required items, the program will display the appropriate number of random numbers for a certain time. This exposure time incidentally, is a function of the number of numbers chosen. So that three numbers will be exposed for a much shorter time than ten numbers. After the exposure of random numbers, the screen is automatically cleared and the subject is asked to input the numbers in the sequence originally shown. The program will terminate upon the first erroneous digit input and give the correct answers.

So, having explained a bit about the psychology of immediate memory and presented a program to test your memory span, what use can this program be put to?

Under the influence of alcohol or sedative drugs the memory span drops in proportion to the amount consumed. Further, in certain conditions of brain damage it is not possible to remember more than two or three digits. Thus it has diagnostic possibilities. But more interesting, in my opinion, is the way the program encourages you to remember. If one starts at a low level, the initial successes create the automatic reinforcements necessary to increase one's memory span. This is remarkable since the numbers displayed are random and there is very little chance of organizing them in any meaningful way.

It can be used for any age group from nine years onwards. The program is both simple and absorbing. It can be adapted for any computer using BASIC, but was specifically written for the Superboard II.

```
1 FOR T=1 TO 32:PRINT:NEXT
2 PRINT "  MEMORY RECALL TEST"
3 PRINT:PRINT:PRINT
8 CLEAR
9 PRINT "  TYPE NUMBER OF NUMBERS TO BE ¬
     ¬RECALLED"
10 INPUT P
15 DIM A(P)
16 FOR I=1 TO 32:PRINT:NEXT
20 FOR N=1 TO P
25 A(N)=INT(RND(1)*10)
26 PRINT TAB(2);
30 PRINT A(N);:IF POS(1)>18 THEN PRINT:
     ¬PRINT
50 NEXT
55 FOR X=1 TO 500*N:NEXT X
56 FOR T=1 TO 32:PRINT:NEXT
60 FOR N=1 TO P
70 INPUT Y
80 IF Y<>A(N) THEN 100
85 NEXT N
90 PRINT " WELL DONE "
95 GOTO 8
100 PRINT " INCORRECT"
110 PRINT " THE CORRECT ANSWERS ARE"
120 FOR N=1 TO P
125 PRINT TAB(2);
130 PRINT A(N);:IF POS(1)>18 THEN PRINT:
     ¬PRINT
135 NEXT N
136 PRINT:PRINT:PRINT
140 GOTO 8
READY.
```

THE PET® GAZETTE

A Look At Superpet

The SuperPET looks about the same as an 8032 PET which is not surprising — it *is* an 8032 with two additional boards inside, 64K more RAM (necessary to hold the new, disk-based languages), and a new I/O system. Externally, it resembles an 8032 except for three things: 1. the logo reads "SuperPET" and, below, "SP9000"; 2. there is a set of stick-on key overlays for APL special characters; and 3. there are two, three-position switches attached to the side of the black base, below the numeric keypad.

SuperPET comes with four high-level languages, "Waterloo micro-" versions of: BASIC (40.5K), PASCAL (40.5K), FORTRAN (52.5K), and APL (64.75K). COBOL is expected soon. In addition, there is an extensive development system (nearly a high-level language) for work in 6809 machine language (an Assembler, Monitor, Linker, and Editor).

Availability

According to Commodore, a SuperPET ($1995) ordered today would arrive in about 45 days. Owners of 8032's could install a single-board upgrade for approximately $500. This upgrade is expected to be available early in 1982.

All the languages are on a single 8050 disk. This does not mean, though, that the SuperPET cannot be used with a 2040 disk system. Program 1 will redefine an 8050 as device #9. It should be linked to a power-off 2040. Turning on the 2040 leaves it as device #8. Program 2 will move the languages from an 8050 to a 2040. The value of F$ must be added to the program when transferring the final two programs (a library of utilities) on the disk, "&00,)".%80" and "7!4,)".%80". The internal quotes cannot, of course, be part of a filename — what's more, the characters are not what they seem and must be defined using CHR$ as shown in Program 3.

Program 1.

```
10 OPEN 15,8,15
20 PRINT #15,"M-W" CHR$(12)CHR$(00
   )CHR$(2)CHR$(9+32)CHR$(9+6
   4)
30 REM THE LAST TWO 9'S DEFINE DEV
   ICE #9.
40 REM FOR THE 2031 (SINGLE DRIVE)
   , USE 119 INSTEAD OF 12.
```

Program 2.

```
5 PRINT"{CLEAR}":CATALOGD0ONU9:IN
   PUT"{DOWN}FILE NAME";F$
7 SCRATCH(F$):POKE59464,0
10 OPEN1,9,8,"0:"+F$+",P,R":K=1:SO
   =59464
20 OPEN2,8,8,"0:"+F$+",P,W":B=255:
   Z=0
30 GET#1,A$:S=ST:B=B+K:IFB>254THEN
   B=Z:NB=NB+K:PRINT"{UP}{ERA
   ERASE END}BLOCK:";NB
35 IFA$=""THENA$=CHR$(Z)
40 PRINT#2,A$;:IFS=ZTHEN30
60 CLOSE1:CLOSE2:POKE 59467,0
70 PRINT"{CLEAR}TRANSFER COMPLETE{
   DOWN}":CATALOGD0
```

Program 3.

```
1 REM CREATES FILENAME FOR NON-ST
   ANDARD CHARACTERS IN SYSTE
   M LIBRARY FILENAME
3 F$=CHR$(102)+CHR$(112)+CHR$(112
   )+CHR$(108)
4 F$=F$+CHR$(105)+CHR$(98)+CHR$(4
   6)+CHR$(101)+CHR$(120)+CHR
   $(112)
5 REM ELIMINATE LINE FIVE
```

Program 3a.

```
1 REM FILENAME 7!4,)".%80
3 F$=CHR$(119)+CHR$(97)+CHR$(116)
   +CHR$(108)
4 F$=F$+CHR$(105)+CHR$(98)+CHR$(4
   6)+CHR$(101)+CHR$(120)+CHR
   $(112)
```

Manuals

The computer comes with six manuals, one for each language plus the *System Overview: Commodore SuperPET*. They are large (the BASIC manual is 221 pages) and contain numerous example programs (a second disk includes some of them). The manuals may be purchased separately from Howard W. Sams & Co., Inc., 4300 West 62nd St., P.O. Box 7092, Indianapolis, IN 46206. A minor annoyance in this otherwise carefully planned documentation is the fact that the number *1* and the lowercase *l* are identical in the Assembler handbook. In general, however, great care has obviously been taken to thoroughly explain each language. The BASIC book, for example, could easily serve as a textbook for learning this version of the language.

The Software Philosophy

Perhaps one of the first questions which comes to the mind of a microcomputerist is: what is a 40K BASIC? Personal computers contain versions of BASIC which are usually 4 to 12K large. An advanced BASIC might reach 18K. What is added when BASIC is 40.5K?

The authors of *System Overview*: "These language interpreters have been designed specifically for educational use in the teaching of computer programming. The design of the interpreters features good error diagnosis and debugging capabilities which are useful in educational and other program-development environments." There are explicit, lengthy error messages, search and replace (from the Editor), a trace facility, and *structured programming*.

Briefly, structured programming is a kind of tightening up of the rules of a language. It eliminates programming shortcuts in an effort to make programs more readable and to make languages more easily learned. Loops, for example, are supposed to be indented so they can be *seen*:

```
10 LOOP
20       X=X+1
30       Y$=VALUE$(X)
40       IF X=5000 THEN QUIT
50 ENDLOOP
```

Multiple statements per line are discouraged, spaces are required between the IF and X in IF X THEN..., the keyword VALUE must be spelled out (it replaces STR$ and VAL), LOAD "FILENAME" must have the second quote, NEXT must have a variable, dir "disk/1" replaces cAd1, and so on. This elimination of shortcuts makes programs more easily debugged, more easily read, but it also makes them larger, slows typing them in, and slows execution times. Comparing the run time of the above with the non-structured equivalent: (FOR I=1 TO 5000: Y$ = STR$(I): NEXT) takes 54 seconds, the structured version takes 119 seconds.

Some abbreviations are permitted: l for LIST, ? for PRINT. Also, the language contains a DEL function for deleting lines, RENUM for renumbering, and A for automatic line numbering.

The BASIC

As might be expected, there are significant additions and some changes to the Microsoft BASIC which is standard on other PET/CBM computers. NEW becomes CLEAR. TI becomes TIME. ! can mean REM. Structured control statements (IF, ELSE, etc.) must not be followed with anything else on a line.

A number of new functions are implemented: CURSOR (i%) sets the cursor to the position on the screen defined by the argument. DATE$ holds the current date. EPS gives the smallest number that the computer can represent. INF gives the largest. FP(x) returns the fractional part of x. IP(x) gives the integer part of x. HEX(x$) will give the hexadecimal equivalent of the decimal argument (up to a value of 32737) and HEX$(x) goes the other way. IDX(a$,b$) returns the position at which b$ first occurs within a$. IO STATUS replaces ST. MOD(x,y) provides the modulus of x for the range y. ORD(s$) returns the position of the one-character s$ in the system's set of characters. PI is pi. RPT$(s$,n) gives a string which is s$ concatenated n times. STR$(a$,s,1) is MID$.

Changes or additions to BASIC statements include: CHAIN provides program overlaying with parameter passing (USE, like DATA, contains the list to be passed). FNEND permits multiple-line function definitions. GUESS...ADMIT...ENDGUESS establish a structure similar to: 10 INPUT A$ <> "YES and A$ <> "NO" THEN PRINT "ANSWER YES OR NO": GOTO 10. (The ADMIT statement replaces an IF THEN.)

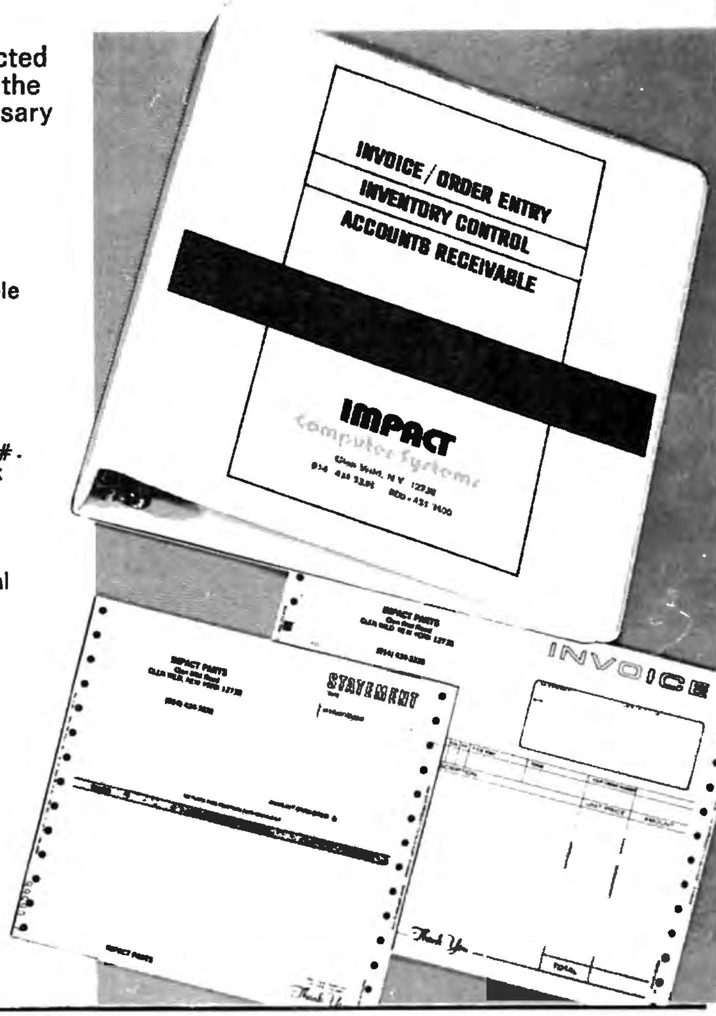
ELSEIF, ENDIF, LOOP, ENDLOOP, UNTIL, ELSE, WHILE, UNTIL, and QUIT are all statements which replace various IF THEN and FOR NEXT loop types. They are aspects of "structured programming." An ON-RESUME/IGNORE-ENDON structure permits control over some error conditions from within a program. Zero division, EOF, pressing the STOP key, under- and overflow are among the conditions which can be trapped.

This brief summary merely hints at the wealth of software and hardware to be explored in the SuperPET. When asked what impresses them most about this machine, each industry expert answers differently. Some say that the introduction of a serious version of APL is the most significant aspect of the computer. Some say it is the built-in RS-232 interface. Some mention the multiple languages or the inherent ability to speak directly to mainframe computers or the massive bank-switched RAM. All seem to agree, however, that the new PET is super. ©

INVENTORY CONTROL PLUS*...

*INVOICING — ORDER ENTRY — ACCOUNTS RECEIVABLE

for CBM 8032 and 8050



***PARTRAC:** Three separate but interconnected computer programs. All are "on line" or in the computer at the same time. It is not necessary to change disks with PARTRAC!

INVOICING SYSTEM

a. Instant stock check on any part #.
b. Instant price check on any part #.
c. Instant customer status check.
d. Writes invoices and posts to accounts receivable automatically.
e. Writes credit memos and posts to accounts receivable automatically.
f. Writes daily totals reports.

INVENTORY SYSTEM

a. Keeps track of all information related to a part #. Prices, costs, quantities on order, in stock, back ordered and vendor.
b. Prints reports
  1.) "Reorder Advice Report"
  2.) "Price List Report"
  3.) "Complete Parts Inventory Report" with total cost of inventory. (Your accountant will like that.)

ACCOUNTS RECEIVABLE

a. Posts all open account sales.
b. Prints "Accounts Receivable Report" with aging function.
c. Prints "Statements".
d. Prints "List of Accounts Names".
e. Allows partial payments.

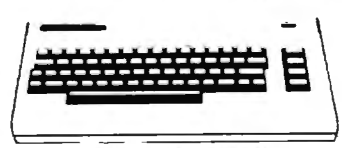
PARTRAC is fully expandable because of modular design. The basic system handles 8000 part #'s, 1000 accounts, 10,000 open invoices. This is expandable to about 20,000 part #'s, 3,000 accounts and 25,000 open invoices.

AVAILABLE AT YOUR LOCAL COMMODORE DEALER

GUARANTEED BEST PRICES!

CBM 8032 $1099

commodore



4016	$799.00
4032	$999.99
8096	$1795.00
CBM 4022 Printer	$629.00
Tally 8024	$1699.00
CBM C2N Cassette Drive	$69.00
CBM 4040 Dual Disk Drive	$999.00
CBM 8050 Dual Disk Drive	$1349.00

VIC 20 $259

Vic-TV Modual	$19.00
Vic Cassette	$69.00
Vic Disk Drive	$ Call
Vic 6 Pack program	$44.00

CBM Software

WordPro 3 Plus	$199.00
WordPro 4 Plus	$299.00
Commodore Tax Package	$399.00
Visicalc	$149.00
EBS Accts. Rec./Inventory Interactive Syst.	$595.00
BPI General Ledger	$329.00
OZZ Information System	$329.00
Dow Jones Portfolio	$129.00
Pascal	$239.00
Legal Time Accounting	$449.00
World Craft 80	$289.00
Word Check	$180.00
Create-A-Base	$180.00
Power	$89.00
Socket-2-Me	$20.00
Jinsam	Call

Printers

Epson MX-70	
Epson MX-80	
Epson MX-80 FT	Call for Prices
Diablo 630	
Nec 5530	$2495.00
TEC 1500 Starwriter 25cps	$1495.00
TEC 1500 Starwriter 45cps	$1995.00

IMPACT
Computer Systems

IN N.Y. CALL
914-434-3338
(800) 431-3400
GLEN WILD, N. Y. 12738

FREE FREIGHT--SAME DAY SHIPPING

VISA MasterCard

TO ORDER: Phone orders invited (800 is for order desk only). Send check or money order and receive free shipping. New York residents add 7% sales tax. Add 3% for Visa or M.C. Equipment is subject to price change and availability without notice. Please call between 9 A.M. and 5 P.M.

SUPERMON:
A Primary Tool For Machine Language Programming

Here is the legendary Supermon — a version for Upgrade (3.0 or "New ROM") and 4.0 PETs, all keyboards, all memory sizes, 40 or 80 column screens. You need not know how to program in machine language (ML) to enter this program — or to use it. In fact, exploring with Supermon, you will find that the mysterious world of your computer's *own* language becomes gradually understandable. You will find yourself learning ML.

Many ML programmers with PET/CBM machines feel that Supermon is the essential tool for developing programs of short to medium length. All Upgrade and 4.0 machines have a "resident" monitor; a program within the computer's ROM which allows you to type SYS 1024 and see the registers, load and save and run ML programs, or see a memory dump (a list of numbers from the computer's memory cells.) But to program or analyze ML easily, disassembler, assembler, hunt, and single-step functions are all practical necessities. Supermon provides these and more.

Even if you've never assembled a single instruction and don't know NOP from ROL, this article will lead you step-by-step through the entry and SAVE of Supermon. And even if you do not plan to explore ML right now, you might consider putting this program into your library. If you ever decide to work a bit in ML, Supermon will prove invaluable.

How To Enter Supermon

1. Type in the BASIC program (Program 1). It is the same for all versions. Then save it normally by typing SAVE "CONTROL". This program will be used later to automatically find your memory size, transfer Supermon to the top, and report to you the SYS address you use to activate it.

2. Now the hard part: type SYS 1024 which enters you into the machine language monitor. You will see something like the following:

Figure 1.

```
B*
     PC   IRQ  SR AC XR YR SP
.;  0401 E455 32 04 5E 00 EE
```

Then type: M 0600 0648 and you will see something similar to (the numbers will be different, but we are going to type over them which, after hitting RETURN on each line, will enter the new numbers into the computer's memory.):

Figure 2.

```
.M 0600 0648
.:  0600 28 58 FF FF 00 0B 06 AD
.:  0608 FF FC 00 21 06 03 AD A9
.:  0610 CB 85 1F A9 0C 85 20 A5
.:  0618 34 85 21 A5 35 85 22 A0
.:  0620 00 93 06 06 D0 16 20 38
.:  0628 06 F0 11 85 23 20 38 06
.:  0630 18 65 34 AA A5 23 65 35
.:  0638 20 43 06 8A 20 43 06 20
.:  0640 50 06 90 DB 60 EA EA A5
.:  0648 1F D0 02 C6 20 C6 1F B1
.
```

We have divided Supermon into 21 blocks with 80 hexadecimal numbers per block to make typing easier. There is a final, shorter block with 64 numbers. Type right over the numbers on the screen so that line 0600 looks like it does in Program 2. Then hit RETURN and cursor over to the A5 on line 0608. (Set a TAB to this position if your keyboard has a TAB key.) Then type over the numbers in this line and so on. When you have finished typing your RETURN on line 0648, type in: M 0650 0698 and the next block will appear for you to type over. Continue this way until you finish entering the new version of line 0CC8 at the end. (Hope that no lightning or fuses blow.)

3. If you have Upgrade ROMs, you will need to correct the lines listed in Program 3 at this point. To change line 06D0, simply type M 06D0 06D0 and it will appear so that you can type over it and RETURN as in step 2.

4. Now Supermon is in your memory and you must SAVE it. Hit RETURN so that you are on a new line and type: S "SUPERMON", 01,0600, 0CCC (to SAVE to tape) or type: S "0:SUPERMON",08,0600,0CCC (to SAVE to disk drive 0).

5. Finally, you will want to use the Checksum program to see if you made any errors during the marathon. You probably did, but to make it as painless as possible, the Checksum program will flash through your Supermon and let you know which blocks need to be corrected. So, type in Program 4 (or if you have Upgrade ROMs, use the first three lines from Program 5). SAVE Checksum just in case. Then LOAD "SUPERMON" (an ordinary LOAD as with a BASIC program will slide it in starting at

address 1536, above the end of Checksum). Then RUN. Incorrect blocks will be announced. When you know where the errors are, type SYS 1024 and then M XXXX XXXX for the starting and ending addresses of the bad block. Check the numbers against Program 2 (or Program 3) and in all corrections. If, despite everything, you cannot find an error within a block, make sure that the corresponding number within the DATA statement of the Checksum program is correct. Then SAVE the good version "SUPERMON1" as in step 4. "SUPERMON1" as in step 4.

6. Your reward is near. LOAD "CONTROL" and then LOAD SUPERMON1. Then type RUN and hold your breath. If all goes well, you should see:

Figure 3.

```
 SUPERMON4!

     DISSASSEMBLER  BY WOZNIAK/BAUM
       SINGLE STEP
 BY JIM RUSSO
 MOST OTHER STUFF ,BY BILL SEILER

 TIDIED & WRAPPED BY JIM BUTTERFIELD


 LINK TO MONITOR -- SYS 31283

 SAVE WITH MLM:
.S "SUPERMON",01,7A33,8000
 READY.
```

And you should be able to use all the commands listed in the Supermon Summary. If some, or all, of the commands fail to function, check the last, short block of code to see if there are any errors.

After Supermon is relocated to the top of your memory, use a ML SAVE to save it in its final form. Instructions are on screen after RUN.

SUPERMON SUMMARY

```
COMMODORE MONITOR INSTRUCTIONS:
G GO RUN
L LOAD FROM TAPE OR DISK
M MEMORY DISPLAY
R REGISTER DISPLAY
S SAVE TO TAPE OR DISK
X EXIT TO BASIC
SUPERMON ADDITIONAL INSTRUCTIONS:
A SIMPLE ASSEMBLER
D DISASSEMBLER
F FILL MEMORY
H HUNT MEMORY
I SINGLE INSTRUCTION
P PRINTING DISASSEMBLER
T TRANSFER MEMORY
SUPERMON WILL LOAD ITSELF INTO THE
TOP OF MEMORY .. WHEREVER THAT HAPPENS
TO BE ON YOUR MACHINE.
   YOU MAY THEN SAVE THE MACHINE CODE
FOR FASTER LOADING IN THE FUTURE.
BE SURE TO NOTE THE SYS COMMAND WHICH
LINKS SUPERMON TO THE COMMODORE
MONITOR.


 SIMPLE ASSEMBLER
.A 2000 LDA #$12
.A 2002 STA $8000,X
.A 2005 (RETURN)
.
     IN THE ABOVE EXAMPLE THE USER
STARTED ASSEMBLY AT 1000 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
PROMPTS WITH THE NEXT ADDRESS.  TO EXIT
THE ASSEMBLER TYPE A RETURN AFTER THE
THE ADDRESS PROMPT.  SYNTAX IS THE SAME
AS THE DISASSEMBLER OUTPUT.

  DISASSEMBLER
.D 2000
(SCREEN CLEARS)
.,   2000 A9 12        LDA #$12
.,   2002 9D 00 80     STA $8000,X
.,   2005 AA           TAX
.,   2006 AA           TAX
(FULL PAGE OF INSTRUCTIONS)
     DISASSEMBLES 22 INSTRUCTIONS
STARTING AT 1000 HEX.  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.  SUPERMON
WILL THEN DISASSEMBLE THAT PAGE AGAIN.

 PRINTING DISASSEMBLER
.P 2000,2040
2000 A9 12         LDA #$12
2002 9D 00 80      STA $8000,XY.
2005 AA            TAX
    ....
203F A2 00         LDX #$00
TO ENGAGE PRINTER, SET UP BEFOREHAND:
     OPEN 4,4:CMD4
ON 4.0, ACCESS THE MONITOR VIA A CALL
SYS 54386 (*NOT* A BREAK) COMMAND

 SINGLE STEP
.I
     ALLOWS A MACHINE LANGUAGE PROGRAM
TO BE RUN STEP BY STEP.
CALL REGISTER DISPLAY WITH .R AND SET
```

THE PC ADDRESS TO THE DESIRED FIRST INSTRUCTION FOR SINGLE STEPPING.
THE .I WILL CAUSE A SINGLE STEP TO EXECUTE AND WILL DISASSEMBLE THE NEXT.
CONTROLS:
  < FOR SINGLE STEP;
  RVS FOR SLOW STEP;
  SPACE FOR FAST STEPPING;
  STOP TO RETURN TO MONITOR.
[ON BUSINESS KEYBOARDS--
      USE 8,←,6 AND STOP].

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. A 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.

LOAD
.L
LOAD ANY PROGRAM FROM CASSETTE #1.
.L "RAM TEST"
LOAD FROM CASSETTE #1 THE PROGRAM NAMED RAM TEST.
.L "RAM TEST",08
LOAD FROM DISK (DEVICE 8) THE PROGRAM NAMED RAM TEST.
THIS COMMAND LEAVES BASIC POINTERS UNCHANGED.

MEMORY DISPLAY
.M 0000 0080
.: 0000 00 01 02 03 04 05 06 07
.: 0008 08 09 0A 0B 0C 0D 0E 0F

DISPLAY MEMORY FROM 0000 HEX TO 0080 HEX. THE BYTES FOLLOWING THE .: CAN BE ALTERED BY TYPING OVER THEM THEN TYPING A RETURN.

REGISTER DISPLAY
.R
     PC   IRQ SR AC XR YR SP
.; 0000 E62E 01 02 03 04 05
DISPLAYS THE REGISTER VALUES SAVED WHEN SUPERMON WAS ENTERED. THE VALUES MAY BE CHANGED WITH THE EDIT FOLLOWED BY A RETURN.
USE THIS INSTRUCTION TO SET UP THE PC VALUE BEFORE SINGLE STEPPING WITH .I

SAVE
.S "PROGRAM NAME",01,0800,0C80
SAVE TO CASSETTE #1 MEMORY FROM 0800 HEX UP TO BUT NOT INCLUDING 0C80 HEX AND NAME IT PROGRAM NAME.
.S "0:PROGRAM NAME",08,1200,1F50
SAVE TO DISK DRIVE #0 MEMORY FROM 1200 HEX UP TO BUT NOT INCLUDING 1F50 HEX AND NAME IT PROGRAM NAME.

TRANSFER MEMORY
.T 1000 1100 5000
TRANSFER MEMORY IN THE RANGE 1000 HEX TO 1100 HEX AND START STORING IT AT ADDRESS 5000 HEX.

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.

Program 1.

```
100 PRINT"{CLEAR}{02 DOWN}{REV} SUP
    ERMON!!"
110 PRINT"{DOWN}      DISSASSEMBLER ~
    {REV}D{OFF} BY WOZNIAK/BAU
    M
120 PRINT"        SINGLE STEP {REV}I
    {OFF} BY JIM RUSSO
130 PRINT"MOST OTHER STUFF {REV},HA
    LT{OFF} BY BILL SEILER
150 PRINT"{DOWN}TIDIED & WRAPPED BY
     JIM BUTTERFIELD"
170 L=PEEK(52)+PEEK(53)*256:SYS1536
    :M=PEEK(33):N=PEEK(34)
180 POKE52,M:POKE53,N:POKE48,M:POKE
    49,N:N=M+N*256
210 PRINT"{02 DOWN}LINK TO MONITOR ~
    -- SYS";N
220 PRINT:PRINT"SAVE WITH MLM:"
230 PRINT".S ";CHR$(34);"SUPERMON";
    CHR$(34);",01";:X=N/4096:G
    OSUB250
```

```
240 X=L/4096:GOSUB250:END
250 PRINT",";:FORJ=1TO4:X%=X:X=(X-X
    %)*16:IFX%>9THENX%=X%+7
260 PRINTCHR$(X%+48);:NEXTJ:RETURN
```

SUPERMON 4.0 Program 2.

```
.
.:  0600 A9 CB 85 1F A9 0C 85 20
.:  0608 A5 34 85 21 A5 35 85 22
.:  0610 A0 00 20 38 06 D0 16 20
.:  0618 38 06 F0 11 85 23 20 38
.:  0620 06 18 65 34 AA A5 23 65
.:  0628 35 20 43 06 8A 20 43 06
.:  0630 20 50 06 90 DB 60 EA EA
.:  0638 A5 1F D0 02 C6 20 C6 1F
.:  0640 B1 1F 60 48 A5 21 D0 02
.:  0648 C6 22 C6 21 68 91 21 60
.
.:  0650 A9 80 C5 1F A9 06 E5 20
.:  0658 60 AA AA AA AA AA AA AA
.:  0660 AA AA AA AA AA AA AA AA
.:  0668 AA AA AA AA AA AA AA AA
.:  0670 AA AA AA AA AA AA AA AA
.:  0678 AA AA AA AA AA AA AA AA
.:  0680 AD FE FF 00 85 34 AD FF
.:  0688 FF 00 85 35 AD FC FF 00
.:  0690 8D FA 03 AD FD FF 00 8D
.:  0698 FB 03 00 00 A2 08 DD DE
.
.:  06A0 FF 00 D0 0E 86 B4 8A 0A
.:  06A8 AA BD E9 FF 00 48 BD E8
.:  06B0 FF 00 48 60 CA 10 EA 4C
.:  06B8 9A FA 00 A2 02 2C A2 00
.:  06C0 00 B4 FB D0 08 B4 FC D0
.:  06C8 02 E6 DE D6 FC D6 FB 60
.:  06D0 20 98 D7 C9 20 F0 F9 60
.:  06D8 A9 00 00 8D 00 00 01 20
.:  06E0 79 FA 00 20 6B D7 20 57
.:  06E8 D7 90 09 60 20 98 D7 20
.
.:  06F0 54 D7 B0 DE AE 06 02 9A
.:  06F8 4C A4 D7 20 31 D5 CA D0
.:  0700 FA 60 E6 FD D0 02 E6 FE
.:  0708 60 A2 02 B5 FA 48 BD 0A
.:  0710 02 95 FA 68 9D 0A 02 CA
.:  0718 D0 F1 60 AD 0B 02 AC 0C
.:  0720 02 4C CE FA 00 A5 FD A4
.:  0728 FE 38 E5 FB 8D 1B 02 98
.:  0730 E5 FC A8 0D 1B 02 60 20
.:  0738 81 FA 00 20 44 D7 20 92
.
.:  0740 FA 00 20 AF FA 00 20 92
.:  0748 FA 00 20 CA FA 00 20 44
.:  0750 D7 90 15 A6 DE D0 65 20
.:  0758 C1 FA 00 90 60 A1 FB 81
.:  0760 FD 20 A8 FA 00 20 39 D5
.:  0768 D0 EB 20 C1 FA 00 18 AD
.:  0770 1B 02 65 FD 85 FD 98 65
.:  0778 FE 85 FE 20 AF FA 00 A6
.:  0780 DE D0 3D A1 FB 81 FD 20
.:  0788 C1 FA 00 B0 34 20 65 FA
.
.:  0790 00 20 68 FA 00 4C 1B FB
.:  0798 00 20 81 FA 00 20 44 D7
.:  07A0 20 92 FA 00 20 44 D7 20
.:  07A8 98 D7 20 63 D7 90 14 85
.:  07B0 B5 A6 DE D0 11 20 CA FA
.:  07B8 00 90 0C A5 B5 81 FB 20
.:  07C0 39 D5 D0 EE 4C 9A FA 00
.:  07C8 4C BA D4 20 81 FA 00 20
.:  07D0 44 D7 20 92 FA 00 20 44
.:  07D8 D7 20 98 D7 A2 00 00 20
.
.:  07E0 98 D7 C9 27 D0 14 20 98
.:  07E8 D7 9D 10 02 E8 20 CF FF
.:  07F0 C9 0D F0 22 E0 20 D0 F1
.:  07F8 F0 1C 8E 00 00 01 20 6B
.:  0800 D7 90 C6 9D 10 02 E8 20
.:  0808 CF FF C9 0D F0 09 20 63
.:  0810 D7 90 B6 E0 20 D0 EC 86
.:  0818 B4 20 34 D5 A2 00 00 A0
.:  0820 00 00 B1 FB DD 10 02 D0
.:  0828 0C C8 E8 E4 B4 D0 F3 20
.
.:  0830 17 D7 20 31 D5 20 39 D5
.:  0838 A6 DE D0 92 20 CA FA 00
.:  0840 B0 DD 4C BA D4 20 81 FA
.:  0848 00 8D 0D 02 A5 FC 8D 0E
.:  0850 02 A9 04 A2 00 00 8D 09
.:  0858 02 8E 0A 02 A9 93 20 D2
.:  0860 FF A9 16 85 B5 20 06 FC
.:  0868 00 20 64 FC 00 85 FB 84
.:  0870 FC C6 B5 D0 F2 A9 91 20
.:  0878 D2 FF 4C BA D4 A0 2C 20
.
.:  0880 79 D5 20 17 D7 20 31 D5
.:  0888 A2 00 00 A1 FB 20 74 FC
.:  0890 00 48 20 BB FC 00 68 20
.:  0898 D3 FC 00 A2 06 E0 03 D0
.:  08A0 13 AC 1C 02 F0 0E A5 FF
.:  08A8 C9 E8 B1 FB B0 1C 20 5C
.:  08B0 FC 00 88 D0 F2 06 FF 90
.:  08B8 0E BD 51 FF 00 20 45 FD
.:  08C0 00 BD 57 FF 00 F0 03 20
.:  08C8 45 FD 00 CA D0 D4 60 20
.
.:  08D0 68 FC 00 AA E8 D0 01 C8
.:  08D8 98 20 5C FC 00 8A 86 B4
.:  08E0 20 22 D7 A6 B4 60 AD 1C
.:  08E8 02 38 A4 FC AA 10 01 88
.:  08F0 65 FB 90 01 C8 60 A8 4A
.:  08F8 90 0B 4A B0 17 C9 22 F0
```

```
.: 0900 13 29 07 09 80 4A AA BD
.: 0908 00 FF 00 B0 04 4A 4A 4A
.: 0910 4A 29 0F D0 04 A0 80 A9
.: 0918 00 00 AA BD 44 FF 00 85
.
.: 0920 FF 29 03 8D 1C 02 98 29
.: 0928 8F AA 98 A0 03 E0 8A F0
.: 0930 0B 4A 90 08 4A 4A 09 20
.: 0938 88 D0 FA C8 88 D0 F2 60
.: 0940 B1 FB 20 5C FC 00 A2 01
.: 0948 20 A1 FA 00 CC 1C 02 C8
.: 0950 90 F0 A2 03 CC 09 02 90
.: 0958 F0 60 A8 B9 5E FF 00 8D
.: 0960 0B 02 B9 9E FF 00 8D 0C
.: 0968 02 A9 00 00 A0 05 0E 0C
.
.: 0970 02 2E 0B 02 2A 88 D0 F6
.: 0978 69 3F 20 D2 FF CA D0 EA
.: 0980 4C 31 D5 20 81 FA 00 20
.: 0988 44 D7 20 92 FA 00 20 44
.: 0990 D7 A9 04 A2 00 00 8D 09
.: 0998 02 8E 0A 02 20 34 D5 20
.: 09A0 0B FC 00 20 64 FC 00 85
.: 09A8 FB 84 FC 20 35 F3 F0 05
.: 09B0 20 CA FA 00 B0 E9 4C BA
.: 09B8 D4 20 81 FA 00 A9 03 85
.
.: 09C0 B5 20 98 D7 20 0B D5 D0
.: 09C8 F8 AD 0D 02 85 FB AD 0E
.: 09D0 02 85 FC 4C E7 FB 00 CD
.: 09D8 0A 02 F0 03 20 D2 FF 60
.: 09E0 A9 03 A2 24 8D 09 02 8E
.: 09E8 0A 02 20 34 D5 78 AD FA
.: 09F0 FF 00 85 90 AD FB FF 00
.: 09F8 85 91 A9 A0 8D 4E E8 CE
.: 0A00 13 E8 A9 2E 8D 48 E8 A9
.: 0A08 00 00 8D 49 E8 AE 06 02
.
.: 0A10 9A 4C 55 D6 20 C0 FC 68
.: 0A18 8D 05 02 68 8D 04 02 68
.: 0A20 8D 03 02 68 8D 02 02 68
.: 0A28 8D 01 02 68 8D 00 00 02
.: 0A30 BA 8E 06 02 58 20 34 D5
.: 0A38 20 23 D5 85 B5 A0 00 00
.: 0A40 20 FE D4 20 31 D5 AD 00
.: 0A48 00 02 85 FC AD 01 02 85
.: 0A50 FB 20 17 D7 20 0E FC 00
.: 0A58 20 35 F3 C9 F7 F0 F9 20
.
.: 0A60 35 F3 D0 03 4C BA D4 C9
.: 0A68 FF F0 F4 4C 5B FD 00 20
.: 0A70 81 FA 00 20 44 D7 8E 11
.: 0A78 02 A2 03 20 79 FA 00 48
.: 0A80 CA D0 F9 A2 03 68 38 E9
.: 0A88 3F A0 05 4A 6E 11 02 6E
.: 0A90 10 02 88 D0 F6 CA D0 ED
.: 0A98 A2 02 20 CF FF C9 0D F0
.: 0AA0 1E C9 20 F0 F5 20 F7 FE
.: 0AA8 00 B0 0F 20 78 D7 A4 FB
.
.: 0AB0 84 FC 85 FB A9 30 9D 10
.: 0AB8 02 E8 9D 10 02 E8 D0 DB
.: 0AC0 8E 0B 02 A2 00 00 86 DE
.: 0AC8 F0 04 E6 DE F0 7B A2 00
.: 0AD0 00 86 B5 A5 DE 20 74 FC
.: 0AD8 00 A6 FF 8E 0C 02 AA BC
.: 0AE0 5E FF 00 BD 9E FF 00 20
.: 0AE8 E0 FE 00 D0 E2 A2 06 E0
.: 0AF0 03 D0 1A AC 1C 02 F0 15
.: 0AF8 A5 FF C9 E8 A9 30 B0 21
.
.: 0B00 20 E6 FE 00 D0 CA 20 E8
.: 0B08 FE 00 D0 C5 88 D0 EB 06
.: 0B10 FF 90 0B BC 57 FF 00 BD
.: 0B18 51 FF 00 20 E0 FE 00 D0
.: 0B20 B3 CA D0 D0 F0 0A 20 DF
.: 0B28 FE 00 D0 A9 20 DF FE 00
.: 0B30 D0 A4 AD 0B 02 C5 B5 D0
.: 0B38 9D 20 44 D7 AC 1C 02 F0
.: 0B40 2F AD 0C 02 C9 9D D0 20
.: 0B48 20 CA FA 00 90 0B 98 D0
.
.: 0B50 05 AE 1B 02 10 0B 4C 9A
.: 0B58 FA 00 C8 D0 FA AE 1B 02
.: 0B60 10 F5 CA CA 8A AC 1C 02
.: 0B68 D0 03 B9 FC 00 00 91 FB
.: 0B70 88 D0 F8 A5 DE 91 FB 20
.: 0B78 64 FC 00 85 FB 84 FC A0
.: 0B80 41 20 79 D5 20 17 D7 20
.: 0B88 31 D5 4C D8 FD 00 A8 20
.: 0B90 E6 FE 00 D0 11 98 F0 0E
.: 0B98 86 B4 A6 B5 DD 10 02 08
.
.: 0BA0 E8 86 B5 A6 B4 28 60 C9
.: 0BA8 30 90 03 C9 47 60 38 60
.: 0BB0 40 02 45 03 D0 08 40 09
.: 0BB8 30 22 45 33 D0 08 40 09
.: 0BC0 40 02 45 33 D0 08 40 09
.: 0BC8 40 02 45 B3 D0 08 40 09
.: 0BD0 00 00 22 44 33 D0 8C 44
.: 0BD8 00 00 11 22 44 33 D0 8C
.: 0BE0 44 9A 10 22 44 33 D0 08
.: 0BE8 40 09 10 22 44 33 D0 08
.
.: 0BF0 40 09 62 13 78 A9 00 00
.: 0BF8 21 81 82 00 00 00 00 59
.: 0C00 4D 91 92 86 4A 85 9D 2C
.: 0C08 29 2C 23 28 24 59 00 00
.: 0C10 58 24 24 00 00 1C 8A 1C
.: 0C18 23 5D 8B 1B A1 9D 8A 1D
.: 0C20 23 9D 8B 1D A1 00 00 29
.: 0C28 19 AE 69 A8 19 23 24 53
```

```
.:   0C30 1B 23 24 53 19 A1 00 00
.:   0C38 1A 5B 5B A5 69 24 24 AE
.
.:   0C40 AE A8 AD 29 00 00 7C 00
.:   0C48 00 15 9C 6D 9C A5 69 29
.:   0C50 53 84 13 34 11 A5 69 23
.:   0C58 A0 D8 62 5A 48 26 62 94
.:   0C60 88 54 44 C8 54 68 44 E8
.:   0C68 94 00 00 B4 08 84 74 B4
.:   0C70 28 6E 74 F4 CC 4A 72 F2
.:   0C78 A4 8A 00 00 AA A2 A2 74
.:   0C80 74 74 72 44 68 B2 32 B2
.:   0C88 00 00 22 00 00 1A 1A 26
.
.:   0C90 26 72 72 88 C8 C4 CA 26
.:   0C98 48 44 44 A2 C8 54 46 48
.:   0CA0 44 50 2C 41 49 4E 00 00
.:   0CA8 DB FA 00 30 FB 00 5E FB
.:   0CB0 00 D1 FB 00 F8 FC 00 28
.:   0CB8 FD 00 D4 FD 00 4D FD 00
.:   0CC0 B9 D4 7F FD 00 4A FA 00
.:   0CC8 33 FA 00 AA AA AA AA AA
```

SUPERMON 3.0 Progrm 3.

```
.:   06D0 20 EB E7 C9 20 F0 F9 60
:
.:   06E0 79 FA 00 20 BE E7 20 AA
.:   06E8 E7 90 09 60 20 EB E7 20
.:   06F0 A7 E7 B0 DE AE 06 02 9A
.:   06F8 4C F7 E7 20 CD FD CA D0
:
.:   0738 81 FA 00 20 97 E7 20 92
:
.:   0748 FA 00 20 CA FA 00 20 97
.:   0750 E7 90 15 A6 DE D0 65 20
:
.:   0760 FD 20 A8 FA 00 20 D5 FD
:
.:   0798 00 20 81 FA 00 20 97 E7
.:   07A0 20 92 FA 00 20 97 E7 20
.:   07A8 EB E7 20 B6 E7 90 14 85
:
.:   07C0 D5 FD D0 EE 4C 9A FA 00
.:   07C8 4C 56 FD 20 81 FA 00 20
.:   07D0 97 E7 20 92 FA 00 20 97
.:   07D8 E7 20 EB E7 A2 00 00 20
.:   07E0 EB E7 C9 27 D0 14 20 EB
.:   07E8 E7 9D 10 02 E8 20 CF FF
:
.:   07F8 F0 1C 8E 00 00 01 20 BE
.:   0800 E7 90 C6 9D 10 02 E8 20
.:   0808 CF FF C9 0D F0 09 20 B6
.:   0810 E7 90 B6 E0 20 D0 EC 86
.:   0818 B4 20 D0 FD A2 00 00 A0
:
.:   0830 6A E7 20 CD FD 20 D5 FD
:
.:   0840 B0 DD 4C 56 FD 20 81 FA
:
.:   0878 D2 FF 4C 56 FD A0 2C 20
.:   0880 15 FE 20 6A E7 20 CD FD
:
.:   08E0 20 75 E7 A6 B4 60 AD 1C
:
.:   0980 4C CD FD 20 81 FA 00 20
.:   0988 97 E7 20 92 FA 00 20 97
.:   0990 E7 A9 04 A2 00 00 8D 09
.:   0998 02 8E 0A 02 2( D0 FD 20
:
.:   09A8 FB 84 FC 20 01 F3 F0 05
.:   09B0 20 CA FA 00 B0 E9 4C 56
.:   09B8 FD 20 81 FA 00 A9 03 85
.:   09C0 B5 20 EB E7 20 A7 FD D0
:
.:   09E8 0A 02 20 D0 FD 78 AD FA
.
.:   0A10 9A 4C F1 FE 20 7B FC 68
:
.:   0A30 BA 8E 06 02 58 20 D0 FD
.:   0A38 20 BF FD 85 B5 A0 00 00
.:   0A40 20 9A FD 20 CD FD AD 00
:
.:   0A50 FB 20 6A E7 20 0E FC 00
.:   0A58 20 01 F3 C9 F7 F0 F9 20
.:   0A60 01 F3 D0 03 4C 56 FD C9
:
.:   0A70 81 FA 00 20 97 E7 8E 11
:
.:   0AA8 00 B0 0F 20 CB E7 A4 FB
:
.:   0B38 9D 20 97 E7 AC 1C 02 F0
:
.:   0B80 41 20 15 FE 20 6A E7 20
.:   0B88 CD FD 4C D8 FD 00 A8 20
:
.:   0CC0 55 FD 7F FD 00 4A FA 00
```

SUPERMON Program 4.

```
100 REM SUPERMON 4 CHECKSUM
110 DATA7331,12186,10071,10387,1082
    9,9175,10314,9823,9715,871
    4,8852
120 DATA8850,9748,7754,10247,10423,
    10948,10075,6093,5492,7805
    :S=1536
130 FORB=1TO21:READX:FORI=STOS+79:N
    =PEEK(I):Y=Y+N
140 NEXTI:IFY<>XTHENPRINT"ERROR IN
    BLOCK #"B:GOTO160
150 PRINT"BLOCK #"B" IS CORRECT"
160 S=I:Y=0:NEXTB:PRINT"CHECK THE F
    INAL, SHORT BLOCK BY HAND"
```

SUPERMON Program 5.

```
100 REM SUPERMON 3 CHECKSUM
110 DATA7331,12186,10467,10880,1112
    4,10005,10906,10196,9951,8
    813
120 DATA8852,9329,10239,8457,10334,
    10423,11047,10311,6093,549
    2,7805:S=1536
```

PET To PET Communication Over The User Port

John Winn
Department of Chemistry
University of California at Berkeley

If you (or you and a friend) have access to two PETs, you may have wanted to connect the two together and transfer data from one to the other. The built-in IEEE bus is not suitable, since each PET is a bus controller and the rules allow only one controller on the bus. You could buy any of a number of attachments for serial, parallel or modem input/output, but the simplest method is to interconnect the PET's through the built-in parallel user port. Here's how it's done, using fairly simple BASIC and twelve wires.

First, what hardware is required? The user port connections are on the *bottom* row of the PC output edge connector. Looking at the rear of the PET, these are labelled A through N with keying slots sawed between A and B and between L and M. A and N are ground connections. C through L are the eight parallel data lines. Each will correspond, in effect, to one of the eight bits in a memory byte. Connection B is called "CA1"; it will be used to signal the presence of data to be read by the receiving PET. Connection M, called "CB2," will control (signal) CA1 on the other PET. (How this is done will be clearer later on.) To connect the two ports together, use two edge connector plugs, wiring A to A, N to N, C through L to C through L, but wire B on one connector to M on the other and vice versa (i.e. CA1 on one to CB2 on the other). The total length of the cable should not be more than about 20 feet. (Longer distances would require external "line drivers" to keep the signal from degrading.)

To control these dozen wires, various PEEKs and POKEs are used. One PET will transmit, and the other will receive at any one time, although each can do both. To send one byte, the transmitter will first activate the eight data lines. Then it will signal the receiver that the byte is set to be read. The receiver will read the byte and signal back to the transmitter that it has done so and is ready for the next.

Suppose we want to send one character from one PET to the other. Program 1 gives the program for the transmitter and Program 2, for the receiver. Line 20 in each program shows how the direction of data transfer is controlled. Line 40 of the transmitter program shows how one byte (ASC(A$)) is placed on the data lines. Meanwhile, the receiver is stuck on its line 40, waiting for bit two of memory location 59469 to be a one instead of a zero. This transition will signal the receiver that it can read the data lines. The signal is sent (from CB2 of the transmitter to CA1 of the receiver) by lines 60 and 70 of the transmitter program. Line 60 forces the three most significant bits of memory location 59468 to be ones. (The other bits are unchanged.) Line 70 forces the third most significant bit back to zero, forcing the first two to be ones and leaving the low order five bits (which are used for other things) as they were. This sequence turns CB2 on, then off.

stuck on line 90 waiting for the receiver to signal back that it has read the data. The receiver signals with lines 70 and 80. It then prints the received character on its screen and goes after another byte. The transmitter will get the signal and ask for another character to send, and the process will repeat.

Most applications will involve the transfer of more than just one character. Transmitting a whole string of many characters or a floating point number requires more elaborate programs, but they will be based on these simple versions. To send a string, the length of the string must be sent first, and then the string can be sent character by character. To send a floating point number, the simplest technique seems to be to use one BASIC variable at a known location in memory as an intermediary buffer, as is done in the programs described below.

You *Could* WAIT

Two other concerns arise. The first is the initial synchronization of the data transfer. This is perhaps best taken care of by a one byte "preamble" sent at the beginning of the program just to clean out any unsuspected data or transfer signals. The second concern is the ability to interrupt the transmission gracefully should something go wrong. (Along this same line, it is worth pointing out that line 90 of Program 1 and line 40 of Program 2 could be written using WAIT statements. But, since WAITs are not interruptable, except by pulling the plug, this is a bit dangerous.) The easiest way to interrupt a program without stopping it directly is to use the SHIFT key in the way described below.

Programs 3 and 4 give more elaborate pro-

grams which send a string of arbitrary length and arbitrary number of random floating point numbers. They both use the SHIFT key to signal an interrupt. (With Original ROM's, location 516 is zero if the SHIFT key is up, and one if it is down. With Upgrade ROM's, it's location 152.) The transmitter sends a preamble — one "%" — to guarantee synchronization. The character is arbitrary, but it should be as unique (or obscure) as possible.

The floating point buffer variable, called QQ in each program, *must* be the first defined variable of the program. This is so its location in memory can be found easily. At the beginning of variable data storage, one finds two bytes for the two character name of the first variable followed by five bytes representing the floating point number itself. Variables start at memory location 256*PEEK(43) +PEEK(42) in Upgrade ROM's (256*PEEK(125) +PEEK(124) in Original ROM's); hence, variable SQ in each program gives the location, two bytes along from the start, for QQ's five data bytes.

Data are transmitted (or received) in subroutines 1000 and 2000. Starting at 1000 is the subroutine for transmitting or receiving the five bytes of QQ. Transmitting or receiving only one byte (variable D in the program) is done by the subroutine starting at line 2000. Note that this subroutine is called by the first one.

Interruption requires that you hold down the SHIFT key until the program can branch to line 3000. Both the transmitter and the receiver have to be interrupted separately, but either can be interrupted first.

These programs illustrate the main techniques needed for more useful and interesting applications. For many games ("Battleship" comes to mind), the transfer rate of the BASIC code is fast enough, around 10 bytes per second or so.

ML For Fast Transfer

For much greater speed, machine language code is needed. Program 5 is a machine language version of the BASIC code in Programs 3 and 4, implemented in a slightly different way. Line 10 sets up a variable, D%, for receiving single bytes. It must be the first variable defined in the program, and the PEEKs must be changed to 125 and 124 for Original ROM's. The POKE 2,3 statement sets part of the linkage for the USR function. Line 20 POKEs the machine language code into the second cassette buffer. Line 30 puts the address of the low-order byte of D% into this code and sets D% back to zero. (Note: POKEX, PEEK(Y) does not work on Original ROM's. That's why line 30 is written the way it is.) The DATA statements contain the machine language for Upgrade ROM's. For Original ROM's, change the two occurrences (lines 1035 and 1057) of 94 both to 176. They locate the floating point accumulator used by USR.

To set the program into the transmit mode (line 100), POKE 1,91 first to complete the USR linkage for transmission. Next, send a one byte preamble ("%" is used here again) to insure synchronization. To send individual bytes (line 200), POKE them into location 832 and call SYS826. To transmit a floating point number (line 300), pass the number (or variable) as the argument of USR. Since USR has to be set equal to something, it can safely be set equal to the variable being passed or to any other variable which you want to equate to the variable being passed.

Of course, when one program is set up to transmit, the other must be set up to receive. First, (line 400), POKE 1,139 to complete the USR linkage for reception. Next, look for the preamble and warn yourself (line 440) if it was not received as expected. The FOR-NEXT loop in 420–430 should never go past I=2. To receive individual bytes (line 500), call SYS873, and find the byte in the variable D%. To receive a floating point number (line 600), equate the variable you wish to input to USR. The argument to USR is not important here, nor is it disturbed if a variable is used.

In most programs, lines 100–120 and 400–440 would best be made subroutines which could be called to switch the program from one mode to the other at will. The main disadvantage of this program is that it cannot be easily interrupted. Data synchronization between the two PETs must be exact or one will finish first, leaving the other hung up. One or more direct SYS826 or SYS873 commands from the un-hung PET will, eventually, clear the other. (Which SYS you use will depend on the state– transmitter or receiver– of the hungup PET.

Transmission Rate

The data rate is quite good. Sending 2000 numbers in a command FOR I=1 TO 2000: X=USR(I): NEXT takes about 8.6 seconds. That works out to (2000x6)/8.6 = 1400 bytes per second. In this test, the receiver just read the numbers, but did nothing with them. When the receiver stuck the numbers into an array, the time went up to 12.5 seconds.

Finally, if you want to locate the machine language somewhere other than 826 to 917 (or $033A to $0395), the only six numbers in DATA which change are the thirty-ninth (64), fortieth (3), forty-second (58), forty-third (3), eighty-fifth (69), and eighty-sixth (3). These, in pairs, are low and high order absolute address bytes (i.e. 64+3*256 =832). They will have to be changed along with the various POKE locations in BASIC (and the numbers POKEd into locations 1 and 2) if the program is relocated. [*It is suggested that 4.0 users move the routine to avoid DOS usage of the bottom of this buffer. — Ed.*]

Program 1.

```
 10 REM **** SIMPLE TRANSMITTER
 20 POKE59459,255 :REM SET DATA
                LINES FOR OUTPUT
 30 INPUT"ENTER A CHARACTER";A$
 40 POKE59457,ASC(A$) :REM OUTPUT
                       CHARACTER
 50 REM NEXT 2 LINES SIGNAL THE RECEIVER
                          TO READ DATA
 60 POKE59468,PEEK(59468) OR 224
 70 POKE59468,PEEK(59468) AND 31 OR 192
 80 REM WAIT FOR RECEIVER TO SIGNAL BACK
 90 IF(PEEK(59469) AND 2)<>2 THEN 90
 100 GOTO30
READY.
```


Program 3.

```
 10 REM **** ELABORATE TRANSMITTER (UPGRADE ROM VERSION)
 20 QQ=0 :REM QQ MUST BE FIRST VARIABLE
 30 SQ=PEEK(42)+256*PEEK(43)+2 :REM ADDRESS OF FIRST QQ DATA BYTE
 40 SH=152 :REM ADDRESS OF 'SHIFT' KEY FLAG
 50 POKE59459,255 :REM SET DATA LINES FOR OUTPUT
 60 REM SEND SYNCHRONIZATION PREAMBLE
 70 D=ASC("%") : GOSUB2000
 80 PRINT"READY TO TRANSMIT":PRINT"USE 'SHIFT' KEY TO INTERRUPT"
 90 INPUT"ENTER A STRING";A$
 100 QQ=LEN(A$) : GOSUB1000 :REM TRANSMIT LEN(A$)
 110 FORI=1TOQQ
 120 D=ASC(MID$(A$,I,1))
 130 GOSUB2000 :REM SEND STRING 1 CHARACTER AT A TIME
 140 NEXT
 150 INPUT"HOW MANY RANDOM NUMBERS";N
 160 QQ=N : GOSUB1000 :REM TRANSMIT N
 170 FORI=1TON
 180 QQ=RND(1)
 190 GOSUB1000 :REM TRANSMIT EACH RANDOM NUMBER
 200 NEXT
 999 END
 1000 REM SUBROUTINE FOR FLOATING POINT TRANSMISSION
 1010 FORIJ=0TO4
 1020 D=PEEK(SQ+IJ)
 1030 GOSUB2000 :REM SEND QQ BYTE BY BYTE
 1040 NEXT
 1050 RETURN
 2000 REM SUBROUTINE FOR BYTE TRANSMISSION
 2010 POKE59459,D :REM OUTPUT BYTE
 2020 REM SIGNAL 'DATA READY'
 2030 POKE59468,PEEK(59468) OR 224
 2040 POKE59468,PEEK(59468) AND 31 OR 192
 2050 REM WAIT FOR RECEPTION AND ALLOW INTERRUPT
 2060 IF((PEEK(59469) AND 2)<>2) AND (PEEK(SH)<>1) THEN2060
 2070 IFPEEK(SH)=1THEN3000 :REM INTERRUPT
 2080 RETURN
 3000 PRINT"INTERRUPTED"
 3010 GOTO999 :REM END IF INTERRUPTED
READY.
```

SM-KIT

for Commodore Computers

A Programming Productivity Tool

ONLY $40

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.

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

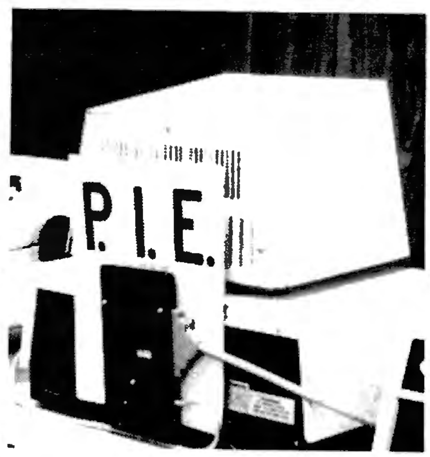
252 Bethlehem Pike
Colmar, PA 18915

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)

Program 2.

```
 10 REM **** SIMPLE RECEIVER
 20 POKE59459,0 :REM SET DATA
                 LINES FOR INPUT
 30 REM WAIT FOR DATA TO BE SENT
 40 IF(PEEK(59469) AND 2)<>2 THEN40
 50 D=PEEK(59457) :REM READ DATA
 60 REM NEXT 2 LINES SIGNAL
       THE TRANSMITTER "DATA READ"
 70 POKE59468,PEEK(59468) OR 224
 80 POKE59468,PEEK(59468) AND 31 OR 192
 90 PRINT CHR$(D) :REM PRINT THE
                  RECEIVED CHARACTER
 100 GOTO30
READY.
```


Program 4.

```
 10 REM **** ELABORATE RECEIVER (UPGRADE ROM VERSION)
 20 QQ=0 :REM QQ MUST BE FIRST VARIABLE
 30 SQ=PEEK(42)+256*PEEK(43)+2 :REM ADDRESS OF FIRST QQ DATA BYTE
 40 SH=152 :REM ADDRESS OF 'SHIFT' KEY FLAG
 50 POKE59459,0 :REM SET DATA LINES FOR INPUT
 60 REM LOOK FOR SYNCHRONIZATION PREAMBLE
 70 FORI=1TO3 : GOSUB2000 : IFD=ASC("%")THEN80
 72 NEXT
 74 PRINT"EXPECTED PREAMBLE NOT RECEIVED" : END
 80 PRINT"READY TO RECEIVE":PRINT"USE 'SHIFT' KEY TO INTERRUPT"
 90 GOSUB1000 :REM READ LENGTH OF TRANSMITTED STRING
 100 A$="" : FORI=1TOQQ : GOSUB2000 :REM INPUT ONE BYTE
 110 A$=A$+CHR$(D) :REM BUILD UP STRING
 120 NEXT
 130 PRINT"RECEIVED ";A$
 140 REM READ NUMBER OF RANDOM INPUTS TO EXPECT
 150 GOSUB1000 : N=QQ
 160 FORI=1TON
 170 GOSUB1000 : PRINT QQ :REM READ RANDOM NUMBERS
 180 NEXT
 999 END
 1000 REM SUBROUTINE FOR FLOATING POINT RECEPTION
 1010 FORIJ=0TO4
 1020 GOSUB2000 :REM READ QQ BYTE BY BYTE
 1030 POKESQ+IJ,D :REM BUILD NEW QQ
 1040 NEXT
 1050 RETURN
 2000 REM SUBROUTINE FOR BYTE RECEPTION
 2010 REM WAIT FOR DATA TO BE SENT AND ALLOW INTERRUPTION
 2020 IF((PEEK(59469) AND 2)<>2) AND (PEEK(SH)<>1) THEN2020
 2030 IFPEEK(SH)=1THEN3000 :REM INTERRUPT
 2040 D=PEEK(59457) :REM READ DATA BYTE
 2050 REM SIGNAL 'DATA RECEIVED'
 2060 POKE59468,PEEK(59468) OR 224
 2070 POKE59468,PEEK(59468) AND 31 OR 192
 2080 RETURN
 3000 PRINT"INTERRUPTED"
 3010 GOTO999 :REM END IF INTERRUPTED
READY.
```

Program 5.

```
 10 D%=256*PEEK(43)+PEEK(42)+3:POKE2,3
 20 FORI=826TO917:READJ:POKEI,J:NEXT
 30 I=PEEK(D%):POKE889,I:I=PEEK(D%-1):
    POKE890,I:D%=0
 100 REM ****** TRANSMIT
 110 POKE1,91:REM SET USR FOR TRANSMISSION
 120 POKE832,ASC("%"):SYS826:REM SEND
     PREAMBLE
 200 REM TRANSMIT A BYTE ("A")
 210 POKE832,ASC("A"):SYS826
 300 REM TRANSMIT THE NUMBER 1.23
 310 X=1.23:X=USR(X)
 400 REM ****** RECEIVE
 410 POKE1,139:REM SET USR FOR RECEPTION
 420 FORI=1TO3:SYS873:IFD%=ASC("%")THEN500
 430 NEXT:REM LOOK FOR PREAMBLE
 440 PRINT"PREAMBLE NOT RECEIVED":STOP
 500 REM RECEIVE A BYTE
 510 SYS873:A$=CHR$(D%):PRINTA$
 600 REM RECEIVE A NUMBER
 610 X=USR(0):PRINTX
 1000 REM DATA & CORRESPONDING MNEMONICS
 1010 REM
 1020 DATA 169,255    : TBYTE  LDA #$FF
 1021 DATA 141,67,232:         STA $E843
 1022 DATA 169,0      :        LDA #$**
 1023 DATA 141,65,232:         STA $E841
 1024 DATA 173,76,232:         LDA $E84C
 1025 DATA 9,224      :        ORA #$E0
 1026 DATA 141,76,232:         STA $E84C
 1027 DATA 41,31      :        AND #$1F
 1028 DATA 9,192      :        ORA #$C0
 1029 DATA 141,76,232:         STA $E84C
 1030 DATA 173,77,232: TWAIT   LDA $E84D
 1031 DATA 41,2       :        AND #$02
 1032 DATA 240,249    :        BEQ TWAIT
 1033 DATA 96         :        RTS
 1034 DATA 162,5      :        LDX #$05
 1035 DATA 181,94:<--: TFLPT   LDA $5E,X
 1036 DATA 141,64,3   :        STA $0340
 1037 DATA 32,58,3    :        JSR TBYTE
 1038 DATA 202        :        DEX
 1039 DATA 16,245     :        BPL TFLPT
 1040 DATA 96         :        RTS
 1041 DATA 169,0      : RBYTE  LDA #$00
 1042 DATA 141,67,232:         STA $E843
 1043 DATA 173,77,232: RWAIT   LDA $E84D
 1044 DATA 41,2       :        AND #$02
 1045 DATA 240,249    :        BEQ RWAIT
 1046 DATA 174,65,232:         LDY $E84C
 1047 DATA 142,0,0    :        STY $****
 1048 DATA 173,76,232:         LDA $E84C
 1049 DATA 9,224      :        ORA #$E0
 1050 DATA 141,76,232:         STA $E84C
 1051 DATA 41,31      :        AND #$1F
 1052 DATA 9,192      :        ORA #$C0
 1053 DATA 141,76,232:         STA $E84C
 1054 DATA 96         :        RTS
 1055 DATA 162,5      :        LDX #$05
 1056 DATA 32,69,3    : RFLPT  JSR RBYTE
 1057 DATA 148,94:<--:         STY $5E,X
 1058 DATA 202        :        DEX
 1059 DATA 16,248     :        BPL RFLPT
 1060 DATA 96         :        RTS
READY.
```

Replacing The INPUT# Command

Jerry E. Dunmire
San Jose, CA

At last you have your PET and now you can keep track of all those magazine articles, recipes, addresses or whatever else you promised your spouse! At least that's how I felt, and I immediately sat down to write the programs.

If you have tried to write a program that uses the INPUT command, then you know the problems I encountered. The INPUT command will not accept commas, quotes, or colons and using the GET command to construct a string is very slow. Since a proper bibliography of magazine articles must contain quote marks, I was stuck with the GET command. There had to be a better way.

There is! Nothing says that all programs must be written in BASIC. I could write a machine language routine to replace the INPUT# command. The new routine would accept all characters. Replacing the INPUT# command would also solve the same problems I encountered when reading from the tape or disk.

There are three items that we need to know in order to write a new version of the INPUT# command: how strings are stored, where the string is located, and how to input characters. The *PET/CBM Personal Computer Guide* by Adam Osborne and Carroll S. Donahue provided the information on string storage. Raymond Diedrichs explained how to input from a file in his article "Pet File I/O in Machine Language" **COMPUTE!** #11.

Strings are stored at the top of the available memory. As each string is entered, it is added to the bottom of the list. In order to identify a particular string we must know where it begins and how long it is. The PET uses one byte to represent the length of the string, and two bytes to identify the address where the string begins. The particular format that identifies a string depends on whether the string is an element of an array or a simple variable.

A simple variable has the form shown in Figure 1. If the string is an element of an array, it would be identified as shown in Figure 2. We can disregard the information in the header of an array.

This is only part of the information we need to locate a string in memory. The location of the pointer to the string is still unknown. Must our routine search for the name of the particular string we wish to input? Well, it could, but there is an easier way. Locations $44 and $45 point to the last variable referenced. If that last variable were the string we wish to input, then these locations will point to the length of the string, and the next two locations will be the address where the string is stored. Figure 3 shows the relationship between locations $44, $45, variables, and strings.

Reading characters from a file is even easier than dealing with strings. If a file has been opened by a BASIC statement, the subroutine at $FFC6 will set the file up so we can read from it. Then the subroutine at $FFCF will input a character from that file. When we have all the characters we want, the default I/O devices should be restored.

Armed with this knowledge, I wrote two routines. The two routines are named READString and INPUTLine. They are located in the second cassette buffer. Both use locations $44 and $45 to locate the variable, so the last variable you reference before calling these routines must be a string.

READS inputs a fixed number of characters from file #1. The number of characters is determined by the length of the string referenced by locations $44, 45. As the characters are read in, they replace the characters that are already in the string. This routine will cause strange problems if locations $44, $45 point to a string with zero length. To prevent this occurrence, I use the following commands to call READS:

```
10 IF LEN(A$) THEN SYS(826)
```

If A$ has a zero length, READS will never be called. As you can see, the starting address of READS is 826 ($033A).

The version of READS shown in Program 1 reads one additional character after it has filled the referenced string. The file has a carriage return at the end of each string. To remove this extra character input, place NOP's ($EA) in locations $0361 through $0363.

INPUTL also uses file #1. A carriage return must mark the end of a string just like the INPUT# command. INPUTL will accept any character other than a carriage return. Up to 80 characters can be input. If more than 80 characters are input, the ST variable will be set to a value of −1.

INPUTL works more like INPUT# than READS does. As the individual characters of a string are input, they are placed in an input buffer. Only after the string has been terminated with a carriage return is it transferred to the string storage area and assigned to the variable pointed to by $44, $45. The string is copied from the input buffer to

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

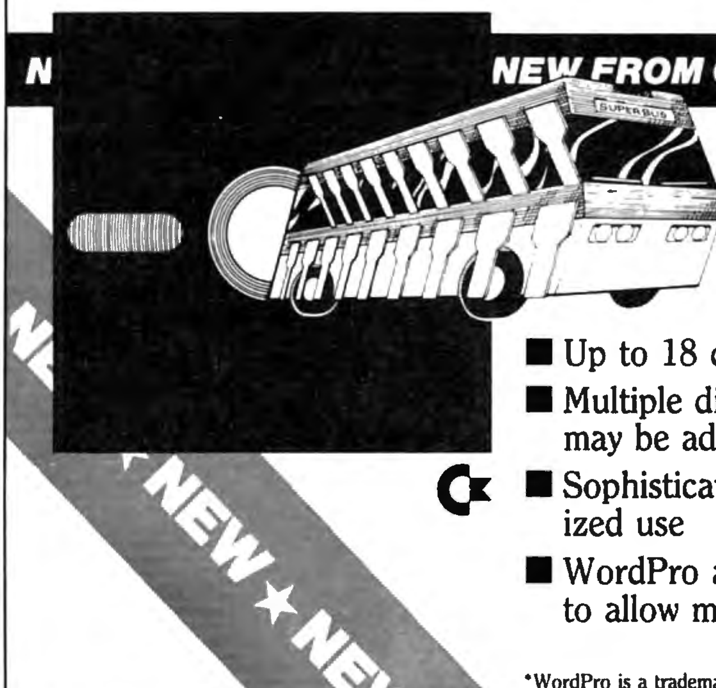
DESIGNED FOR COMMODORE 8000 SERIES COMPUTERS AND DISK DRIVES

- 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



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 just below the string storage area. Then the pointer to the beginning of the string storage area is adjusted to account for the new string.

I use the following line to call INPUTL, but you can use any function that leaves locations $44, $45 pointing to the variable you wish to input.

10 A$=" " : SYS(872)

As you can see, the starting address for INPUTL is 872. As with READS, if the last variable you referenced were not a string then the results are almost unpredictable and certainly bad.

You can change the file number used by these programs to suit your needs. Simply POKE the number of a file you have opened into location 827 for READLINE and 873 for INPUTSTRING.

INPUTL and READS will work with BASIC 3.0 or BASIC 4.0. If you need to use them with BASIC 1.0 then you will have to adjust all of the references to memory locations less than $0400 (1024 decimal).

INPUT# is still the fastest way to input a string. However, both INPUTL and READS are at least three to four times faster than using GET# commands. If you are short on memory, using the GET# command will be exceedingly slow since it will cause the garbage collection routine to execute more often than any of the other methods.

Program 1.

```
800 FOR ADRES=826TO949:READ DATTA:P
    OKEADRES,DATTA
805 NEXT
826 DATA 162, 1, 32, 198, 255, 160
832 DATA 0, 177, 68, 133, 96, 200
838 DATA 177, 68, 133, 94, 200, 177
844 DATA 68, 133, 95, 169, 0, 133
850 DATA 97, 32, 207, 255, 164, 97
856 DATA 145, 94, 200, 132, 97, 198
862 DATA 96, 208, 242, 32, 207, 255
868 DATA 32, 204, 255, 96, 162, 1
874 DATA 32, 198, 255, 169, 0, 133
880 DATA 5, 32, 207, 255, 201, 13
886 DATA 240, 15, 166, 5, 232, 224
892 DATA 81, 240, 47, 157, 0, 2
898 DATA 134, 5, 76, 113, 3, 166
904 DATA 5, 160, 0, 198, 48, 165
910 DATA 48, 201, 255, 208, 2, 198
916 DATA 49, 189, 0, 2, 145, 48
922 DATA 202, 208, 238, 165, 5, 145
928 DATA 68, 165, 48, 200, 145, 68
934 DATA 165, 49, 200, 145, 68, 76
940 DATA 178, 3, 169, 255, 133, 150
946 DATA 32, 204, 255, 96
```

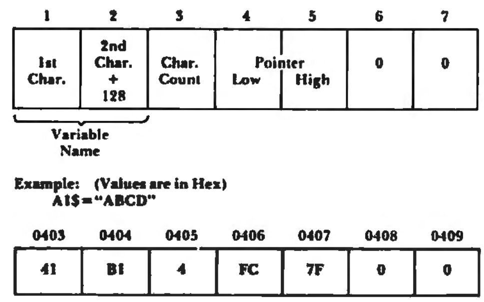


Figure 1. Simple String Variable Storage

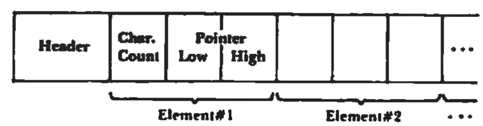


Figure 2. Array String Storage

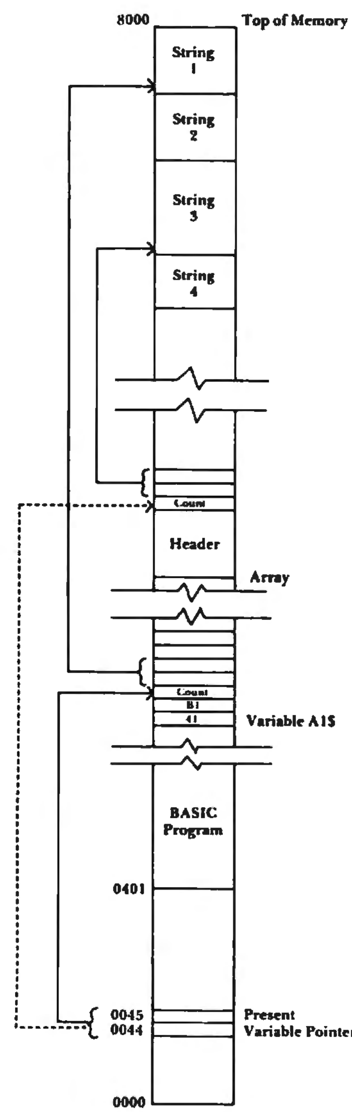


Figure 3. (Upgrade or 4.0 BASIC) Memory Map

Typing Foreign Language Text With The Commodore Printer

Zoltan Szepesi
Pittsburgh, PA

Most languages, unlike English, use different kinds of marks or accents above some of the vowels or even above or below some consonants. The French has the "accent aigu" (´), "accent grave" (`) and the "accent circonflexe" (^) placed in many words above vowels e,a,o,u and they have also the "cédille" placed below C (as ç) in some words. English typewriters and printers generally do not have the facility for printing these orthographic signs. However, with the CBM series 2022 and 2023 printers one can create special characters, thereby printing any of the wanted letters.

We could create the complete special character (letter + accent) for each vowel. However, for 4 vowels and 3 accents we would need 4x3 = 12 special characters. It is simpler to program only the 3 accents and, any time one needs the accent on the vowel, one goes to a subroutine to print the accent in the proper position. After the accent is printed, one has a carriage return without line feed, and the standard characters are printed after a line was typed.

At first, I made a program according to this plan. However, as each accent needs a full printer head scan, the printing time was slowed down very much if the number of accents in a line were great. Therefore, I modified the program so that the accents are printed after the full line has been printed, and any number of accents of one kind is printed in one printer head movement. This improved the speed to a practically acceptable level.

This paper and program will not handle the printing of special symbols below the letters as the cédille in French. This problem is the same as printing descenders on letters g,j,q,p,y. According to the same principles as described above for the accents, one can create these special characters. However, the printing can be done only with the tractor feed printer (2022 series). A paper and program on this problem will be published elsewhere.

In writing this program, I started with the "TYPEWRITER 1.5" program of Warren D. Swan, published in THE PAPER (pages 11-15, Vol. II, Issue 10, January 1980), modified for the new CBM/PET (ROM 3) and the new printer ROM (4). It is a very simple, but powerful, mini-word-processor.

Listing 1 is the accent printing program. There, first we have to design the special character strings A$(I), where I = 0 TO 2. They are defined in statement 240, using the DATA in statements 150 to 170 for the three above mentioned French accents. (See instructions in the Printer User manual or in Swan's article). Second, one has to decide which keys to sacrifice for calling the subroutine for the special accents. We used the "and" key (&) for the accent aigu (´) as specified in statement 410, the "shift and" for accent grave (`) — statement 420 — and the "shift apostrophe" key for accent circonflexe (^) — statement 430. For the printing of the accents in the proper place, the strings S$(I) are created, one for each accent type. For tape recording and reprinting the text from tape the string T$ is created.

Swan gives the instructions for how to use the original program. I will tell you shortly what to do and how to do it in the modified program.

For Input one can choose:

1. The keyboard (device #0)
2. The tape recorder (device #1 or #2)

For Output:

1. Tapes 1 or 2.
2. Screen (device #3)
3. Printer (device #4)

For the tape files you can give a file title. If you do not need a title just press two apostrophes ("").

When using the DEL command in or after an enhanced text, the following rule has to be applied:

Within the enhanced text the correction can be made the normal way, if the SHIFT- (key was not yet pressed.

If you want to go back to the enhanced characters after an exit from them, use the DEL until you delete the exit character. Here press the enhanced command (SHIFT-BACKSLASH) and continue with DEL. Do not again use the SHIFT-BACKSLASH when you type the corrected text, but use the SHIFT-(key when you want to continue with standard characters.

Boldface characters can be printed also on the full line by printing the same text on the same line

KEY	ASCII	Statement No.	Function
CLR	147	320	Sets the paging mode to the printer.
HOME	19	330	Sends a cursor-home character to the output device only.
DEL	20	340	Deletes the last character.
SHIFT BACKSLASH	220	350	Enhanced print
SHIFT (	168	360	Unenhanced print for stopping enhanced
INST	148	370	Deletes entire line
BACKSLASH	92	380	TAB for next 8 spaces
SHIFT #	163	400	Prints programmable character defined previously.
RETURN	13	310	Brings the printer to the print subroutine.
SHIFT RETURN	141		
&	38	410	Accent aigu
SHIFT &	166	420	Accent grave
SHIFT APOSTROPHE	167	430	Accent circonflexe
CURSOR LEFT	157	390	Program goes to the special command mode, where one can ask: 1. A programmable character 2. Change the # of lines/inch 3. End the program.

Table 1. List of operations for the TYPEWRITER ACCENT program.

two times. For achieving this, do the following:

After the line you want to print bold face, do not press RETURN, but use SHIFT-RETURN. For the next line press SHIFT-BACKSLASH and RETURN.

If the language you want to print has other accents than the ones given in this program, just construct their forms according to the instructions of the printer and substitute the resulting six numbers into the data statements 150-170. E.g.: the German text needs only the "Umlaut," which could be printed by the following data:

0,0,64,0,64,0

Since more accents are not used in the German, the other 2 accents can be deleted in the program. Since there will be just a single A$ and S$, statements 240,270,280,500,510,2090,2510 could be modified accordingly and statements 160,170,420,430,1110,1120,1220,1230 could be deleted.

In the Hungarian, beside the accent grave and the Umlaut, one needs an accent similar to the quotation mark. The following data would define this:

0,32,64,32,64,0

In several other languages one uses a wave-shaped accent. The previous Hungarian accent could be acceptable for this accent too.

Copyright registration of this program is being requested. You can use this program for your personal use, or you can have it on tape by sending $3.00 to my address: 2611 Saybrook Drive, Pittsburgh, PA 15235.

```
10 REM TYPE ACCENT  PROGRAM BY Z.SZEPESI
     ¬(COPYRIGHT REGISTRATION APPLIED)
20 REM MODIFIED FROM TYPEWRITER1.5 BY ¬
     ¬W.D.SWAN(THE PAPER VOL.II.ISSUE ¬
     ¬10)
30 REM            INITIALIZATION
35 REM          "---------------"
40 POKE 59468,14:OU=4:IN=1:Q=205:
     ¬CO=59467:T0=59466:R=59464
50 K=0:H=2:PRINT"WHAT IS THE INPUT ¬
     ¬DEVICE # (0 TO 2)?";:GOSUB3000
60 K=1:H=4:C=D:IFC=1ORC=2THENINPUT"FILE ¬
     ¬TITLE:";TL$:GOTO80
70 OPENIN,C
80 PRINT"WHAT IS THE OUTPUT DEVICE # (1 ¬
     ¬TO 4)?";:GOSUB3000
90 IFC=DGOTO80
100 IFD=1ORD=2THENINPUT"FILE TITLE";TL$
105 IFC$=""GOTO100
110 S=-(D<4):IFC=1ORC=2THENOPENIN,C,0,
     ¬TL$
120 IFD=1ORD=2THENOPENOU,D,S,TL$:GOTO140
130 OPENOU,D,S
140 PRINT"ĥ":OPEN5,OU,5:DIM A$(2),S$(2)
150 DATA 0,0,0,32,64,0:REM ACCENT AIGU
160 DATA 0,64,32,0,0,0:REM ACC. GRAVE
170 DATA 0,32,64,32,0,0:REM ACC. CIRC.
240 FORI=0TO2:A$(I)="":FORJ=1TO6:READA:
     ¬A$(I)=A$(I)+CHR$(A):NEXTJ:NEXTI
250 REM          MAIN PROGRAM LOOP
255 REM        "-----------------"
```

```
260 OPEN7,4,7:PRINT#7:CLOSE7:OPEN6,4,6:
    ¬M=6
270 POKEQ,0:PRINT"h↓↓↓↓↓$<";:P=1:L$="":
    ¬G$="":T$="":FORI=0TO2:S$(I)="":
    ¬NEXT
280 AA=0:A0=0:A1=0:A2=0:E=0
290 GET#IN,C$:IF64ANDSTGOTO7000
300 IFC$=""GOTO290
310 IFC$=CHR$(13)ORC$=CHR$(141)GOTO1000:
    ¬REM TO PRINT LINE
320 IFC$="ĥ"THENPRINT#OU,"ĥ":GOTO290
330 IFC$="h"THENPRINT#OU,"h":GOTO290
340 IFC$=CHR$(20)THENGOSUB2000:GOTO290
350 IFC$="\"THENC$=CHR$(1):E=1:IFD=1ORD=
    ¬2THENT$=T$+"\":GOTO290
360 IFC$="⌊"THENC$=CHR$(129):E=0:
    ¬IFD=1ORD=2THENT$=T$+"⌊":GOTO290
370 IFC$=CHR$(148)THENFORK=1TOLEN(L$):
    ¬GOSUB2000:NEXT:GOTO270
380 IFC$="\"GOTO2500
390 IFC$="<"GOTO4000
400 IF C$="£"THENC$=CHR$(254)
410 IFC$="&"THENT$=T$+C$:GOSUB1100:AA=1:
    ¬A0=1:GOTO290
420 IFC$="&"THENT$=T$+C$:GOSUB1110:AA=1:
    ¬A1=1:GOTO290
430 IFC$="!"THENT$=T$+C$:GOSUB1120:AA=1:
    ¬A2=1:GOTO290
500 FORI=0TO2:S$(I)=S$(I)+" ":NEXTI:
    ¬IF E=0GOTO520
510 FORJ=0TO2:S$(J)=S$(J)+" ":NEXTJ
520 C=ASC(C$)AND127
530 L$=L$+C$:T$=T$+C$
540 IFC>31ORC$=">"THENP=P+1:IFE=1THENP=P
    ¬+1
550 IFP=72THENGOSUB2600
560 POKEQ,1:G$=G$+C$:IFE=1THENG$=G$+" "
570 PRINTC$;:IFE=1THENPRINT" ";
580 POKEQ,0:PRINT"$<";:GOTO290
998 REM          PRINT THE LINE
999 REM         "---------------"
1000 IFL$="rA"THENL$=M$:G$=H$
1010 PRINT:IFD=4THENPRINT#OU,L$;CHR$(141
     ¬);
1020 IFD=1ORD=2THENPRINT#OU,T$;CHR$(141)
     ¬;
1030 PRINT"ĥ";:FORK=1TOLEN(G$):POKEQ,1:
     ¬PRINTMID$(G$,K,1);:NEXT:M$=L$:
     ¬H$=G$
1040 POKEQ,0:IF AA=1 THENGOSUB1200
1060 PRINT#OU,CHR$(13);:GOTO270
1100 PRINT#5,A$(0):S$(0)=LEFT$(S$(0),
     ¬P-2)+CHR$(254)
1105 RETURN
1110 PRINT#5,A$(1):S$(1)=LEFT$(S$(1),
     ¬P-2)+CHR$(254)
1115 RETURN
1120 PRINT#5,A$(2):S$(2)=LEFT$(S$(2),
     ¬P-2)+CHR$(254)
1125 RETURN
1200 IFD<>4THENRETURN
1210 IFA0=1THENPRINT#5,A$(0):PRINT#OU,
     ¬S$(0);CHR$(141);
1220 IFA1=1THENPRINT#5,A$(1):PRINT#OU,
     ¬S$(1);CHR$(141);
1230 IFA2=1THENPRINT#5,A$(2):PRINT#OU,
     ¬S$(2);CHR$(141);
1240 RETURN
1998 REM          DELETE A CHAR.
1999 REM         "---------------"
2000 IFLEN(L$)=0ORLEN(T$)=0THENRETURN
2010 PRINTCHR$(20);:IFE=1THENPRINTCHR$(2
     ¬0);
2020 F$=RIGHT$(G$,1):G$=MID$(G$,1,
     ¬LEN(G$)-1):IFE=1THENG$=LEFT$(G$,
     ¬LEN(G$)-1)
2030 F1$=RIGHT$(T$,1):T$=LEFT$(T$,
     ¬LEN(T$)-1)
2040 IF F1$="&"ORF1$="&"ORF1$="!"THENT$=
     ¬LEFT$(T$,LEN(T$)-1)
2050 IFF$<>"<"GOTO2080
2060 O$=RIGHT$(L$,1):L$=MID$(L$,1,
     ¬LEN(L$)-1):IFO$<>"<"ORF$<>"<"GOTO2
     ¬050
2070 RETURN
2080 L$=MID$(L$,1,LEN(L$)-1):P=P+((ASC(F
     ¬$)AND127)>31)+(F$=">"):IFE=1THENP=
     ¬P-1
2090 FORI=0TO2:S$(I)=LEFT$(S$(I),P-1):
     ¬NEXT
2100 IFD=1ORD=2GOTO2120
2110 RETURN
2120 IF F1$="&"ORF1$="&"ORF1$="!"THENT$=
     ¬LEFT$(T$,P-1)
2130 IFE=1THEN T$=LEFT$(T$,LEN(T$)+1)
2140 RETURN
2498 REM          TAB TO NEXT STOP
2499 REM         "-----------------"
2500 T=8-(PAND7):P=P+T:FORK=1TOT:
     ¬L$=L$+" ":G$=G$+" ":T$=T$+" ":
     ¬PRINT" ";
2510 FORI=0TO2:S$(I)=S$(I)+" ":NEXTI
2520 NEXTK:PRINT"$<";
2530 IFP>=72THENGOSUB2600
2540 GOTO290
2598 REM          END OF LINE BEEP
2599 REM         "-----------------"
2600 POKER,0:POKECO,16:POKET0,15:
     ¬POKER,150:FORK=1TO2E2:NEXT
2610 POKER,0:POKET0,0:POKECO,0::RETURN
2998 REM          GET A DEVICE
2999 REM         "------------"
3000 GETC$:IFC$=""GOTO3000
3010 D=ASC(C$)-48:IFD<KORD>HGOTO3000
3020 PRINTD:RETURN
3998 REM          EXTRA COMMANDS
3999 REM         "----------------"
4000 PRINT"ENTER COMMAND:"
4010 PRINT"↓1. DEFINE A PROGRAMMABLE ¬
     ¬CHARACTER."
4020 PRINT"↓2. SET LINES/INCH.":
     ¬PRINT"↓3. END PROGRAM"
4030 GET#IN,F$:IF64ANDSTGOTO7000
4040 IFF$<"1"ORF$>"3"GOTO4030
4050 IFD=1ORD=2THENL$=L$+"<"+F$:
     ¬G$=G$+"<"
4060 IF F$="3"GOTO7000
4070 IF F$="2"GOTO6000
4998 REM          DEFINE CHARACTER
4999 REM         "-----------------"
```

```
5000 PG$="":PRINT"ĥENTER 6 NUMBERS TO ¬
     ¬DEFINE THE CHARACTER:
5010 PRINT"(ONE AT A TIME FOLLOWED BY ¬
     ¬RETURN)
5020 IFD=4THENCLOSE5:OPEN5,OU,5
5030 FORK=1TO6:PRINTK;:INPUT#IN,F$:
     ¬IF64ANDSTGOTO7000
5040 C=VAL(F$):PRINTC:IFD=1ORD=2THENL$=L
     ¬$+F$+CHR$(13)
5050 PG$=PG$+CHR$(C):NEXT:IFD=4THENPRINT
     ¬#5,PG$
5060 PRINT"ĥ";:FORK=1TOLEN(H$):POKEQ,1:
     ¬PRINTMID$(H$,K,1);:NEXT
5070 POKEQ,0:PRINT"ĥ↓↓↓↓":FORK=1TOLEN(G$
     ¬):POKEQ,1:PRINTMID$(G$,K,1);:NEXT
5080 POKEQ,0:PRINT"$<";:GOTO290
5998 REM          SET LINES/INCH
5999 REM        "----------------"
6000 PRINT"ĥNUMBER OF LINES PER INCH? ";
6010 IFD=4THENCLOSE6:OPEN6,OU,6
6020 INPUT#IN,F$:IF64ANDSTGOTO7000
6030 M=VAL(F$):PRINTM:IFD=1ORD=2THENL$=L
     ¬$+F$+CHR$(13)
6040 IFD=4THENPRINT#6,CHR$(144/M)
6050 GOTO5060
6998 REM          END OF PROGRAM
6999 REM        "----------------"
7000 PRINT:IFD=1ORD=2THENCLOSEOU
7010 END:IFD>2GOTO5060
7020 PRINT"?CAN'T CONTINUE ERROR":END:
     ¬RUN
```


Three Reviews:

Superchip, Spacemaker, Sort

Harvey B. Herman
Associate Editor

The Petmaster Superchip

Some of us may have envied the tricks one can play with the new 80 column PETs using BASIC 4.0. For example, one can define a window which is seemingly immune from scrolling. SUPERCHIP, firmware from our English cousins, is intended to provide some of these screen handling functions and additional goodies also. It is available for all the current PET ROMs and does not conflict with the TOOLKIT.

The first feature I made use of (and liked) is called single key BASIC. That is, G stands for GO, N stands for NEXT, R stands for RETURN, etc. The full word appears, as if by magic, when a control key is pressed simultaneously with a letter. Another function that caught my fancy is called escape. This allows you to toggle back and forth between quote and direct modes of cursor control. If you ever get stuck in the wrong mode you know how useful that could be. I also made frequent use of the hold function which suspends execution until RETURN is pressed.

SUPERCHIP has a total of 18 functions:

erase begin	**erase end**
scroll up	**scroll down**
escape	**retrace**
message	**functions**
movit	**single key**
delete line	**insert line**
scroll window	**graphics toggle**
hold	**stop**
shrink	**reverse**

Most functions can be accessed either in immediate mode or from a BASIC program. A concise reference chart on the rear cover of the 26 page user manual summarizes the functions and states any exceptions or limitations. The manual is, for the most part, easy to understand by a first time user. However, I did have trouble with the scroll window section and I was confused by the use of the word "bracket" for "open parenthesis."

SUPERCHIP will appeal, I think, to many people and, if the reader is in this group, by all means buy it. You will have added quite a few useful functions to your repertoire which are not available elsewhere. Programs which use these features will be able to generate displays which ordinary PETs cannot do without great difficulty. As for me, I am not convinced that it is a good buy. Even with the recent drop in the pound exchange rate, SUPERCHIP still costs more than comparable firmware such as the TOOLKIT. In its favor, however, is that it offers some desirable features of BASIC 4.0 without losing the use of previously developed machine language software, which may be ROM-dependent.

Supersoft
28 Burwood Ave.
Eastcote, Pinner, Middx., England
£45

Spacemaker II

New Commodore CBM/PETs have empty ROM sockets on the main logic board which allow users to install special software packages. These include the CBM word processors, VISICALC, and the TOOLKIT. Recently I received for review two firmware (EPROM) programs which, alas, required installation in the same empty ROM socket. This meant that I could not switch back and forth between the two programs without risk of permanent damage to the IC pins (or to my psyche). My problem was solved when I received the SPACEMAKER II for review. This nicely crafted piece of hardware is capable of switching between as many as four different ROMs when plugged into a single socket on the PET logic board.

I had no trouble working with SPACEMAKER II. The hardest part is insertion of the ROMs, but this time everything went smoothly. Jumpers which depend on ROM type, are placed on posts and no soldering is required. SPACEMAKER II is particularly easy to plug into a socket on the PET as you can get a grip on it more easily than a much smaller ROM. The four page instruction leaflet had quite explicit directions and I noticed only one typo (figure 2 instead of figure 3). The version I received employed manual switching with a switch mounted on the side of the PET (no drilling necessary). It is also possible to switch using software, with control by the User Port or with optional hardware (ROMDRIVER).

I have no hesitation about recommending this hardware to PET users who require software on ROM, but have addressing conflicts. SPACEMAKER II is professionally done and is reasonably priced. My only gripe is that they did not include a circuit diagram in the unlikely event that service is needed. In a way I'm glad they didn't as I was hard pressed to find any negative comments.

CGRS Microtech
P.O. Box 102
Langhorne, PA 19047
$39.00

SORT

(3.0 or 4.0 ROMs)
(40 or 80 column screen)

If you do much computing you will eventually need a good sort routine. I started to write a program recently which sorted and printed the names of up to 256 programs on PEDISK I diskettes. To my horror, I realized that I did not have, in my "junk box" of programs, a fast sort routine. **COMPUTE!** came to my rescue. An early issue compared sort routines and I was able to adapt one of the BASIC listings in the article. However, not everyone has the ability or inclination to fit published programs to their own use. Matrix software offers a SORT program (on EPROM) for people who need a fast machine language sort that can be used with a minimum of effort even by novice programmers.

I had little trouble writing my first simple sort program. Their seven pages of instructions were quite helpful. I was able to do a four character sort on 1000 items in under seven seconds (average). Try doing that in BASIC sometime and you will be as impressed as I was. The program is executed with a SYS call after a few required POKEs. For example,

```
POKE 905,a — which dimensioned array
POKE 906,b — number of keys
POKE 907,c — dimension of array
POKE 927,d — number of characters to evaluate
POKE 947,e — what character to begin sort at
SYS 36864 — for EPROM at $9000 (specify when
ordering)
```

For review purposes only, the company included a demonstration program. I believe they should include a listing of this program with each order. Otherwise, I have no complaints about this package. The sort is fast. It works with integers, real numbers, or strings. And, as a bonus, they include a printer screen dump in the unused space on the ROM. Check this program out if you do lots of sorts and you need a fast routine resident at all times. You should find it very useful.

Matrix Software
315 Marion Ave.
Big Rapid, MI 49307
$55

Machine Language: Jumbo Numbers

Jim Butterfield
Toronto, Canada

A single byte will hold an unsigned number whose value may be from 0 to 255. Most of us, sooner or later, want to handle larger numbers. The techniques are fairly straightforward.

A number may occupy several bytes of storage. The usual convention is for the higher order bytes to contain powers of 256. In simple terms, this means that one byte counts in "ones"; another byte counts in "256-s"; the next byte, if used, counts in "4096-s" and so on. It's easier than it sounds if you convert the number to hexadecimal. One million, which in hexadecimal is 0F4240, fits nicely into three bytes: from high order to low order these bytes contain 0F, 42, and 40 hexadecimal.

It is possible to hold numbers in a decimal type of format. This makes input and output easy, since no conversion is needed to convert the decimal digits, and addition and subtraction can be quite easily accomplished. More complex arithmetic is difficult — even multiplication and division requires an effort — so that we choose binary if any real math crunching is needed. Decimal numbers can be held two ways: packed, with two digits to a byte; and unpacked, with one digit to a byte.

Sizing

We must make room for the largest possible numbers we expect to handle. The following table may be halpful:

	Unsigned	Signed	Packed Decimal
1 Byte:	0 to 255	−128 to +127	0 to 99
2 Bytes:	0 to 65535	−32768 to +32767	0 to 9999
3 Bytes:	0 to 1677215	−8388608 to +8388607	0 to 999999

The table grows proportionately; if a count of over sixteen million in three bytes won't do, four bytes reaches to over four billion (after taxes, that's four thousand million in Great Britain). Enough for most applications, but you can continue to add bytes as you wish.

What about fractions? The most common method is to use an assumed decimal point. In other words, count in pennies instead of in dollars and you won't need fractions. There are more exacting methods, but most of us sidestep them if we can.

Memory Arrangement

There's really no special law regarding how you arrange these bytes in memory. You can have high order values at the higher addresses, or turn it around and have high order values at the low end. I like to have low order at the low address end, etc.: it's easier to remember and is more consistent with address modes. On the other hand, storing the bytes the other way around (high order at the low address) makes it a little easier to handle a number with indexing. Why? Well, if we have to test an index register for the end of its range with CPX or CPY, we'll affect the Carry flag ... and we often need that flag to link information between the various bytes. A fine point; the choice is really up to you.

You can even scatter the values through memory rather than having them consecutive. Often it's better to keep them together so that you can "walk through" a number using indexing. But there are exceptions to every rule.

Some Simple Operations

We can manipulate multi-byte numbers just as readily as single bytes. All we need is some new rules.

For the following sample code, let's assume a two-byte value stored in locations 0300 low-order and 0301 high-order.

Moving: move both bytes instead of one. To move 0300/0301 to 0320/0321 we might code: LDX #$01; LOOP: LDA $0300,X; STA $0320,X; DEX; BPL LOOP. We have moved the high order byte first, but this makes no difference.

Addition and subtraction: start at the low end; fix up the Carry flag before you start, and then let the Carry link the bytes together. To add the contents of $0300/0301 to $0320/0321 and place the result at $0320/0321, we might code: CLC; LDA $0300; ADC $0320; STA $0320; LDA $0301; ADC $0321; STA $0321. Note that it's vital that we start at the low end of the numbers, in this case the low addresses. We might wish to check to insure that the result hasn't overflowed (overflew?) the space available. For unsigned numbers, we do this by checking that the Carry flag is clear.

Subtraction goes the same way, except we give SEC and use the SBC command. A valid subtraction will complete with the Carry flag set; otherwise there's an unsigned number overflow.

Comparisons

Comparison is a little different from the single-byte compare. We need to decide in advance if we're testing for equality or for greater-than; it's hard to check for both in a single sequence.

80 x 25

PET/CBM™

2000/3000/4000 Series

not using a CRT, or display controller chip

$275.00*

Select either **80** x **25** or **40** x **25** On The Built-in Display

From the keyboard or program

Displays the full, original character set

Available from your local dealer or:

EXECOM CORP.
1901 Polaris Ave.
Racine, WI 53404
Ph. 414-632-1004

*Plus installation charge of $75.00

Available only for Basic 3.0 & Basic 4.0

PET& CBM™s trademark of Commodore Business Machines

Commodore 3.0/4.0/2.1 Dos Source Code!

Complete annotated source code for 3.0 ROMs, 4.0 ROMs and 2.1 DOS (4040-,8050-disk) now available. All entry points, routine/variable names, info on routine operation, register contents, etc. included. 4.0 source code comes with 3.0 correspondence codes so you can change 3.0 machine-language programs to 4.0 and vice versa. Hardcopy only. $129. each. All 3 (3.0, 4.0, 2.1 DOS) for $310.00. Source files on diskette on special order.

Software/Firmware

- EXTRAMON extended monitor in 2K EPROM for any slot, 3.0/4.0 ROMs, plus hardcopy documentation. $19.95
- DISK MONITOR© - reads T/S from disk to screen at 0.21 seconds/blk. Allows editing like the resident monitor, updates disk, traces links, etc. FAST! 31.1 Kbytes/sec. $49.95
- BULLETPROOF!© - full-screen input routine. Uses machine-language to get user input, places it in a user-defined variable. Text on screen outside fields impossible to disturb. Documented source code incl. with disk. $29.95

THE GREAT WESTERN SOFTWARE CO.

Authorized dealer for RPL® compilers

1624 N.W. 9th Ave., Ft. Lauderdale, FL 33311 (305) 523-1351

PETTERM

INTELLIGENT TERMINAL

Turn your PET into an intelligent terminal with one of our terminal packages. These are complete assembled hardware and software packages. All include line editing/resend, repeat key, shift lock, output to CBM printer, and more . . . Delivered on PET cassette with manuals.

PETTERM I	All features above	$75.00
PETTERM II	All features of I, plus local text editor with down-loading capability . .	$90.00
PETTERM III	All features of II, plus 80/132 column scrolling window for viewing formatted outputs wider than 40 columns.	$100.00

forth

for PET/CBM

FORTH is a new concept in programming, with the speed of compilers and interactive ease of BASIC. Programs become a part of FORTH extending the power of FORTH and your PET.

8050,4040 disk, cassette all PET-CBMs 16k+

Starter	fig-FORTH w editor assembler	$35
Personal	floating point; strings; source	$75
Professional	turnkey development/data base	$259

F S S

software for small computers

1903 Rio Grande
Austin, Texas
78705

1-512-477-2207

P.O. Box 8403
Austin, Texas
78712

DEALERS INQUIRE

Equality tests are quite straightforward: test each of the pairs of bytes, and if any are not the same, the two values are unequal. We might code: LDX #$01; LOOP: LDA $0300,X; CMP $0320,X; BNE UNEQUAL; DEX; BPL LOOP; EQUAL: ... The code is fairly self-evident.

To compare for greater-than, we might do a full subtraction. We won't need to keep the result; the flags will tell us the answer. We might code: SEC; LDA $0300; SBC $0320; LDA $0301; SBC $0321. At this point, the Carry flag will be set if the value in $0300/0301 is greater than or equal to that in $0320/0321.

It's possible to compare from the high-order end down, on the theory that if the first byte is. different, you don't need to look at the rest. Additionally, such a comparison can more easily test both equal and greater-than conditions. There's often not much difference; speed is likely to depend on whether or not the numbers are likely to be close or far apart.

Shifts And Rotates

Shifts and Rotates propogate readily through the Carry bit. The first operation must start at the proper end of the number: Right shifts start from the high end, Left shifts from the low. The remaining operations, which work their way through the number, must always be Rotates, regardless of whether the overall operation is Shift or Rotate.

To shift the two-byte number at $0300/0301 left, we might code: ASL $0300; ROL $0301. To rotate the same number, we would give ROL $0300; ROL $0301.

To shift the same number right, we would code; LSR $0301; ROR $0300. Finally we would rotate the number right with ROR $0301; ROR $0300.

Big numbers are not much harder to work with than small ones. All the usual operations are still available to you. There are more items to keep track of, but that's a natural result of expansion.

Make provision for future big numbers now. You wouldn't want to tell your boss that he can't give you a raise because there isn't room enough in the computer to hold what he wants to pay you...

File Recovery

M. R. D'Amato
Plainfield, NJ

If you have inadvertently scratched a file (and who hasn't?) on the 2040 (DOS 1.0) or the 8050 disk drive you can easily recover the information if you avoid saving additional files on the disk. (For really bad (slipped?) disk problems, see Cones' more sophisticated file recovery program; **COMPUTE!** #10.) The task is easier on the 8050, so let's start there.

On The 8050

When a program file is scratched, the file identifier, located in byte #0 of the 30-byte file entry in the directory, is changed from 130 to 0. Also, the blocks in which the file was written are recovered by DOS for subsequent reuse. Program 1 does the following. It searches the directory for scratched files and presents them on the screen one at a time. Press key "N" if the scratched file is not the one you want. (Also press "N" if all zeros appear as the file name, but don't respond to the directory track and sector numbers, which are provided for your information.) When the desired file appears on the screen, press "Y." This results in changing the file label from 0 to 130 and depositing this value on the disk. If the directory is then accessed, the name of the scratched file will appear in the displayed directory, and the file can be loaded into memory.

It is essential that the file be reSAVEd or it will be again lost when DOS assigns one or more of its blocks to a subsequently saved file. Therefore, after recovering the file and loading it into memory you should (a) SCRATCH the file (to remove it from the directory) and (b) SAVE the file under its original or a new name. That's all there is to it.

And the 2040

The task is a bit more difficult on the 2040 with DOS 1.0 because the track number of the first data block is, like the file identifier, also set to 0. However, it is usually possible to infer the number of the initial track by examining the starting track numbers of the neighboring files in the directory. Program 2, which lists a block of data on the screen, is meant to help in this task. When prompted for track and sector, enter 18,1, the first directory block. If your scratched file is in this directory block, it can be identified by its name, which appears in ASCII format. The first two bytes of the block give the track and sector of the next directory block (18,4). The 30 bytes of the first file entry follow.

As already noted, byte 0 holds the file type. The track and sector of the first data block are located in bytes one and two, followed by a file name in bytes 3-18, padded with shifted spaces (160's). A total of eight file entries (separated by two zero bytes) can be contained within a directory block. If your scratched file is not in the first directory block, access the next directory block by running Program 2 and entering 18, 4, continuing the process as necessary.

After locating the lost file, compare the track and sector number of the first data block in the preceding and the following file entries. These values will often immediately suggest the number of the first track of the scratched file. Having inferred the initial track of the scratched file, add lines 145, 414, and 416 to Program 1, change T in line 170 to 18 and run the program. It's a good idea to work with a duplicate disk, just in case you have the wrong track number and cause DOS some confusion.

Sequential files can also be recovered by changing the CHR$ (130) in line 410 of Program 1 to CHR$ (129). As with program files, once the sequential file is recovered, the name of the original file should be scratched from the directory and the recovered file saved.

The omission of a disk-error handling routine in the program is a concession to simplicity. It seemed just as easy to rerun the program if anything went wrong, but it's simple enough to include an error routine if needed.

The 8050 recovery program also works for files generated by the Wordpro 4 word processor, which stores text as program files. In fact, it was the humiliation of having mindlessly scratched a couple of such files that led to the development of the present program.

Murky BAM

For those of you who might want to poke around in murky BAM (block availability map), Programs 3 and 4 will illuminate things a bit by highlighting, in reverse field, the BAM bytes associated with a particular track (four on the 2040 disk and five on the 8050). In both cases, the first byte reveals the number of free blocks in the specified track and the subsequent bytes indicate their identity. Block zero is represented by byte zero of the second byte, block eight is represented by byte zero of the third byte, and so on. Not a very intuitive layout, but computers have little concern for such matters.

Because the DOS support program ("wedge") jumps into action when it sees the ASCII of >, /, ↑ or @, it's best not to have the wedge concurrently in memory when using Programs 1-3 on the 2040 with DOS 1.0. This is not a problem on the 8050 and may be on the 2040 with versions of DOS ›

1.0. As if in compensation, the 8050 (but not the 2040) may give a "70, no channels" error if the disk holding the scratched program is not accessed with a load or a directory command after initialization. If this occurs, simply display the directory and rerun the program.

Program 1.

```
100 REM ***RESTORE SCRATCHED PROGRA
    M FILE***********
110 REM ***ON 8050 DISK DRIVE*** M.
     R. D'AMATO *****
120 OPEN15,8,15
130 OPEN1,8,3,"#"
140 PRINT"DRIVE NUMBER 0 OR 1?":INP
    UT D
150 PRINT"PRESS Y IF FILE IS FOUND,
     PRESS N IF NOT"
160 REM ***FIND AND PRINT SCRATCHED
     FILES***********
170 T=39:S=1
180 PRINT#15,"B-R:"3;D;T;S
190 PRINT#15,"B-P:"3;2+32*R
200 GET#1,A$:IFA$=""THENA$=CHR$(0)
210 IFASC(A$)>128THEN300
220 FOR K=5+32*R TO 20+32*R
230 PRINT#15,"B-P:"3;K
240 GET#1,A$:IFA$=""THENPRINT"0";:G
    OTO260
250 PRINTA$;
260 NEXT
270 PRINT
280 GETA$:IFA$=""THEN280
290 IFA$="Y"THEN400
300 R=R+1:IFR<8THEN180
310 REM ***GET NEXT DIRECTORY TRACK
     & SECTOR********
320 PRINT#15,"B-P:"3;0
330 GET#1,A$:IFA$=""THENA$=CHR$(0)
340 T=ASC(A$):IFT=0THENPRINT"FILE N
    OT FOUND":GOTO430
350 PRINT#15,"B-P:"3;1
360 GET#1,A$:IFA$=""THENA$=CHR$(0)
370 S=ASC(A$):PRINT T,S
380 R=0:GOTO180
390 REM ***RESTORE PROGRAM FILE LAB
    EL (130)*********
400 PRINT#15,"B-P:"3;5+32*R-3
410 PRINT#1,CHR$(130);
420 PRINT#15,"U2:"3;D;T;S
430 CLOSE1:CLOSE15
```

Program 2.

```
100 REM **READ A BLOCK ON 2050 OR 8
    050**
110 OPEN15,8,15
120 OPEN1,8,3,"#"
130 PRINT"ENTER TRACK AND SECTOR (B
    LOCK)"
135 INPUT T,S:D=1:REM D=DRIVE NUMBE
    R
140 PRINT:PRINTT"-"S":";
150 PRINT#15,"B-R:"3;D;T;S
160 FORK=0TO255
170 PRINT#15,"B-P:"3;K
180 GET#1,A$
190 IFA$=""THENPRINT"0";:GOTO210
200 PRINTASC(A$);
210 NEXTK
220 CLOSE1:CLOSE15
```

Program 3.

```
100 REM **BAM HIGHLIGHT PROGRAM 204
    0**
105 REM ******** M. R. D'AMATO ****
    ***
110 OPEN15,8,15
120 OPEN1,8,3,"#"
130 PRINT"ENTER TRACK FOR WANTED BA
    M"
135 INPUT T:D=1:REM D=DRIVE NUMBER
140 PRINT:PRINT18"-"0":";
150 PRINT#15,"B-R:"3;D;18;0
160 FORK=0TO255
170 PRINT#15,"B-P:"3;K
180 GET#1,A$
182 IFK=>(4*T)ANDK<4*(T+1)THENPRINT
    "{REV}";
190 IFA$=""THENPRINT"0";:GOTO210
200 PRINTASC(A$);
210 PRINT"{OFF}";:NEXTK
220 CLOSE1:CLOSE15
```

Program 4.

```
100 REM **BAM HIGHLIGHT PROGRAM 805
    0**
105 REM ******** M. R. D'AMATO ****
    ***
110 OPEN15,8,15
120 OPEN1,8,3,"#"
130 PRINT"ENTER TRACK FOR WANTED BA
    M"
135 INPUT T:D=1:REM D=DRIVE NUMBER
137 IF T>50THENS=3:T=T-50:GOTO140
138 S=0
140 PRINT:PRINT38"-"S":";
150 PRINT#15,"B-R:"3;D;38;S
160 FORK=0TO255
170 PRINT#15,"B-P:"3;K
180 GET#1,A$
182 IFK=>5*T+1ANDK<5*T+6THENPRINT"{
    REV}";
190 IFA$=""THENPRINT"0";:GOTO210
200 PRINTASC(A$);
210 PRINT"{OFF}";:NEXTK
220 CLOSE1:CLOSE15
```

©

DR. DALEY'S SOFTWARE

Software with a difference.

THE MAIL LIST

A Proven Performer

Proven Performance with The Mail List is now in use by many satisfied customers.

It has been nearly two years since the first version of The Mail List was introduced. Only the BEST software will stand this test of time.

The Mail List has been specially designed to be used by the novice computer operator. All operations in the system are menu driven with built in user protection. This insures that you can have hassle free and error free operation.

Why waste time with other inferior mailing lists? Compare these features:

1. **User defined data structures.** You are the best judge of how your files should be organized. DR. DALEY'S mailing list package is unique in this feature. With The Mail List you can divide each record to suit your needs.
2. **User defined label format.** You can print from one to eight labels across and up to 10 lines per label.
3. **Interface to WordPro 3 or 4.**
4. **Fast and easy input and editing.**
5. **Easy to use 'wild card' sorting.** This will allow searches through the file using up to 3 fields.
6. **Multiple disk files.** Maximum capacity is 80 disks per file.

This powerful package comes on diskette with nearly 100 pages of documentation. It is packaged in an attractive binder.

When ordering please specify the version you are ordering. It is available for the following systems:

Version 4.4 $159.95
(Any computer with Commodore BASIC 4.0 and 32K memory with the 4040 (or upgraded 2040) disk drive.)

Version 4.8 $179.95
(Commodore 8032 with the 8050 disk drive.)

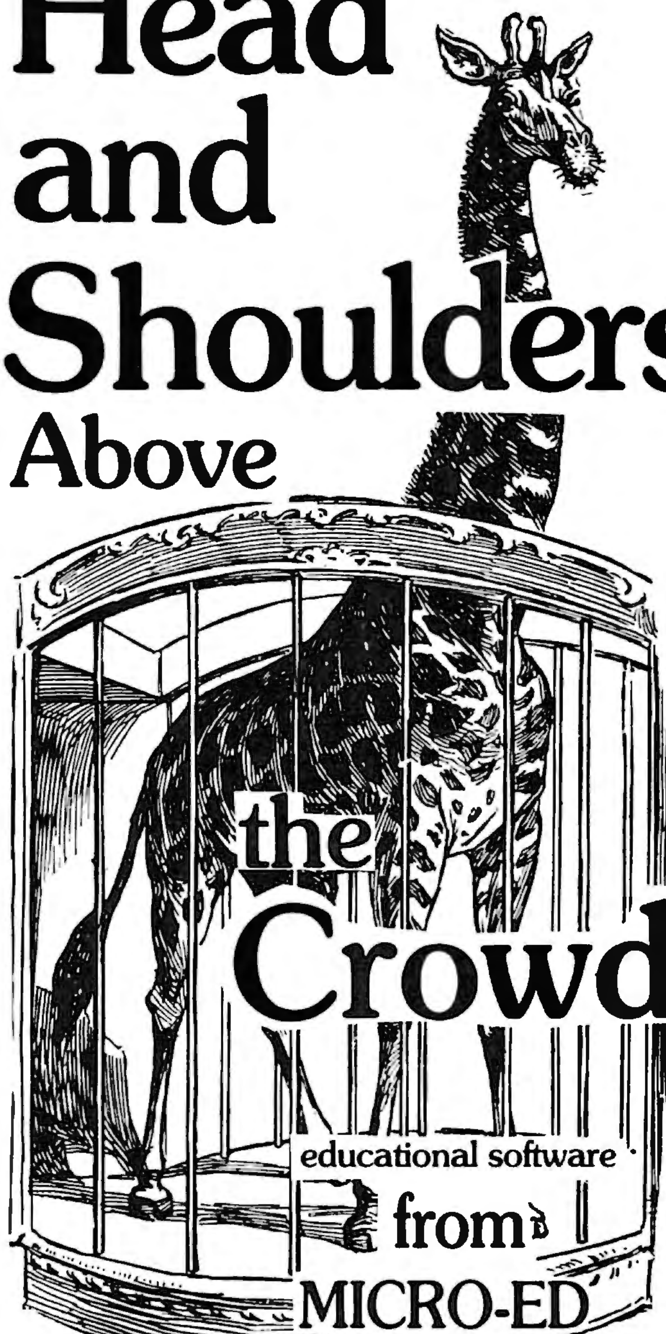
Call or write for details of our other software.

NOTE OUR NEW ADDRESS

DR. DALEY'S SOFTWARE
Water Street
Darby, MT 59829
Phone: (406) 821-3924
(Hours: 10 a.m. to 6 p.m. Mountain Time)

MasterCard VISA

Head and Shoulders Above



the Crowd!

educational software from MICRO-ED for

PET® and VIC®

Send for our free catalog*

*please specify PET or VIC

MICRO-ED, Inc. • P.O. Box 24156
Minneapolis, MN 55424
or telephone us at (612) 926-2292

VIC and PET are the registered trademarks for Commodore Business Machines

Looney Line Numbers

Jim Butterfield
Toronto, Canada

It should never happen. You have a program that you've been working on for hours (days? weeks?) and then suddenly a line number goes wrong. In between lines 6340 and 6360 the line number that should be 6350 has suddenly changed to 2254. Not only is that wrong — the GOTO's won't work right — but you can't get rid of it! The line seems stuck in your program forever. How does it happen? More to the point, how do you get rid of it without completely reentering the program?

How It Happens

It won't happen under normal circumstances. BASIC guards carefully against this kind of error.

An unwise POKE instruction or a SYS to a machine language program that's not completely debugged can get you into all sorts of trouble. If you're lucky, all you'll get is a looney line number.

Sometimes a bad LOAD will do the trick. In theory, the computer should guard against load errors; but it doesn't always tell you the whole story. If you're loading tape on a CBM/PET, always ask for the Status value (type PRINT ST): if the value is zero, the load is reliable; otherwise, you're taking your chances.

Bad RAM (Random Access Memory) can plague you with faults. It's not always obvious. Memory can sometimes fail erratically: perhaps the power supply voltage drops for a moment, and a bit disappears; or the malfunction only starts after the computer's innards get hot. If you're plagued with this type of problem, have your machine checked out.

All of the above may cause goofy line numbers; but they also may randomly cause other errors. Some are fatal, and some cause your program to look weird. Try to pin down the cause; it's worth the effort.

Fixing Numbers That Are Too High

There are two cases: high line numbers (out of proper order) and very high line numbers.

If an out-of-sequence line number is too high, but less than 64000, the trick is easy: delete the bad line and reenter it with the proper line number.

If the line number is 64000 or more, we must go to the next section and run the program there. You're not allowed to enter a line number of 64000 or more, even to delete the line concerned. Try typing 64000 followed by RETURN; you'll get a ?SYNTAX ERROR.

Fixing Low And Super-high Numbers

Type in the following lines at the front of your program. If your program happens to have lines numbered in the range from 0 to 8, take them out and put them back later.

```
1 A=1025: V=256: X=-1
2 B=A:A=PEEK(B)+PEEK(B+1)*V
3 PRINT : IF A=0 THEN END
4 Z=PEEK(B+2):Y=Z+PEEK(B+3)*V
5 PRINT CHR$(145);Y;:IF Y>X AND Z<250 GOTO 8
6 Y=X+1 : Y%=Y/V:PRINT"TO";Y
7 POKE B+2,Y-Y%*V:POKE B+3,Y%
8 X=Y:GOTO 2
```

The above coding is for the PET/CBM; you can adapt it to other computers by changing the value 1025 in line 1 to the Start-of-BASIC address in your machine. The CHR$(145) can be changed to match your machine's Cursor-Up character.

Meaning of the variables: B is the address of the current line of BASIC being examined; A is the address of the next line. X is the previous line number and Y is the new line number. Z is the "high byte" of the new line number; it's used to test for a super-high number. V is a constant of 256.

The program goes through each line of BASIC including itself and checks that each line number is higher than the previous one and not over 63999. If the line number fails the test, it is set to one higher than the previous line number.

Note the logic: can you see why the program must not be used on a normal "too-high" looney line number? It would "pass" the bad line number, and then bump up the numbers on all following lines.

What do you do if you have both too-low and too-high? Fix the too-high first before you run this program. If you do have multiple faults, chances are your program is in really bad shape anyway; get your computer fixed and redo the whole program.

Looney line numbers should never happen. Look for the cause if it happens to you.

You can fix them, however. And the mechanics of fixing bad line numbers has a tiny bonus: look at the coding and see if you can gain an insight into how BASIC is put together.

Super-coders can go after the same problems by attacking the program directly as it lies on disk, copying the program over and correcting it on the way. Users with BASIC enhancement packages (Toolkit, Command-O, Power, etc.) can fix everything in a trice with program renumber.

There are many ways of fixing it ... once you know how. ©

Editor's Note: Stephen suggests that this game can be a source of ideas for a variety of other games. If you come up with an exciting variation, send it in and share it with other **COMPUTE!** *readers. — RTM*

Mine Maze

Stephen Vermeulen
Calgary, Canada

This two-player game for 40 column PETs illustrates how the character oriented graphics of the PET, coupled with keyboard input from only three keys, can produce a fast, frustrating, and addictive game. Also, by keeping the graphics as simple and as clear as possible, implementation in BASIC is practical.

Rules Of The Game

A random maze with a clear border around it is drawn on the screen. The two players are then placed within the clear border and play starts. The object, for both players, is to be the survivor of what might appear at first to be a one-sided conflict. The aggressor in the battle is the left-hand player who is represented by the solid ball (shifted Q) graphic character. The ball can only win by running into his opponent, the circle (shifted W).

The circle is usually the defensive player, and is able to drop mines on the playing field. When the ball hits one of these mines it looses one life. To even things up a bit, the ball is given one free life for every ten mines the circle deposits on the playing field and, also, the number of mines that must be dropped before the ball gets its first free life is set randomly.

Lines 100 to 280 print the instructions and get the players' names and the difficulty factor. Default values for these inputs are provided so that eager (or lazy) players can get into play by pressing return 3 times. The next portion (lines 290–380) sets up the playing field and starts the play. The graphic characters used are Q [81] for the ball, W [87] for the circle, [102] for the maze, and * [42] for the mines. The values in brackets are the screen POKE values. The last section (lines 600–680) of the program displays the score and prompts the players for another game.

Heart Of The Program

Now that the sundries have been discussed, the heart of the program can be dissected. The keypress interpreter is the code found in lines 390–410. The branches on "@" and "=" to lines 470 and 480 serve to rotate the player's direction of movement 90 degrees clockwise. The branch on "*" proceeds to lines 490–510 which drop a mine and increment the mine counter and the ball's life counter. After the present key press has been processed the program then moves both players. Before a player can be moved, the program must check for obstacles and hazards by PEEKing the next position and, if necessary, going to one of the two obstruction test routines. The first of these (lines 520–560) is for use when moving the ball. Here the next square is checked to see if it is a maze wall (the ball bounces), or the circle (the ball wins the game), or a mine (the ball looses a life and possibly the game). The second routine (lines 570–590) is the obstacle routine for the circle, here the next square is checked to see if it is a maze wall (making the circle bounce back) or the ball (the ball wins). If the square happened to contain a mine the circle would just erase it (which can be very frustrating if you are controlling the circle).

Official Decrees

And now for some final rules for situations which do arise (these rules were adopted for play in the most recent World Mine Maze 1.9 Championships consisting of a round-robin three player tournament):

1. It is decreed that, the Mine Layer shall *not* lock himself in between the outer boundary and a wall of the maze (see Figure 1).
2. It is decreed that, if the Mine Layer has successfully sealed himself off in the inner part of the maze, the Ball must commit suicide by running into as many mines as it takes to die.

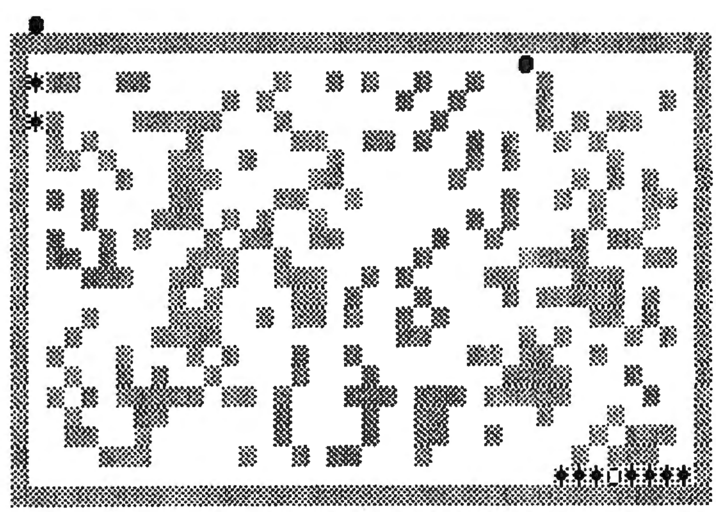


Figure 1: The circle is not allowed to win by blocking off an edge position such as this because it is so easy to do.

```
100 POKE59468,14
110 REM MINES VS DESTROYER
120 PRINT"{CLEAR}{04 DOWN}            MINE
    S OR BUST"
130 PRINT"{02 DOWN}INSTRUCTIONS:
140 PRINT"{DOWN}THE PLAYER ON THE LEFT US
    ES THE @ KEY
150 PRINT"TO STEER THE BALL CLOCKWISE."
160 PRINT"{DOWN}THE PLAYER ON THE RIGHT U
    SES THE = KEY
170 PRINT"TO TURN THE CIRCLE CLOCKWISE, A
    ND THE *
180 PRINT"TO PLACE A MINE ON THE FIELD.
190 PRINT"{DOWN}IF THE BALL [@] HITS A MI
    NE THE CIRCLE
200 PRINT"WINS, IF THE CIRCLE HITS A MINE
     THE MINE";
210 PRINT"IS DESTROYED.
220 PRINT"{REV}THE{OFF} {REV}BALL{OFF} {R
    REV}GETS{OFF} {REV}AN{OFF} {REV}
    EXTRA{OFF} {REV}LIFE{OFF} {REV}F
    OR"
230 PRINT"{REV}10{OFF} {REV}MINES{OFF} {R
    REV}PLACED{OFF} {REV}ON{OFF} {RE
    REV}THE{OFF} {REV}FIELD"
240 PRINT"{DOWN}IF THE BALL HITS THE CIRC
    LE THEN THE
250 PRINT"BALL WINS.
260 INPUT"{DOWN}RIGHTIST___RIGHT{08 LEFT}
    ";C$
270 INPUT"{DOWN}LEFTIST____LEFT{07 LEFT}"
    ;B$
280 INPUT"{02 DOWN}DIFFICULTY 1...20  9{0
    3 LEFT}";DD
290 POKE59468,12
300 MC=INT(9*RND(0)+1)
310 REM SET UP PLAYING FIELD
320 PRINT"{CLEAR}":S=32768
330 FORI=0TO39:POKES+40+I,102:POKES+1000-
    40+I,102:NEXTI
340 FORI=2TO23:POKES+40*I,102:POKES+40*I+
    39,102:NEXTI
350 FORI=1TO25*DD:POKES+INT(36*RND(0)+2)+
    40*INT(20*RND(0)+3),102:NEXTI
360 U=-40:D=40:L=-1:R=1
370 B=S+81+160:BD=D:DB=D:C=S+78+40:DC=D:C
    D=D
380 POKEB,81:POKEC,87
390 GETP$:IFP$="@"THEN470
400 IFP$="="THEN480
410 IFP$="*"THEN490
420 X=PEEK(B+BD):IFX<>32THEN520
430 POKEB,32:B=B+BD:POKEB,81
440 X=PEEK(C+CD):IFX<>32THEN570
450 POKEC,32:C=C+CD:POKEC,87
460 GOTO390
470 DB=-1/DB:BD=BD*DB:GOTO390
480 DC=-1/DC:CD=CD*DC:GOTO390
490 MC=MC+1:IFMC=10THENMC=0:BL=BL+1:POKES
    +BL,81
500 IFPEEK(C-CD)<>102THENPOKEC-CD,42:GOTO
    390
510 GOTO390
520 IFX=102THENBD=-BD:GOTO440
530 IFX=87THENBS=BS+1:GOTO600
540 IFX<>42THEN430
550 BL=BL-1:POKES+BL+1,32:IFBL>=0THENGOTO
    430
560 CS=CS+1:GOTO600
570 IFX=102THENCD=-CD:GOTO390
580 IFX<>81THEN450
590 BS=BS+1
600 FORI=1TO100:GETP$:NEXTI
610 GETP$:IFP$=""THEN610
620 POKE 59468,14:PRINT"{CLEAR}{05 DOWN}
                  SCORE
630 PRINT"{03 DOWN}"B$" ="BS;TAB(37-LEN(C
    $+STR$(CS)));CS"= "C$
640 PRINT"{04 DOWN}NEW DIFFICULTY (YES, N
    O, END)
650 GETP$:IFP$="E"THENEND
655 MC=0:BL=0
660 IFP$="Y"THENPRINT"{CLEAR}":POKE59468,
    14:MC=0:BL=0:GOTO280
670 IFP$=""THEN650
680 GOTO290
```

©

COMAL: Another Language?

Jim Butterfield
Toronto, Canada

There are a lot of "languages" around for small computers. Only two are in common use: BASIC (most often by Microsoft) and Machine Language.

Most of the others classify as in-between languages: not as friendly as BASIC, not as fast as machine language. Each have their advocates (fanatics?) who extol the advantages which a specific language can bring to a specific application. But BASIC and machine language look like they will reign supreme for quite a while yet. Until local computer shops bristle with book titles such as "101 FORTH Games," "Some Common SMALL-TALK Programs," and "Hands-On SNOBOL For The PET/CBM," many users will prefer to stay in the mainstream of the home computing community and learn one or two simple languages well.

Now we have COMAL to add to the computer Tower of Babel. Another language, another chance to sidetrack? COMAL does have its interesting features, especially to those raised on BASIC. Let's take a look.

Free!

COMAL is public domain. In other words, it's free; and you can help yourself to a friend's copy with a clear conscience. In Canada, Commodore has distributed copies to all dealers; you can get one by asking your dealer to make you a copy of the disk and documentation. In the USA, the COMAL Users Group, 5501 Groveland Terrace, Madison WI 53716, will send you the disk for a charge of $12.95; or for $47.50 you get a kit including disks, documentation, binder, and a newsletter subscription; in either case, add $2.00 for shipping/handling.

You get what you pay for, right? Not in this case: COMAL is a massive system (it will fit only into a 32K system with disk) and has features that make it well worth considering, particularly for educational use.

COMAL came into existence in Denmark. It was first defined by Borge R. Christensen and Benedict Leofstedt in 1974. At that time it was a form of extended BASIC. It has been expanded and refined into the current version, COMAL-80, by Mogens Kjaer.

Super-Basic?

COMAL still retains much of the flavor of BASIC, and for that reason it's a very easy language for BASIC users to pick up. The first impression that a user gets is rather intriguing: it seems as if you may type in your program in BASIC — and when you say LIST, it comes back in a PASCAL-like language!

How can this happen? Most BASIC users learn that their program is "tokenized" as it is input. The individual letters of PRINT get scrunched together and stored as a single byte called a token; that's why you can type in a line such as 100 ? A and list it back as 100 PRINT A. COMAL tokenizes your input to a remarkable extent, so that a line input as 5 FOR J = 1 TO 8 will list back as 5 FOR J: = 1 TO 8 DO. Note that a colon has crept in after the J, and that the word DO has been added.

So: with a few new rules, you may just type in a program in its BASIC form, and COMAL will adapt it into its own internal format. Some of the new rules are easy: for example, be sure to put a space after each keyword (don't say FORK, say FOR K). Others take a little more practice: subroutines are now called Procedures and, instead of GOSUB-ing to them, you EXEC-ute them. And you don't use line numbers for GOTO and GOSUB, you use names (or "Labels"). But these are not difficult. It's easy for a BASIC person to jump into COMAL.

...And More

But it's more than just a reworded BASIC system. There are a whole new series of capabilities.

Some are easy to understand and on most people's wish-lists. You may now use variables HORSESHOE and HOUSEFLY without confusion (in BASIC, only the first two letters are meaningful). IF has been beefed up to include ELSE and other features, allowing you to code IF M = 12 THEN M = 1; Y = Y + 1 ELSE M = M + 1 ENDIF. Note that we are using a semicolon instead of a colon to separate statements, and we terminate the IF sequence with an ENDIF. This isn't just needless bookkeeping: ENDIF allows us to set the range of the IF statement to part of a line or multiple lines.

Other COMAL features are recognizable as structured language extensions. The user will find CASE (replacing ON A GOTO...), WHILE, and REPEAT .. UNTIL. Procedures can be used as subroutines or as function definitions; and you may pass parameters to or from procedures. Strings require a little more care than in BASIC, but string handling is more powerful after you get used to it.

COMAL is fast. You'll see no loss of speed from BASIC.

However...

The language is nice, but it's new. You may have to wait a while before you find a community of other COMAL users around you. The BASIC language feature I miss most is the SYS command — at least I haven't found it yet. I like to be able to extend some programs with machine language inserts if necessary. Some 4.0 disk commands don't appear to be supported by COMAL; I haven't found a way to initiate a SCRATCH or COLLECT from the language. There doesn't seem to be a built-in exit to BASIC cold start, which would be a way around the previous problem.

COMAL for the PET/CBM comes in two forms; the smallest takes up 16K of memory. This may cramp the size of programs. COMAL does pack programs in memory more efficiently, but you have less space to work with from the start.

COMAL seems ideal for educational environments, particularly for those who like to teach structured programming techniques.

It's a helpful language in many ways: as easy as BASIC and very like it. It has goodies that BASIC can't match. The structure and balance of COMAL lead me to suspect that there will be a compiler along one of these days.

And the price is right. ©

CAPUTE!:
Corrections And Amplifications

—**COMPUTE!** #14, pg. 106, "The Apple Hi-Res Shape Writer." The following routine was missing (Program 2):

Program 2.

```
10 D$ =  CHR$ (4):INC = 10:S = 16
     384
20  PRINT D$"OPEN POKE ROUTINE"
30  PRINT D$"WRITE POKE ROUTINE"
90 LINE = 5000:B = 16500
100  PRINT LINE"FORI=16384TO"B":R
     EADA:POKEI,A:NEXTI"
120  FOR I = S TO B: IF (I - S) /
     10 =  INT ((I - S) / 10) THEN
      PRINT :LINE = LINE + INC: PRINT
     LINE"DATA" PEEK (I);: GOTO 1
     40
130  PRINT "," PEEK (I);
140  NEXT : PRINT : PRINT D$"CLOS
     E"
]LIST
```

—**COMPUTE!** #16, pg. 66: Line 62005 should read:

FOR I=LO TO HI

— **COMPUTE!** #16, pg. 118: "The = sign does concatenation...". No it doesn't! The + sign does concatenation. Who wrote that? Who is this guy Butterfield anyway? He deserves thirty lashes with a wet noodle.

Unless, of course, he wrote it correctly and somebody goofed it up down there. In which case, transfer the lashes (and the noodle) to the appropriate culprit. Heck, I have enough trouble spelling concatenation without worrying about how to do it...

The whole thing is bristling with = signs that shouldn't be there. M$=A$=B$=C$ should be M$=A$+B$+C$; PRINT J$=" "=M$ should be PRINT J$=" "+M$. However, to make up for it, you've changed a character the opposite way in your last paragraph: Z$=Z$+"+" should read Z$=Z$+"=".

Try typing in this line:

FOR J=1 TO 10:PRINT"TSK.":NEXT J

[Our thanks to Jim for his corrections. The typos (and the noodle) belong here.]

—**COMPUTE!** #16, pg. 134: line 9010 should read FOR I=4 TO 35*PV STEP 5*PV. The value POKEd in line 9520 should be 43.

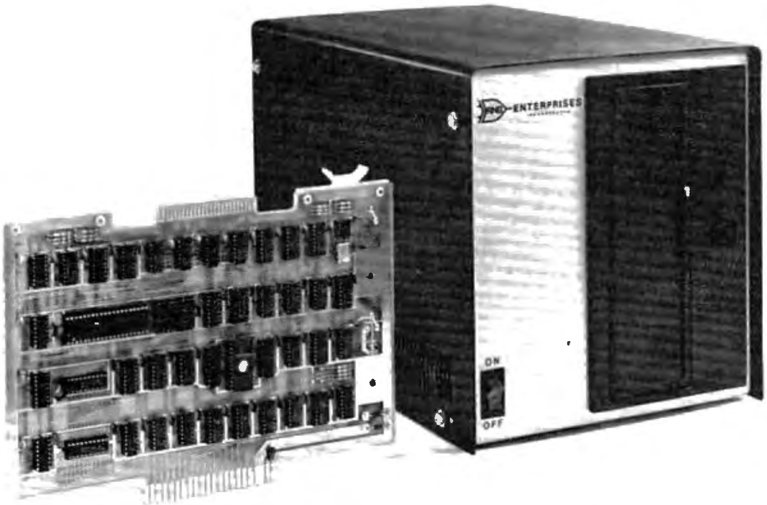
—**COMPUTE!** #16, pg. 124: To permit the program to also work on the 8050 drive, change line 290 to PRINT#15, "M–E"CHR$(180)CHR$(255)

—**COMPUTE!** #16, pg. 10: Many readers suggested modifications to David Thornburg's excellent 20-questions program to permit more random responses and to prevent the same response if the questions always began: "is it animal," "is it vegetable," "is it mineral." One of Mr. Thornburg's points was the brevity of the program in contrast to the "intelligence" it seemed to evidence. There are indeed a variety of ways to make the program even more remarkable for Turing tests on the unsuspecting.

—**COMPUTE!** #10, pg. 112: To allow the disk program to work with the 4044 CBM disk drives (or 2040s with upgraded ROM), change line 5012 to: IF P(0) < 25 THEN SM = 18 ©

VAK-7 8″ FLOPPY DISK SYSTEM

FOR AIM-65, SYM-1



The VAK-7 Disk System was specifically designed for use with AIM-65 and SYM-1 Microcomputer Systems. The VAK-7 is a complete full size (8″) Floppy Disk System. This system will Read, Write and Format IBM Single and Dual Density diskettes. The VAK-7 is available with both Single and Dual Sided Disk Drives. Each Disk Drive comes with its own cabinet and Power Supply. The VAK-7 can handle up to 4 disk drives, totaling more than 4.98 Megabytes of storage.

The VAK-7 Disk System incorporates both advanced hardware and innovative software designs. The addition of the VAK-7 produces a very powerful and useful computer system. Unlike most other disk systems, there is no requirement for the user to provide RAM to hold the Disk Operating System software. No valuable time is wasted loading in the DOS from cassette tape, because the VAK-7 DOS software is in onboard ROM. The VAK-7 is located above the 32K boundary (8000 HEX), leaving the user with a full 32K bytes of contiguous address space for his own use.

AIM-65—Allows the user to save and load object code thru the AIM Monitor; to load, save, and append Text thru the AIM Editor; to load, save, and append Basic Programs thru the BASIC INTERPRETER; to assemble directly from disk single or multiple file programs.

SYM-1—Allows the user to save and load Files for use with the SYM Monitor, SYM Basic, and RAE-1.

ADDITIONAL COMMANDS:

ACTIVATE A DELETED FILE
COMPRESS A DISK
RENAME A DISK FILE
COPY A DISK
FORMAT A DISK
DELETE A DISK FILE
INITIALIZE A DISK
LIST CATALOG

SPECIFICATIONS:

- Completely assembled, tested, and burned in.
- Occupies address 8000–8FFF for AIM-65, $9000–9FFF for SYM-1, or $E000–EFFF for KIM-1.
- IBM Format; Single Density (128 bytes/sector); Dual Density (256,512, or 1024 bytes/sector).
- All ICs are in sockets.
- Fully buffered address and data bus.
- Standard KIM-4* BUS (both electrical pin-out and card size).
- Designed for use with a regulated power supply, but has provisions for adding regulators for use with an unregulated power supply.
- Dimensions: Board—10″ wide x 7″ high (including card-edge). Cabinet—9.25″ wide x 10″ high x 16″ deep.
- Power Requirements: +5v DC @ 2 Amps.
  117 AC 60Hz @ 2 Amps.
- Shipping Weight: 25 lbs.

PRICING:

VAK-7 **$1,299.00**
Controller and One Single-Sided Drive

VAK-7A **$599.00**
Additional Single-Sided Drive with Cabinet and Power Supply

VAK-7B **$1,599.00**
Controller and One Dual-Sided Drive

VAK-7C **$899.00**
Additional Dual-Sided Drive with Cabinet and Power Supply

CALL OR WRITE FOR FREIGHT CHARGES

**KIM-4 is a product of MOS Technology/C.B.M.*

RNB ENTERPRISES INCORPORATED

4030 N. 27th Avenue, Suite D
Phoenix, AZ 85017
(602) 265-7564

New Products

Advanced Operating Systems Publishes 10 Program Household Package

Michigan City, IN, October 1, 1981 — Advanced Operating Systems has published a package of 10 programs useful in performing household duties. The programs are written in BASIC.

The "Mostly BASIC Household Program" package offers two programs which give a synopsis of energy consumption. The "Electric Energy Usage" selection compares energy units used during two different years. The "Gas Mileage Calculator" uses basic data to figure the gas mileage of a vehicle.

Two programs focusing on diet and eating habits of the user are the "Recipe Amount Calculator" and "The Basic Diet."

"The House Buying Guide" and "The Amortization Schedule" focus on budgeting and investing of money. The schedule calculates the balance, principal, interest, and cumulative interest for each month of a loan. The buying guide weighs answers from questions it gives the user and advises on the possibilities of making a profit from buying a structure. The program can be used whether the operator will be renting out the structure or selling it at a later date.

Two programs which offer no frills, but are unlimited in their usefulness, are the "Digital Stopwatch" and "The Message Taker." The stopwatch counts off minutes, hours, and seconds until told to stop. The message program records up to six messages and enables members of the household to leave "notes" for one another.

Medical expenses can also be cataloged through use of this package. The "Medical Expense Record files away the type of expense, cost, and cumulative total for each year.

"The Tarot Card Reader" is an entertaining program based on a deck of 78 cards used in fortune telling. The program answers questions from the user by picking 10 of the cards. The position they hold after being layed out has a meaning, as well as whether the figure on the card is right side up or upside down. The translation of the 10 cards is listed briefly by the computer. This program requires 16K to run.

The package is available through computer retail outlets. Advanced Operating Systems is the microcomputer software division of Howard W. Sams & Co., Inc., a subsidiary of International Telephone and Telegraph Corporation.

Smart Terminal Program For Atari® Features Autodialing

Redmond, WA — The MicroPeripheral Corporation has announced TSMART, the first smart terminal program written for the ATARI 800®, with provision for autodialing other computers. The program is availble on cassette with instructions for transferring to disk. TSMART permits transfer of BASIC programs between a remote host computer and an ATARI cassette or disk storage device. The autodial feature works in conjunction with the AUTO-MICROCONNECTION, a direct connection modem ($199.50), manufactured by the MicroPeripheral Corporation.

The program permits off-line text preparation (messages, manuscripts, letters, etc.) with a text editing or word processing program for on-line transmission. A built-in feature permits creation and storage of text, then transmission by TSMART for those who do not have a text editor.

TSMART also permits transfer of source code assembler files. The recipient can create the object code using an editor/assembler program. A separate command is available for transferring object (hexadecimal) code files, such as ATARI Music Composer Files.

An AUTOBUF feature will open and close the memory storage buffer automatically when uploading or downloading. TSMART recognizes the automatic buffer open/close (X-on/X-off) codes transmitted by TSMART or other compatible programs. Downloading from FORUM 80 bulletin boards is also accomplished automatically. The buffer can be "toggled" on and off as many times as desired while data is being downloaded. Another feature is software selectable half or full duplex operation.

The program will also automatically send messages to bulletin boards using the standardized block mode or 16 line prompt recognition message entry.

The program was written for

the ATARI 800® by Dr. James W. Clark. It can be used with any RS-232 compatible modem, although the dialer feature cannot be used with obsolete acoustic modems. TSMART is supplied in a protective binder with extensive easy-to-use operating instructions and is priced at $79.95. For additional information contact the Micro-Peripheral Corporation, 2643 151st Place N.E., Redmond, WA 98052, Telephone (206)881-7544.

Service Of Commodore Computers Begins At 38 TRW Service Centers

Valley Forge, PA, October 9, 1981 — Service of Commodore Business Machines, Inc., microcomputers has begun at 38 service centers operated by the Customer Service Division of TRW Inc.

As per a five-year agreement signed between Commodore and TRW in August, TRW will service and maintain Commodore microcomputers throughout the United States both on-site and at walk-in depots.

The first 38 service centers to complete comprehensive training on Commodore equipment and go on-line as part of the agreement cover some 75 percent of the nation's microcomputer users. Additional TRW centers will be brought on-line over the next year.

Commodore products covered by the TRW service agreement include the CBM™ 8032 central processing unit, the 4032N and 4032B central processors with 12-inch monitors and 8040 universal logic boards, 8050 disk drive, 4022 matrix printer, and 8010 communications modem.

The first group of TRW service depots handling Commodore equipment are in Atlanta, Birmingham (AL), Boston, Charlotte (NC), Chicago, Cincinnati, Columbus (OH), Dallas, Denver, Detroit, Fresno, Grand Rapids (MI), Hartford (CT), Houston, Indianapolis, Jacksonville, Little Rock, Los Angeles, Memphis, Miami, Milwaukee, New Orleans, Oklahoma City, Philadelphia, Phoenix, Pittsburgh, Richmond, Rochester, Sacramento, St. Louis, Salt Lake City, San Diego, San Francisco, Seattle, Springfield (NJ), Tampa, and Washington, DC.

"The opening of these TRW service centers for Commodore equipment effectively triples our service capability overnight," said James Finke, president and chief operating officer of Commodore.

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

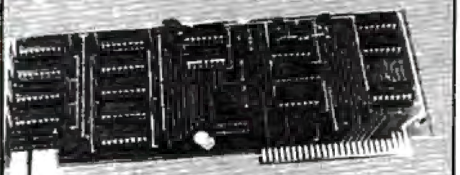
Language System w/Pascal & BASICS 379
Silentype Printer w/Interface card..........349
Hayes Micromodem II 299
Novation Apple-Cat 339
Videx Videoterm 80 column card 269
Videx Keyboard Enhancer 115
Z-80 Softcard by Microsoft..................299
16 K RamCard by Microsoft 169
CPS Multi-function card.....................189

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

Okidata Adds Interface Options To Microline Printers

Mount Laurel, NJ, October 14 — Okidata Corporation, a supplier of quality dot-matrix printers announced the availability of significant new interface options for its Microline family of printers. These options are an IEEE 488 bus adapter and a current loop interface.

The bus adapter option will make all new and existing Microline printers compatible with the IEEE 488 bus. Users of Commodore Pet personal computers will find this option particularly helpful in integrating Microline printers into systems. The IEEE 488 bus adapter option comes in the form of a plug-in board which is easily installed by the user. The connecting cable converts Centronics-compatible parallel data into data compatible with the IEEE 488 bus.

The IEEE 488 bus adapter option will be available later this month. This feature is priced competitively, and quantity pricing is availalble. The board can be used with the Microline 82A, 83A, and 84.

For those users who need a current loop interface, the optional Microline RS-232C high speed serial interface now offered will add a current loop interface capability as a standard feature. The interface board can also be used with the Microline 82A, 83A, and 84.

The high speed RS-232C interface board comes with two different buffer sizes: a two-kilobyte memory or a 256-character storage buffer. The interface has switch-selectable baud rates of up to 9,600 bits per second. The built-in current loop feature will be available soon. The single quantity price for the RS-232C board with two kilobytes of buffer memory and the current loop interface is $160; the price for the one with 256-characters of memory is $130.

Okidata Corporation is headquartered at 111 Gaither Dr., Mt. laurel, NJ 08054.

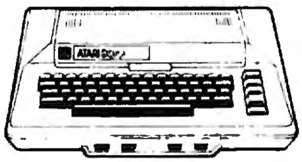
Software For The Very Young

In October Edu-Ware Services, Inc. releases a new addition to the EDU-WARE line, COUNTING BEE. This friendly system introduces young learners (ages 3-6) to counting, addition, subtraction, shape discrimination, weight, and measurement.

COUNTING BEE features a learning management mode, which allows parents and teachers to preset the system, with emphasis and duration tailored to an indi-

ATARI

Computers for people.™



800™ $749

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

400™ $329

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

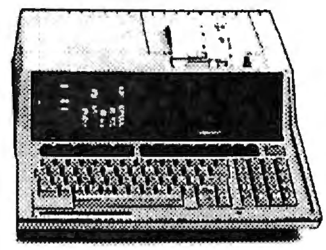
SOFTWARE

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

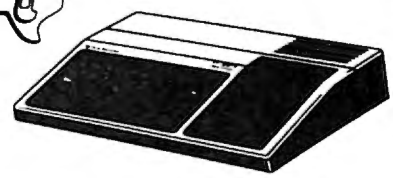


HP•85 $2595

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-99/4 $379

PHC 004 TI-99/4 Home Computer	$379.00
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

DISCS

Sycom Blank Disk (10)	$29.00
Maxell MD I	$36.00
Maxell MD II (10)	$46.00

PRINTERS

Epson MX-70	Call for Prices
Epson MX-80	
Epson MX-80 FT	
Diablo 630	
TEC 1500 Starwriter 25cps	$1495.00
TEC 1500 Starwriter 45cps	$1795.00

No Risk, No Deposit On Phone Orders, COD or Credit Card, Shipped Same Day You Call *

* on all in stock units

IN PA, CALL (717) 327-9575

(800) 233-8950

COMPUTER MAIL ORDER

501 E. THIRD ST., WILLIAMSPORT, PA 17701

OVER 40 YEARS EXPERIENCE
IN SOPHISITICATED ELECTRONICS

To Order:
Phone orders invited (800 number is for order desk only). Or send check or money order and receive free shipping. Pennsylvania residents add 6% sales tax. Add 3% for Visa or M.C. Equipment is subject to price change and availability without notice. Please call between 11 AM & 6 PM

vidual child's needs. COUNTING BEE joins other EDU-WARE systems designed for young learners, SPELLING BEE and ARITHMETIC SKILLS. The sys uses Applesoft, 48K, DOS 3.2 or 3.3, and retails for $29.95.

Special Learning Ed Software Presents Educational Software For The PET®

The program consists of individualized computer-based instructional activities in spelling for students with specific language learning disabilities. However, this program will improve the skills of any persons with spelling deficits. The ramifications of using the tapes are many which include teaching average students, slow learners, and those who are studying English as a second language. The program can be used by anyone who can read.

The author is a Language Therapist who has for many years tutored students in grades kindergarten through twelve. These students with specific learning differences are in need of specialized tutoring in the language areas.

The spelling program includes spelling rules, exceptions, and generalizations which provide repetitive exercises and reinforcement as well as motivation to the learner. The drill and practice which persons with specific language learning disabilities require to learn to spell can be provided through tapes to be used on the PET Commodore. All tapes work with any 40-column PET, old or new.

The program consists of 21 tapes covering five basic spelling rules essential to encode words. The complete program is also available on disk.

The copyrighted program includes:

F-L-S-or Z rule .... 4 Tapes / $39.80
Doubling Rule or One, One, One .... 3 Tapes / $29.85
Silent E Rule .... 4 Tapes / $39.80
The Y Rule .... 4 Tapes / $39.80
Extension of the Doubling Rule .... 6 Tapes / $59.70

You can SAVE by purchasing the entire series, tapes or disk, for $199.00. Individual spelling rules can be purchased.

For further information on this program write to:

SLED SOFTWARE
P.O. Box 16322
Minneapolis, MN 55416
Telephone (612)926-5820

PDI Establishes Division To Produce And Distribute Games

Program Design, Inc., the 4-year old Greenwich, CT educational software company is forming a division to produce and distribute entertainment software for the Atari 400 and 800 computer.

The division, called BEYOND SOFTWARE, is designed with independent software writers in mind. "We are interested in acquiring high-quality arcade, space, and simulation games," says PDI President, John Victor. "We are setting up a system that will cater to the needs of games designers. We will offer attractive terms, plus consulting services to top programmers."

BEYOND SOFTWARE is being established because the Program Design management feels the Atari computers offer exceptional entertainment possibilities.

BEYOND SOFTWARE will be managed by Craig Patchett, the author of CAPTIVITY, a 3-D maze game, and several other game programs.

For additional information, contact Patchett at Program Design, 11 Idar Court, Greenwich, CT 06830; (203)661-8799.

Six New SuperPET Books Now Available From Commodore

Valley Forge, PA, October 5, 1981 — Commodore Business Machines, Inc., has announced the availability of six new reference books to be used with its SuperPET "micro-mainframe-style" computer. The books are provided with the SuperPET system, but can also be purchased separately.

The SuperPET, which is Commodore's latest product addition, offers expanded capabilities by providing 96K RAM, an additional microprocessor, five languages, and a standard data communications interface.

The books include a System Overview of the SuperPET, and one book for each of the five languages available with the product — Waterloo microAPL, Waterloo microBASIC, Waterloo microFORTRAN, Waterloo microPASCAL, and Waterloo 6809 Assembler.

The System Overview book provides an introduction to the hardware of the SuperPET, an overview of the Waterloo software for the computer, and various descriptions that apply to the Waterloo software systems in general. The book retails for $5.95.

The Waterloo microAPL book, which retails for $9.95, is a tutorial introduction to the language, as well as a comprehensive reference manual.

The reference series also includes a Waterloo microBASIC book, which is divided into four parts: an introduction to the general characteristics of the system; a comprehensive reference guide describing the command language; an additional reference guide describing the programming language; and appendices containing summaries of both command and programming languages, as well as describing use of files with Waterloo microBASIC. It retails for $10.95.

The Waterloo microPascal book, also retailing for $10.95, features a tutorial introduction of Pascal language, and a reference manual defining the language.

The Waterloo microFORTRAN book is also divided into tutorial and reference sections, and retails for $10.95.

The final book in the series, pertaining to the Waterloo 6809 Assembler, describes the 6809

Assembler, Linker, and Monitor systems. It contains all the details necessary to develop and debug programs written in the 6809 assembly language for the SuperPET. Retail price is $10.95.

An additional book for COBOL will be available in the first quarter of 1982. These books may be ordered through Commodore dealers nationwide.

Link Systems Introduces DataFax™

Santa Monica, CA, October 19, 1981 — DataFax™, a new approach to information management, is now being marketed by Link Systems. This new Pascal program is designed to allow the user to enter and access information according to individual needs without programming.

DataFax™ allows the user to enter data in virtually any form and to retrieve it in an individually meaningful way...thus eliminating the constrictions of set programs. Each screen of information you enter can be cross-referenced or categorized by the keywords associated with them such as names, dates, or any categories relevant to your use. Screens may be chained together if necessary and a hard copy obtained.

The DataFax™ user enters all data via a simple screen editor... anywhere on the screen. The screen editor features cursor movement, along with tabbing and word tabbing. Data can be printed exactly as it appears on the screen or in a personalized report form.

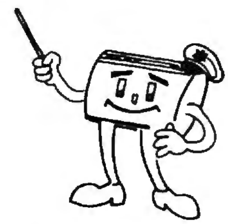
With the cross-referencing and indexing feature, a set of keywords can be entered for each screen. Keywords may be logically ANDed and ORed in any combination, thus providing a powerful data retrieval mechanism.

The program runs on the Apple II™ and will soon be available for the Apple III™. The suggested retail price is $250.

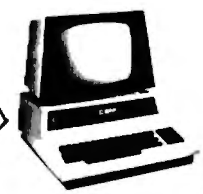
All Link software is hard disk compatible. For further information, or to order, contact Pat Merryman at (213)453-1851. Or write to Link Systems, 1655 26th St., Santa Monica, CA 90404.

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**
4032-32K 40 COL CRT	REG. $1295 **$995**
4016-16K 40 COL CRT	REG. $995 **$795**
64K ADD ON MEMORY	REG. $500 **$395**
9000-134K SUPER PET	REG. $1995 **$1795**
8050-DUAL DISK 950K	REG. $1795 **$1345**
4040-DUAL DISK 343K	REG. $1295 **$995**
2031-SINGLE DISK 170K	REG. $695 **$555**
C2N-CASSETTE DRIVE	REG. $95 **$69**
4022-80 COL PRINTER	REG. $795 **$629**
8023P-136 COL PRINTER	REG. $995 **$849**
8300P-40 CPS LETTER QUALITY	REG. $2250 **$1995**
8024-MANNSMAN TALLEY	REG. $1995 **$1595**
8024L - LETTER TALLEY	REG. $2495 **$1995**
25 CPS-STARWRITER	REG. $1895 **$1445**



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

CBM-IEEE MODEM	REG. $279 **$229**
VOICE SYNTHESIZER	REG. $395 **$329**
VIC 20 ★★★★	REG. $299 **$269**
VIC 1011 RS-232 INTER.	REG. $49.95 **$39**
VIC 1112-IEEE INTER.	REG. $99.95 **$79**
VIC 1540 - DISK 170K	REG. $599 **$499**
VIC 1515 - 30 PS PRINTER	REG. $395 **$349**
PET TO IEEE CABLE	REG. $39.95 **$34**
IEEE TO IEEE CABLE	REG. $49.95 **$39**

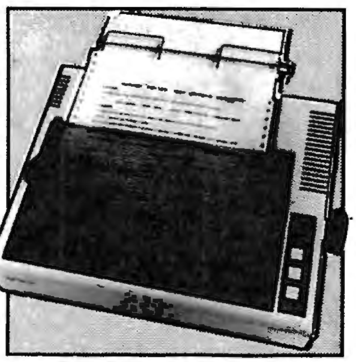
Philadelphia Computer Discount

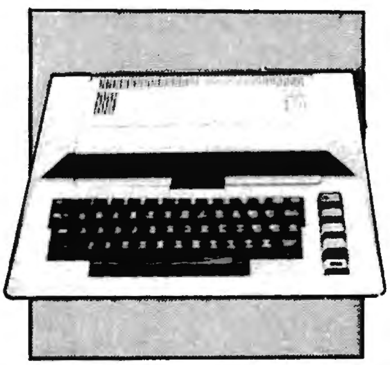
155 E. LANCASTER, PA 19087

MAIL ORDER
800-345-1289
IN PA CALL (215) 687-8540

- PREPAID ORDERS SHIPPED FREE
- VISA & MASTERCARD ADD 3%
- COD - UPS
- PA RESIDENTS ADD 6% PA TAX

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
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
NEW 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:
Wordcraft 80 Wordprocessor 295
Wordpro 4+ Wordprocessor 325
OZZ Data Base 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
CBM 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 VIC Peripherals and Software
CBM 8 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
COMFORT ZONE™
10:30
62° SA
COMPUTERIZED ENERGY MANAGEMENT
FOR THE HOME
$ 275.00





Advertisers Index

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) $1.00 billing fee

Name, Address: ________________________

Charge my: ☐ Visa ☐ MC
Number________________ Expires_____/_____/

12

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) $1.00 billing fee

Name, Address: ________________________

Charge my: ☐ Visa ☐ MC
Number________________ Expires_____/_____/

12

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 _____/_____/.

The Editor's Feedback:

My computer is:
☐ **PET** ☐ **APPLE** ☐ **ATARI** ☐ **OSI**
☐ **KIM** ☐ **SYM** ☐ **AIM** ☐ **OTHER**________
☐ **Don't yet have one...**

During the next year I expect to buy:
☐ **computer** ☐ **printer**
☐ **disk drive** ☐ **other peripherals**

Content
Best Article This Issue (page # title)

Other suggestions

12

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

Place
Stamp
Here

COMPUTE! Magazine
Post Office Box 5406
Greensboro, NC 27403

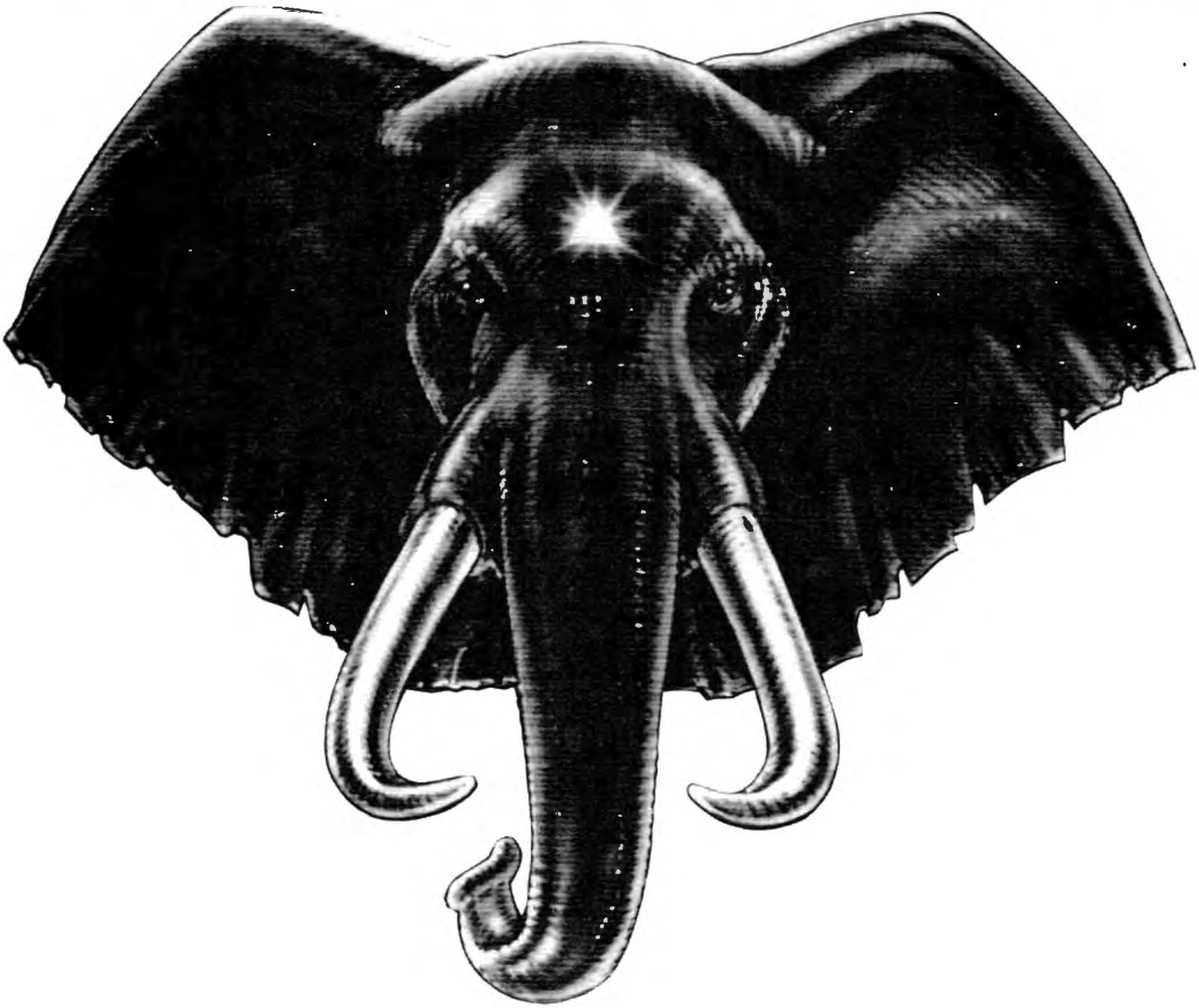
NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 236 BROOMALL, PA

POSTAGE WILL BE PAID BY ADDRESSEE

COMPUTE! Magazine
515 Abbott Drive
Broomall, PA 19008

REMEMBER.



THE COMMODORE COMPUTERS

"FROM $300 TO $1995, THEY COST LESS AND GIVE YOU MORE FOR YOUR MONEY. READ OUR CHART."

—William Shatner

The idea of a computer in every office and home used to be science fiction. Now it's becoming a reality. The question is, with so many to choose from, which computer should you buy? When you consider the facts, the clear choice is Commodore.

COMPARE OUR $995 COMPUTER

FEATURES	COMMODORE 4016	APPLE II	IBM
Base Price	$995	$1,330	$1,565
12" Green Screen	Standard	299	345
IEEE Interface	Standard	300	NO
TOTAL	$995	$1,929	$1,910
Upper & Lower Case Letters	Standard	NO	Standard
Separate Numeric Key Pad	Standard	NO	Standard
Intelligent Peripherals	Standard	NO	NO
Real Time Clock	Standard	NO	NO
Maximum 5½" Disk Capacity per Drive	500K	143K	160K

Prices are as of the most recent published price lists, September, 1981 and approximate the capabilities of the (16K) PET® 4016. Disk Drives and Printers are not included in prices. Models shown vary in their degree of expandability.

Many experts rate Commodore Computers as the best desk-top computers in their class. They provide more storage power — up to 1,000,000 characters on 5¼" dual disks — than any systems in their price range. Most come with a built-in green display screen. With comparable systems, the screen is an added expense. Our systems are more affordable. One reason: we make our own microprocessors. Many competitors use ours. And the compatibility of peripherals and basic programs lets you easily expand your system as your requirements grow. Which helps explain why Commodore is already the No. 1 desk-top computer in Europe with more than a quarter of a million computers sold worldwide.

WE WROTE THE BOOK ON SOFTWARE. The Commodore Software Encyclopedia is a comprehensive directory of over 500 programs for business, education, recreation and personal use. Pick up a copy at your local Commodore dealer.

FULL SERVICE, FULL SUPPORT. Commodore dealers throughout the country offer you prompt local service. In addition, our new national service contract with TRW provides nationwide support. Visit your Commodore dealer today for a hands-on demonstration.

Commodore Computer Systems
681 Moore Road
King of Prussia, PA 19406

Canadian Residents:
Commodore Computer Systems
3370 Pharmacy Avenue
Agincourt, Ontario, Canada, M1W 2K4

Please send me more information. CRM-CO

Name ______________________

Company ______________ Title __________

Address ______________________

City ______________ State ______ Zip ______

Telephone ______________________

Interest Area ______________________

☐ Business ☐ Education ☐ Personal

commodore
COMPUTER