

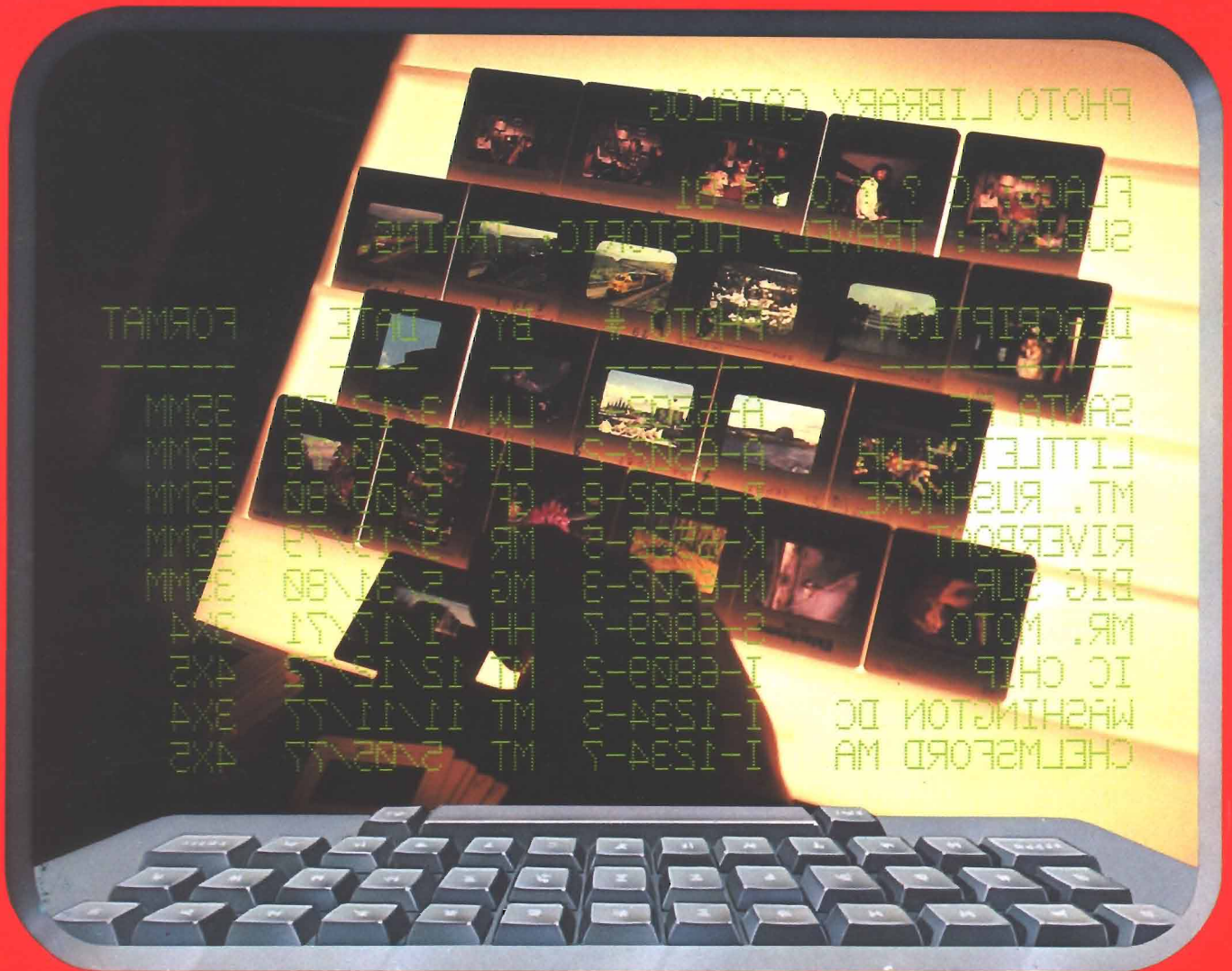
NO. 36

\$2.00

MAY 1981

MICRO™

THE 6502 JOURNAL



MacApple

How Microsoft BASIC Works

More Output from your Micro

Cursor Control for the C1P

The Atari Dulcimer

KIM/SYM Home Accounting System

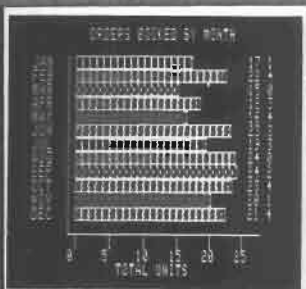
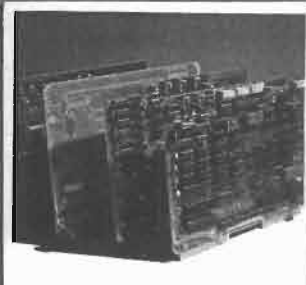
You can use MICRO PLUS™ as a

SINGLE BOARD COMPUTER

OEM BUILDING BLOCK

INTELLIGENT TERMINAL

SOPHISTICATED SYSTEM



MICRO PLUS is a 6502-based **Single Board Computer** with extensive video capabilities, communications support and keyboard interface. As an **OEM Building Block**, it allows selection of the keyboard, monitor, enclosure and power supply best suited to your application. As an **Intelligent Terminal**, it provides full RS232 and 20 mA communication at baud rates from 50 to 19.2K, with superior text editing features. It may be combined with FLEXI PLUS to form a **Sophisticated System** with 8" and 5 1/4" diskettes, an IEEE-488 controller, numerous I/O ports, up to 56K memory, and an optional 6809 microprocessor.

Video Features:

- Programmable screen format up to 132 characters by 30 lines
- Reverse video on character-by-character basis
- EPROM character set for user-definable characters
- RAM character set for dynamically changing characters under program control
- Light pen input
- Programmable character width
- Up to 4K display memory

Communications Features:

- Programmable baud rates from 50 to 19.2K baud
- Parity generation and checking
- Programmable word length and stop bits
- Full-duplex or half-duplex operation
- Both RS232C and 20-milliamp current loop interfaces provided
- ASCII keyboard interface

Monitor Features:

- Memory examine and modify
- Auto-increment mode
- Single-step
- Break at specified address
- Break on specified op code

Editor Features:

- Cursor up, down, left, right, home
- Scroll up/down
- Insert/delete line or character
- Fill/clear line or window
- Find character
- Set/clear window limits

System Features:

- Up to 7K RAM—4K display RAM, 2K programmable character generator RAM (which may be used for program RAM), 1K program RAM
- MicroMon 2 operating system software in EPROM
- Can be directly expanded with DRAM PLUS, FLEXI PLUS and PROTO PLUS
- Single voltage required +5V

Call or write for free catalog.
Let us build your custom system.

THE COMPUTERIST®
34 Chelmsford St., Chelmsford, MA 01824
617/256-3649



MICRO PLUS TCB-111	\$375
Communications option	50
Documentation	10

For US, add \$3.00 surface postage. Prices quoted are for US only. For foreign shipments write for rates. Massachusetts residents add 5% sales tax.



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.

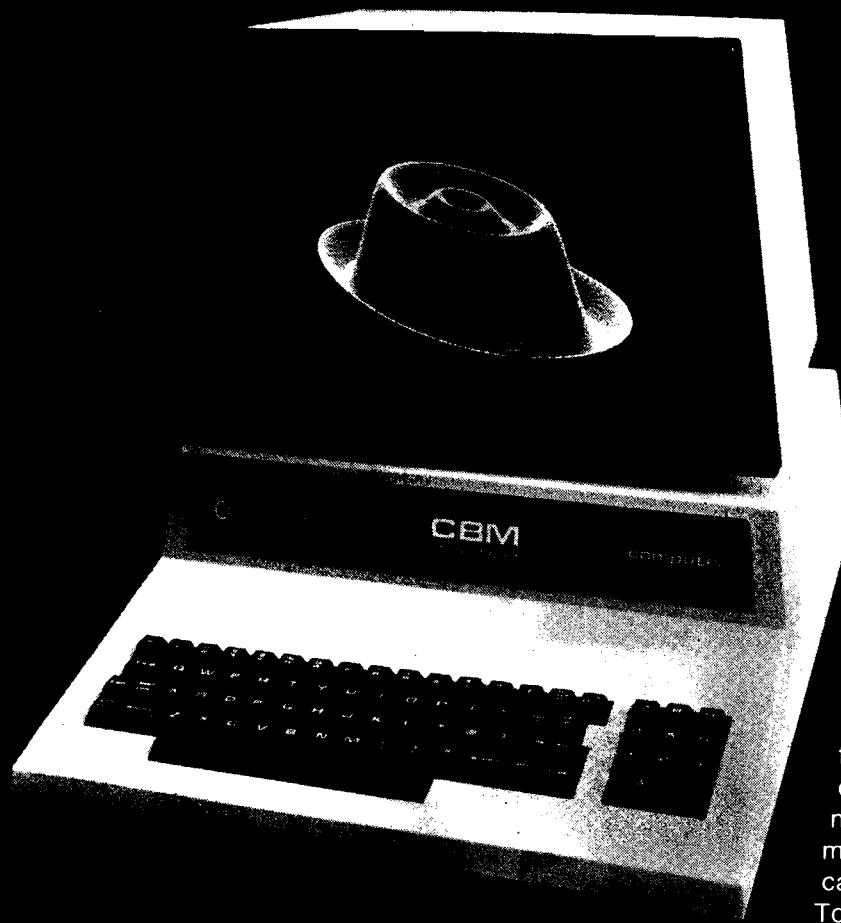
Complete information? It's at your dealer's now. Or, we'll send it to you and include a dealer list. Write us. Call us. Or, circle the reader service card number below.

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

MICROSOFT
CONSUMER PRODUCTS

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

80 COLUMN GRAPHICS



The Integrated Visible Memory for the PET has now been redesigned for the new 12" screen 80 column and forthcoming 40 column PET computers from Commodore. Like earlier MTU units, the new K-1008-43 package mounts inside the PET case for total protection. To make the power and flexibility of the 320 by 200

The image on the screen was created by the program below.

```
10 VISMEM: CLEAR
20 P=160: Q=100
30 XP=144: XR=1: Z=3.1415927
40 YP=56: YR=1: ZP=64
50 XP=XR/XP: YP=YR/YR: ZP=XR/ZP
60 FOR ZI=0 TO Q-1
70 IF ZI<ZP OR ZI>ZP GOTO 150
80 ZT=ZI*XP/ZP: Z3=ZI
90 XL=INT(.5+SQR(XP*XP-ZT*ZT))
100 FOR XI=-XL TO XL
110 XT=SQR(XI*XI-ZT*ZT)*XF: XN=XI
120 YX=(SIN(XT)+.4*SIN(3*XT))*YF
130 GOSUB 170
140 NEXT XI
150 NEXT ZI
160 STOP
170 X1=XX+Z2+P
180 Y1=YY-Z2+Q
190 GMODE 1: MOVE X1,Y1: WRPIX
200 IF Y1=0 GOTO 220
210 GMODE 2: LINE X1,Y1-1,X1,0
220 RETURN
```

bit mapped pixel graphics display easily accessible, we have designed the Keyword Graphic Program. This adds 45 graphics commands to Commodore BASIC. If you have been waiting for easy to use, high resolution graphics for your PET, isn't it time you called MTU?

K-1008-43M Manual only \$10 (credited toward purchase)
k-1008-43 Complete ready to install package \$495

Mastercharge and Visa accepted

Write or call today for our full line catalog describing all MTU 6502 products, including our high speed 8" Floppy Disk Controller for up to 4 megabytes of PET storage.



NOW 80 COLUMN PETS CAN HAVE MTU HIGH RESOLUTION GRAPHICS

MICRO™

THE 6502 JOURNAL

STAFF

Editor/Publisher
ROBERT M. TRIPP

Associate Publisher
RICHARD RETTIG

Associate Editor
MARY ANN CURTIS

Special Projects Editor
MARJORIE MORSE

Art Director
GARY W. FISH

Typesetting
EMMALYN H. BENTLEY

Advertising Manager
L. CATHERINE BLAND

Circulation Manager
CAROL A. STARK

MICRO Specialists
APPLE: FORD CAVALLARI
PET: LOREN WRIGHT
OSI: PAUL GEFFEN

Comptroller
DONNA M. TRIPP

Bookkeeper
KAY COLLINS

MICRO™ is published monthly by:
MICRO INK, Inc., Chelmsford, MA 01824
Second Class postage paid at:
Chelmsford, MA 01824 and Avon, MA
02322
USPS Publication Number: 483470
ISSN: 0271-9002

Subscription Rates:	Per Year
U.S.	\$18.00
Foreign surface mail	\$21.00
Air mail:	
Europe	\$36.00
Mexico, Central America	\$39.00
Middle East, North Africa	\$42.00
South America, Central Africa	\$51.00
South Africa, Far East,	
Australasia	\$60.00

For back issues, subscriptions, change of address or other information, write to:
MICRO
P.O. Box 6502
Chelmsford, MA 01824
or call
617/256-5515

Copyright © 1981 by MICRO INK, Inc.
All Rights Reserved

CONTENTS

- 9 **MACAPPLE**
Shorthand for commonly used Integer BASIC commands
By David Lubar
- 13 **KIM/SYM HOME ACCOUNTING SYSTEM**
Simple application requiring little hardware
By Robert Baker
- 19 **MORE OUTPUT FROM YOUR MICRO**
Add extra output bits to your AIM, SYM, KIM, Superboard or C1P
By H.H. Aumann
- 23 **APPLESOFT VARIABLE DUMP**
Handy debugging utility for Applesoft in ROM
By Scott D. Schram
- 31 **HOW MICROSOFT BASIC WORKS**
Explanation of variables and FNX definitions
By Greg Paris
- 39 **SYM-1 COMMUNICATIONS INTERFACE**
Direct messages to SYM or modem
By Nicholas J. Vrtis
- 45 **APPLE MEMORY MAPS, PART 2**
Listing and program description of memory maps
By Peter A. Cook
- 59 **THE ATARI DULCIMER**
Simulation of a 3-string dulcimer, in real time
By Mike Dougherty
- 65 **AN INEXPENSIVE WORD PROCESSOR**
Interface an IBM 2740 terminal to an 8-bit parallel port
By William F. Pytlik
- 71 **TINY PILOT FOLLOW-UP**
More information about Tiny, plus a programming example
By Nicholas J. Vrtis
- 75 **CURSOR CONTROL FOR THE C1P**
Give your C1P user-selectable windows, one-key screen clear, and the ability to edit
By Kerry V. Lourash
- 81 **PROTECTING MEMORY FROM DOS**
Protect and use RAM above DOS
By Glenn R. Sogge

DEPARTMENTS

- 5 Editorial — The Changing Scene — R.M. Tripp
- 6 Letterbox
- 17 Challenges — Paul Geffen
- 25 Microprocessors in Medicine — Jerry W. Froelich, M.D.
- 40 Annual Index
- 62 PET Vet — Loren Wright
- 72 Microbes
- 88 The MICRO Software Catalog: XXXII
- 92 6502 Bibliography: Part XXXII — William R. Dial
- 95 Advertisers' Index

DATA CAPTURE 4.0

The most advanced and easiest to use telecommunications program for use with the MICROMODEM II™ or the Apple COMMUNICATIONS CARD™

- Q. Will DATA CAPTURE 4.0 work with my Communications Card® and a modem?**
A. It makes using the Comm. Card almost as easy as using the Micromodem II.
- Q. Do I need an extra editor to prepare text for transmission to another computer?**
A. No. DATA CAPTURE 4.0 gives you control of the text buffer. You can use DATA CAPTURE 4.0 to create text.
- Q. Can I edit the text I have prepared?**
A. Yes. You can insert lines or delete any lines from the text.
- Q. How about text I have captured. Can I edit that?**
A. As easily as the text you have prepared yourself. You can delete any lines you don't want to print or save to a disk file. You can also insert lines into the text.
- Q. Just how much text can I capture with DATA CAPTURE 4.0?**
A. If the system with which you are communicating accepts a stop character, most use a Control S, you can capture an unlimited amount of text.
- Q. How does that work? And do I have to keep an eye on how much I have already captured?**
A. When the text buffer is full the stop character is output to the other system. Then DATA CAPTURE 4.0 writes what has been captured up to that point to a disk file. This is done automatically.
- Q. Then what happens?**
A. Control is returned to you and you can send the start character to the other system. This generally requires pressing any key, the RETURN key or a Control Q.
- Q. Are upper and lower case supported if I have a Lower Case Adapter?**
A. Yes. If you don't have the adapter an upper case only version is also provided on the diskette.
- Q. Do I need to have my printer card or Micromodem II® or Communications Card® in any special slot?**
A. No. All this is taken care of when you first run a short program to configure DATA CAPTURE 4.0 to your system. Then you don't have to be concerned with it again. If you move your cards around later you can reconfigure DATA CAPTURE 4.0.
- Q. Do I have to build a file on the other system to get it sent to my Apple?**
A. No. If the other system can list it you can capture it.
- Q. How easy is it to transmit text or data to another system?**
A. You can load the text or data into DATA CAPTURE 4.0 from the disk and transmit it. Or you can transmit what you have typed into DATA CAPTURE 4.0.
- Q. How can I be sure the other system receives what I send it?**
A. If the other system works in Full Duplex, it 'echoes' what you send it, then DATA CAPTURE 4.0 adjusts its sending speed to the other system and won't send the next character until it is sure the present one has been received. We call that 'Dynamic Sending Speed Adjustment'.
- Q. What if the other system works only in Half Duplex.**
A. A different sending routine is provided for use with Half Duplex systems.
- Q. What if I want to transmit a program to the other system?**
A. No problem. You make the program into a text file with a program that is provided with DATA CAPTURE 4.0, load it into DATA CAPTURE 4.0 and transmit it.

- Q. What type files can I read and save with DATA CAPTURE 4.0?**
A. Any Apple DOS sequential text file. You can create and edit EXEC files, send or receive VISICALC® data files, send or receive text files created with any editor that uses text files.
- Q. Can I leave DATA CAPTURE 4.0 running on my Apple at home and use it from another system?**
A. Yes. If you are using the Micromodem II® you can call DATA CAPTURE 4.0 from another system. This is handy if you are at work and want to transmit something to your unattended Apple at home.
- Q. Where can I buy DATA CAPTURE 4.0?**
A. Your local Apple dealer. If he doesn't have it ask him to order it. Or if you can't wait order it directly from Southeastern Software. The price is \$65.00. To order the Dan Paymar Lower Case Adapter add \$64.95 and include the serial number of your Apple.
- Q. If I order it directly how can I pay for it?**
A. We accept Master Charge, Visa or your personal check. You will get your order shipped within 3 working days of when we receive it no matter how you pay for it. Send your order to us at the address shown or call either of the numbers in this advertisement. You can call anytime of day, evening or Saturdays.
- Q. I bought DATA CAPTURE 3.0 and DATA CAPTURE 4.0 sounds so good I want this version. What do I do to upgrade?**
A. Send us your original DATA CAPTURE 3.0 diskette and documentation, the \$35.00 price difference and \$2.50 for postage and handling. We will send you DATA CAPTURE 4.0 within 3 working days of receiving your order.
- Q. What kind of support can I expect after I buy it?**
A. If you have bought from Southeastern Software in the past you know we are always ready to answer any questions about our products or how to use them.

Requires DISK II®, Applesoft II® and 48K of Memory

DATA CAPTURE 4.0©

Copyright© 1980-Southeastern Software

* Apple®, Apple II Plus®, Disk II® and APPLESOFT II® are trademarks of Apple Computer Company.

* Micromodem II® is a trademark of D.C. Hayes Associates, Inc.

* Visicalc©-Copyright by Software Arts, Inc.



We welcome your personal check. We also accept Visa and Master Charge.

Southeastern Software

Dept. MK

6414 Derbyshire Drive • New Orleans, LA 70126
504/246-8438 504/246-7937

MICRO

Editorial

The Changing Scene

With this issue, MICRO completes its fourth volume. This fact inspired me to spend some time reviewing MICRO's past, its position now, and its future.

The Past

The microcomputer world of 1977 was very different from today. The first wave of microcomputerists—the hardware types who could build a system from a kit or scratch—has started to decline in numbers and in importance. A second generation has emerged, composed of individuals with computer knowledge who are not interested in building a microcomputer. Early purchasers of the 6502 were true pioneers. There was no certainty that the new 6502 would survive in the already established 8080/6800 world. There was little vendor support for the 6502, no books, and mysteriously little material appearing about it in national computer magazines.

MICRO was started to provide a formal, regular publication with provision for quality 6502-based advertising. Early MICRO articles discussed basic problems encountered in getting systems to operate, and presented new 6502-based products. MICRO was aimed at the knowledgeable user who possessed some programming skills, but might be a novice in the microcomputer field.

The Present

Four years have witnessed the explosion of the Apple II, the addition of the AIM, SYM, Atari, OSI Superboard and Challenger systems, and the growth of the PET/CBM systems. Now thousands of programs are available. The 6502 has moved from a poor third, behind the 8080 and 6800 in the personal computing market, to a strong position ahead of both of these processors and equal to the Z80. Support for the 6502 is much broader now. There are many magazines devoted to the 6502 or one of its microcomputers; major microcomputer national magazines now offer 6502-related material on a regular basis; book shelves are well stocked with 6502 books.

The needs of today's 6502 users are changing. They are not buying a micro to get into microcomputers—they are buying micros to solve problems. Today's users are buying larger systems, and may require 80-character upper and lower case displays, quality keyboards, sophisticated disk systems, printers and more. They need ready-to-use software, and are willing to pay for it.

To serve this expanded 6502 population, MICRO has made many changes over the years, including the addition of news and idea columns. MICRO now includes articles which are less technical in nature, plus generalized material applicable to a number of microcomputers.

The Future

The microcomputer market will continue to change. Manufacturers are aiming many new products at the business market and microcomputers are now regularly advertised on the financial pages of major newspapers and are featured in radio promotions.

The new Apple III and CBM products are definitely for the businessman, not the "hacker." These business users will require different levels of support than the current users.

Another group of users emerging is the consumers—the home market. The Atari, VIC, and Intellivision are based on pre-programmed packages which require no user modification or programming. Anyone can use them, instantly!

We have some ideas which will be implemented in MICRO over the coming months. These include "bonus" sections providing focused coverage of particular topics such as graphics, programming languages, games, printers, disk systems, art, business, education, and expanded coverage of the Apple, PET/CBM, and other systems. We are planning a MICROScan section which will provide a systematic evaluation of products within an area. We expect to cover the new microprocessors which may gradually supplant the 6502; as the processors change, our readers will be kept informed.

I am sure there are many other areas in which MICRO can help serve its readers. The staff of MICRO is very interested in hearing from you. Please write and let us know about your interests, how your use of the microcomputer is changing, and how we at MICRO can continue to support your efforts.

Robert M. Tripp
Editor/Publisher

About the Cover

PHOTO LIBRARY CATALOG

FLAGS: C ? ? 0 78 81
SUBJECT: TRAVEL, HISTORIC, TRAINS.

DESCRIPTION	PHOTO #	BY	DATE	FORMAT
SANTA FE	A-6502-1	LW	3/12/79	35MM
LITTLETON NH	A-6502-2	LW	8/28/78	35MM
MT. RUSHMORE	B-6502-8	GF	5/09/80	35MM
RIVERBOAT	K-6502-5	MR	9/19/79	35MM
BIG SUR	M-6502-3	MG	5/31/80	35MM
MR. PHOTO	S-6809-7	HH	1/17/71	3X4
IC CHIP	I-6809-2	MT	12/12/72	4X5
WASHINGTON DC	I-1234-5	MT	11/11/77	3X4
CHELMSFORD MA	I-1234-7	MT	5/05/77	4X5

Information Retrieval

The cover depicts an information retrieval application in which a photographer with a collection of many photographs needs to select a subset of particular categories. These could include black and white or color; a slide, negative, print or other; where it had been previously published; etc. Categories dealing with subject matter could be broken down to include indoor/outdoor, people/scenic, day/night, and others. A data base would consist of individual records with FLAGS and a portion which would contain other information about the photo for sorting purposes. For example, the photographer could request photos which are in color (C), scenic (S), outdoor (O), and so forth, skipping

categories which he does not wish to select by entering a question mark. He could further select the fields of each record by specifying key words which are to be matched once a record has passed the basic FLAG tests. The tests can be combined and can be as complex as necessary.

Does this all sound very difficult? Not at all. A system with all of these features, and more, was implemented on a KIM-1 with 1K of RAM. It provided six tests on the FLAGS and one test on each of six data fields in the record. It provided up to 900 entries on a single 30-minute cassette tape. The information retrieval process can be applied to almost any data base.

(Photo by Loren Wright.)

MICRO

Letterbox

MICRO's February editorial, "Too Many Apples!" brought us a flood of responses. Here are just a few of the letters offering comments and suggestions on our Apple backlog problem.

Dear Editor:

In regard to the editorial "Too Many Apples!" my solicited comment as an avid reader of your magazine is this: as an amateur futurologist, I predict your editorial of February 1983 will be entitled: "Too Many PETs!"

It appears to me possible that the \$299 VIC 20 by Commodore may have sold well over a million units by that date and you will have an unwieldy excess of good articles on this machine!

George Earl
1302 South General McMullen
San Antonio, Texas 78237

Dear Editor:

I have been reading and enjoying MICRO for several years, and have all the issues since the beginning. I have seen the magazine grow in size and quality, and consider it my favorite of several magazines I read regularly.

This letter is in response to your editorial, "Too Many Apples!" My first microcomputer was an AIM-65, and I enjoyed reading MICRO, because it didn't ignore the board level computer. Now I have an Apple II, and I can appreciate your concern about giving equal coverage to all the 6502-based systems.

Of the six options suggested in the editorial, allocating a larger portion of MICRO to Apple, I believe, is fair since it is the most popular 6502-based computer, and by your own admission, has the most articles available. The addition of 16 to 32 extra pages is something which is inevitable, if the past growth of MICRO is any indication.

I think the Apple is the best 6502-based micro on the market, (that's why I bought it), and I think it is natural for it to receive extraordinary coverage. Also, I still enjoy, and learn, from the hardware articles which appear in MICRO (such as Marvin DeJong's article on the 6522), and if you were to publish a separate magazine only on the Apple, I probably would subscribe to it, but would drop MICRO, and would miss the 'hardware stuff.' Since MICRO is the 6502 journal, it would be a shame to divide it into a lesser pair of magazines.

Keep up the good work on MICRO, and don't be afraid to 'overload' with Apple stuff—there are a lot of Apples out here!

Edward Janeczek
6121 Carnation Road
Dayton, Ohio 45449

Dear Editor:

I have been an avid reader of MICRO since Issue #7, and it has never been better than it is today. I'd like to congratulate you on the vastly improved appearance of the magazine. The typography is far better than it used to be before last December.

I own an Ohio Scientific C1P, and I was interested in your February editorial about your surfeit of Apple articles. One of the reasons I like MICRO so well is that there are a number of articles every month that I can use with my own computer. I would hate to see MICRO become devoted entirely to the members of the Apple corps. Still, I have always felt more of a kinship to the Apple and KIM owners than to the PET owners, who seem to dominate other magazines.

So what should MICRO do? Hardware is interesting, particularly general purpose "how I connected a DAC to my 6522" material. General short 6502 software ["how to convert ASCII to EBCDIC in seven bytes of code"] is usually interesting, but you shouldn't include any listings that are more than a page. Avoid "POKE 67 into location \$E5 on your Apple and see what happens!" articles. Don't assume that everyone has dual disk drives and a Diablo printer. Avoid large turn-key type software for specific systems. [Nobody out here really cares about small business software, you know. Some people think that computers—

small computers, that is—should be useful for *something*, and we really ought to be able to help the small businessman drop \$5,000 or \$10,000 on small computer hardware and a like amount on software.] What we want to read about is systems software, and FORTH, and UNIX, and C, and bubble memories, and Winchester disk drives that cost less than \$500, and color graphics systems, and music synthesizers, and Dragons and Dungeons, and Ethernet, and good text editors, and material like that.

John P. Sohl
20446 Orey Place
Canoga Park, California 91306

Dear Editor:

You backed me into a corner. When I looked toward the right, I saw the excitement to renew for another year at the \$15 rate. When I looked toward the left, I saw an ever-expanding Apple orchard.

I'm a single board man. I have a KIM and an AIM. I enjoy that level of computing. I'm not ignorant of the capabilities of a larger system—I also have a TRS-80 Level II. I get my satisfaction out of making a \$200 KIM do the same things a \$2000 Apple can do [almost!].

In your editorial, you asked for opinions. I look to MICRO for ASK articles. I've been with you since Issue #1. I've seen the larger systems come down in price and increase in popularity. The days of every computer hobbyist knowing what a KIM is are gone.

Those days are gone, but your readers, like myself, are not. I don't ask for 100% ASK articles—that would be unrealistic. What I do ask is that you keep the same carefully planned balance that you struck in Issue 33. If you find yourself overloaded with excellent Apple articles, by all means, publish them. But, go with the Apple supplement idea—that looks best to me. This assumes that others, like myself, will continue to keep you fed with good articles on the various other systems. If that fails, you have no choice other than to become the "MICROApple."

Jody Nelis
132 Autumn Drive
Trafford, Pennsylvania 15085

(Continued on page 16)

MR. RAINBOW

presents our valuable free catalog (over 100 pages). He **PROMPTS** you to **PEEK** at the latest collection of software and hardware products for your **APPLE II™**



A STELLAR TREK

the definitive Hi-Res color version of the classic Startrek game. Three different Klingon opponents. Many command prerogatives from use of weapons to repair of damages. Needs 48K Applesoft ROM.

Disk... **\$24.95**

VERSAWRITER II

A drawing tablet, simply plugs into your game I/O port. Trace, draw, design, or color any type of graphic. Adds words to pictures. Creates schematics. Computes Distance/Area of any figure. New - fill any area on the screen in seconds with over 100 different and distinct colors. Needs 32K Applesoft ROM and disk drive. A bargain at...

\$249.95

BOWLING DATA SYSTEM

This data management program provides accurate record keeping and report generation for bowling leagues of up to 40 teams with 6 bowlers per team. Needs 80-column printer, 32K Applesoft ROM.

Disk... **\$79.95**

SUPER SOUND

Musical rhythms, gunshots, sirens, laser blasts, explosions... add these and many more exciting sounds to your Apple. Use them in your programs, or create your own SUPER SOUNDS. Needs 16K Applesoft.

Have a blast for only

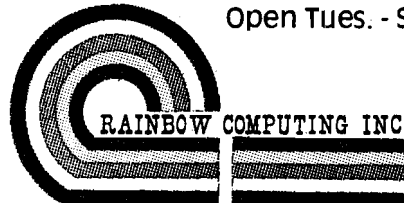
\$12.95... Tape
\$16.95... Disk

ADD \$2.00 U.S. \$10.00 FOREIGN FOR SHIPPING
CALIFORNIA RESIDENTS ADD 6% SALES TAX

Don't see what you want here, then write or call today for your free catalog. We're saving one just for you.

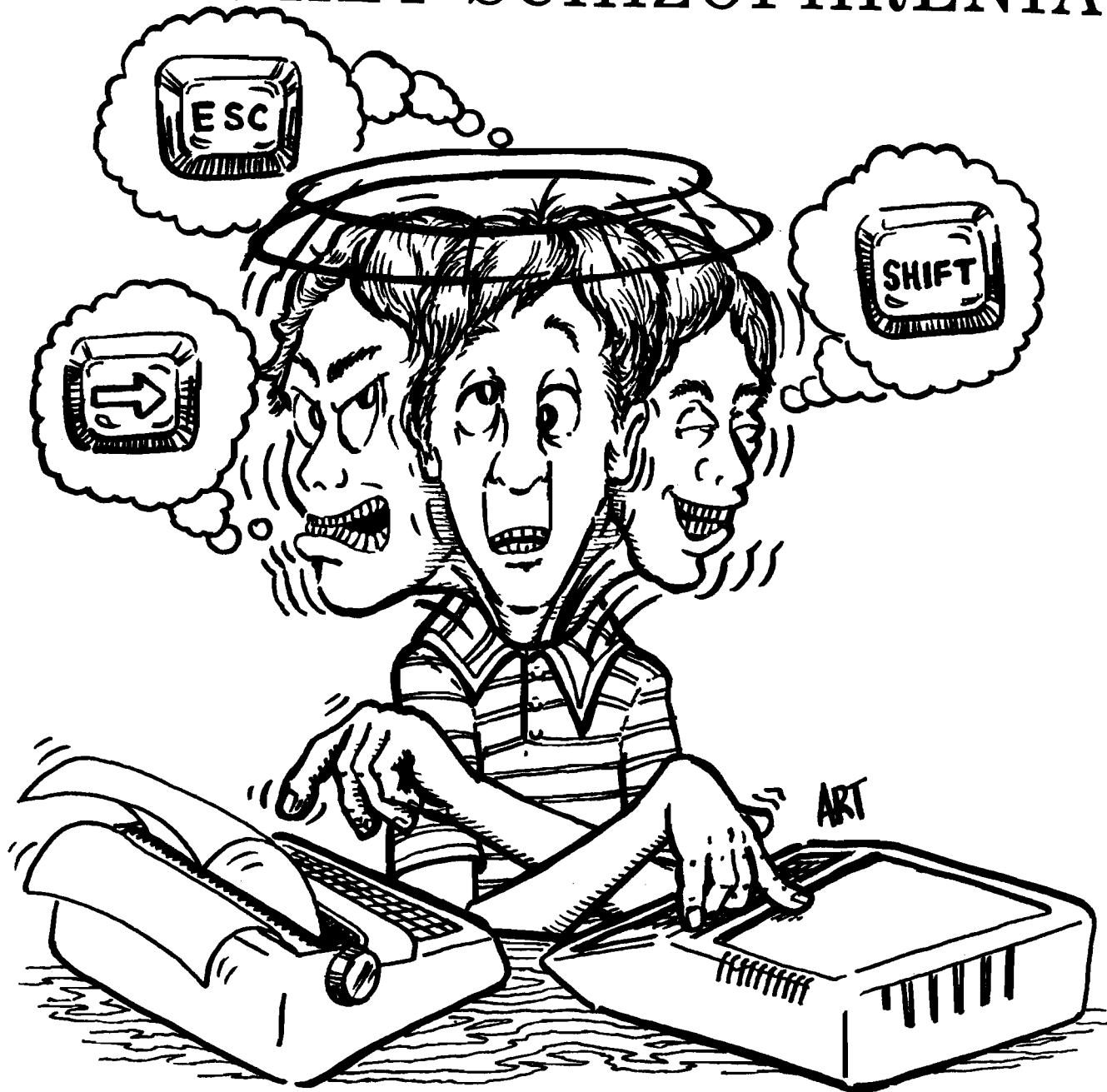
Visa / Mastercharge welcome.

Open Tues. - Sun.



GARDEN PLAZA SHOPPING CENTER
9719 RESEDA BOULEVARD DEPT. 1M1
NORTHRIDGE, CALIFORNIA 91324
PHONE (213) 349-0300

ARE YOU DEVELOPING SHIFT KEY SCHIZOPHRENIA?



Cure it with the lower case system from



The Keyboard +Plus from Lazer MicroSystems turns your Apple's shift key into a . . . shift key! The Keyboard +Plus transforms your Apple's upper case only keyboard into a typewriter style keyboard capable of entering all 128 ASCII characters into your DOS, Pascal, and CP/M applications programs. The use of the shift key is automatic with most programs. You do not have to write complicated "driver programs" or fuss with obscure "BIOS gatches" in order to fully utilize this board. For those situations where upper case only input is desired, the Keyboard +Plus supports a caps lock mode that returns the Apple keyboard to it's former state.

Best of all, the keyboard +Plus features a typeahead buffer that gives you the ability to continue typing even though the computer is busy performing other tasks such as accessing the disk.

The keyboard +Plus is the second component of Lazer MicroSystems' lower case system. When teamed with the Lazer MicroSystems' highly praised Lower Case +Plus, the lower case system turns your Apple into a sophisticated, user oriented, problem solving machine.

See the Lazer MicroSystems' lower case system at your local Apple dealer. If he is all out, you can order direct from us.

P.O.Box 55518
Riverside, Calif. 92517
(714) 682-5268

* Keyboard +Plus \$119.95

* Lower Case +Plus \$69.95

- * Calif. residents add 6% tax.
- * Outside U.S.A. add \$15.00 for Shipping & extra handling
- * Allow 2 weeks extra for checks to clear. (personal & business)
- * MC/VISA accepted. Include card number, exp date, and signature.

Lower Case +Plus, Keyboard +Plus and +Plus are all trademarks of Lazer MicroSystems Inc.

MacApple

This routine allows substitution of unreserved control keys as shorthand for commonly used Integer BASIC commands. Since it is table driven, extension to Applesoft or machine language is possible.

David Lubar
249 Loring Ave., Apt. 3
Edison, New Jersey 08817

The program "Applesoft Shorthand" (23:5) was impressive. Here was a way to shorten those long hours spent thrusting two fingers at the keyboard. I had several long programs to enter the other night, but they were in Integer BASIC. This left two choices; wait until MICRO published what I needed, or write it myself. I took the second option. The program (minus the one bug which kept me up until 6:30 a.m. on a bleary Sunday) is described in the following article. Since I didn't have the locations of Integer's keywords, I took another approach, making a table from which the keywords are printed. While this lengthens the program, it also gives the program multilingual potential—by changing the table, you can use it with Applesoft or even assembly language.

Using MacApple

MacApple loads from \$1000.\$1131. The table goes up to \$10CF followed by a \$62 byte program. Turn it on with a CALL 4383, off with a CALL 4393. Once MacApple is turned on, any control key which is not reserved will produce a keyword. Using the \$1000 area allows the program to lie between a BASIC program and the variable table. This way, you can leave it in memory while entering and modifying BASIC programs.

How It Works

The table contains the ASCII values of the keywords. The end of an entry is signalled with a null character (\$80). With a range of 26 letters, eight characters per letter seemed like a good amount of storage. The CALL 4383 changes the pointers at \$38,\$39, causing the monitor to go to MacApple instead of the normal KEYIN routine. At the start of MacApple, the KEYIN routine is duplicated in part, without the portion that increments the random number. If the ASCII value of the character entered is less than \$9B, it is a control character. Other characters are sent back to KEYIN at the point where the strobe is reset. From there, they follow the normal path into the buffer and onto the screen.

For control characters, a check has to be made. Certain of these characters should be left alone. For example, control-M is the carriage return. This, obviously, is needed. The front and

back arrows, controls U and H, were also left alone. The other reserved control characters, B, C, D, and X, aren't essential, but I left them alone, giving the user the option to do as he wishes. Once you're in BASIC, control-B isn't needed. Control-C can be replaced with a JSR from the monitor. Its other function, stopping a program, can be replaced with a brute-force RESET, though you lose the ability to see where the program stopped. Control-D is left free for disk users. Instead of control-X, you can cancel a line by adding a syntax error.

You can check for these reserved characters by the series of CMP's and BEQ's. The checks are written in ascending order. The program can be speeded up [for those of you who can count microseconds] by placing the most common ones (controls M, H, and U) at the top of the series.

Once an input passes this far, the heart of the program goes into action. First, the ASCII value is reduced from a

Table 1

Control Key	Result	Control Key	Result
A	ASC{"	N	NEXT
B	reserved	O	COLOR ⁵
C	reserved	P	PRINT
D	reserved	Q	PLOT
E	PEEK	R	RETURN
F	POKE	S	SCRN{
G	GOTO	T	THEN
H	reserved	U	reserved
I	INPUT	V	VLIN
J	GOSUB	W	HLIN
K	CALL	X	reserved
L	LEN{	Y	REM
M	reserved	Z	DIM

Note: Though not all keywords could be paired with their initial letter, an attempt was made to produce a meaningful relation. For example, to remember that J produces GOSUB, just think of GOSUB as JSR. Pairs were placed together when possible (VLIN, HLIN and PEEK, POKE).

range of \$81-\$9A to a range of \$00-\$19. Next, this value is multiplied by 8 with three ASL's. These steps result in a pointer to the character table. The pointer is put into the Y register. The A register is loaded with DATA, Y, getting the first character for the desired keyword. This character is compared to \$80. If it isn't \$80, the character is stored in the input buffer and sent to the screen through the COUT1 routine in the monitor [FDF0].

Note: No check is made to see if the buffer has been filled. If a keyword puts the buffer count too high, it will do a hatchet job on the line and begin filling the buffer from the start. Just keep this limitation in mind and there will be no problems.

After this, Y is incremented to point to the next character in the table, and X is incremented to point to the next location in the buffer. Once the keyword has been output, there is an \$80 in the A register. It might seem that there would be no harm in sending this null character out. In most cases, this is true. But it could cause problems. For example, the ASC function returns the value of the first character after the quote. If \$80 is sent out, it won't be on the screen, but it will become the argument for the ASC function. No matter what letter follows, BASIC will return a decimal value of 128. To avoid this, the routine clears the strobe, resets the cursor, and goes back for the next input. If the cursor isn't reset with LDY 24, strange things happen. Try deleting this command. Then, in BASIC with MacApple turned on, enter the control keys for PRINT followed by ASC["]. (Control-P, control-A.) The next key entered will cause the T in PRINT to turn into a @.

The Table

Changing the keywords, either for Integer or for other languages, is simple. You can either step through the monitor or go directly to an entry. To step through the monitor, enter FFF and hit RETURN. Hitting RETURN again will cause a list of bytes \$1000-\$1007. These locations contain the keyword printed by control-A. Each RETURN will advance to the next letter, up to \$10C8-\$10CF, which is the location for control-Z. If you don't feel like stepping through the monitor, use the following method. Take the letter

```

0800 ;*****
0800 ;*
0800 ;* MACAPPLE *
0800 ;* BY DAVID LUBAR *
0800 ;*
0800 ;*****
0800 ;*
0800 ;*
0800 CH EPZ $24
0800 BASL EPZ $28
0800 KSWL EPZ $38
0800 KSWH EPZ $39
0800 IN EQU $200
0800 DATA EQU $1000
0800 KBD EQU $C000
0800 STROBE EQU $C010
0800 KEY1 EQU $FD2B
0800 COUT1 EQU $FDF0
0800 ;
0800 ;
10D0 ; ORG $10D0
10D0 ; OBJ $800
10D0 ;
10D0 ;DATA GO FROM $1000-$10CF
10D0 ;
10D0 ;
10D0 2C00C0 START BIT KBD ;KEY DOWN?
10D3 10FB BPL START ;NO
10D5 9128 STA (BASL),Y ;YES. REPLACE CURSOR
10D7 AD00C0 LDA KBD ;GET CHARACTER
10DA C99B CMP #$9B ;CONTROL CHARACTER?
10DC 9003 BCC MAIN ;YES
10DE 4C2BFD BACK JMP KEY1 ;NO. OUTPUT IT
10E1 C982 MAIN CMP #$82 ;CONTROL-B?
10E3 F0F9 BEQ BACK ;YES
10E5 C983 CMP #$83 ;CONTROL-C?
10E7 F0F5 BEQ BACK
10E9 C984 CMP #$84 ;CONTROL-D?
10EB F0F1 BEQ BACK
10ED C988 CMP #$88 ;CONTROL-H?
10EF F0ED BEQ BACK
10F1 C98D CMP #$8D ;CONTROL-M?
10F3 F0E9 BEQ BACK
10F5 C995 CMP #$95 ;CONTROL-U?
10F7 F0E5 BEQ BACK
10F9 C998 CMP #$98 ;CONTROL-X?
10FB F0E1 BEQ BACK
10FD 38 SEC ;REDUCE VALUE TO A RANGE
10FE E981 SBC #$81 ; OF $00-$19
1100 0A ASL ;MULTIPLY BY 8
1101 0A ASL
1102 0A ASL
1103 A8 TAY
1104 B90010 LOOP LDA DATA,Y ;GET TABLE ENTRY
1107 C980 CMP #$80 ;END OF ENTRY
1109 F00B BEQ BACK1 ;YES
110B 9D0002 STA IN,X ;NO. PUT CHARACTER IN BUFFER
110E 20F0FD JSR COUT1 ;PRINT CHARACTER
1111 E8 INX ;INC BUFFER POINTER
1112 C8 INY ;INC TABLE POINTER
1113 4C0411 JMP LOOP ;DO IT AGAIN
1116 2C10C0 BACK1 BIT STROBE ;CLEAR KEYBOARD STROBE
1119 A424 LDY CH ;RESET CURSOR VALUE
111B 4CD010 JMP START ;CURE FOR DEAD BATTERIES?
111E EA NOP ;EXTRA BYTE SO CALL FROM BASIC
111F ;WILL BE AN EASY NUMBER
111F ;TO REMEMBER
111F ;*
111F ;*
111F ;CALLS FROM BASIC ENTER HERE
111F ;*
111F A9D0 ON LDA #START ;SET VALUES FOR INDIRECT JUMP
1121 8538 STA KSWL
1123 A910 LDA #START
1125 8539 STA KSWH
1127 60 RTS
1128 EA NOP ;ANOTHER FILLER BYTE
1129 A91B OFF LDA #$1B
112B 8538 STA KSWL
112D A9FD LDA #$FD
112F 8539 STA KSWH
1131 60 RTS

```


you want and subtract 1 from its location in the alphabet. Then multiply this by 8. Add this, in hex, to \$1000. [Congratulations, you have just performed a machine-language subroutine in your head.] That value gives the start of the table for the desired letter.

Once you've found the starting point, enter the ASCII values for the desired keyword, followed by an \$80. If you don't have an ASCII table, use the ASC function from BASIC, then convert the number to hex. For those of you who are lazy, I've included an Integer BASIC program which constructs keyword tables in listing 1.

While the op codes for assembly language are only three letters long, you could save some typing by putting together a table which included the leading and trailing spaces and other special characters. For example, [space]LDA[space]#, for immediate commands could be printed with one control character.

Modifications

Relocating the program isn't difficult. Only a few changes are needed. The JMP LOOP and JMP START are the only jumps which refer to the program. The value of DATA would have to be changed, as would the values set by the ON portion.

Final Notes

The pointers to the KEYIN routine cannot be reset in the direct mode. Suppose, for example, you change the lo byte with POKE 56,NN. So far, there is no problem. But as soon as you hit RETURN, the monitor will go to the input routine. When it hits the indirect jump to KEYIN, it will find a value with a new lo byte and an old hi byte. Unless you are incredibly lucky, this new value will not be one which has anything to do with input. To see this in action, enter POKE 56,7. [You can reset the pointers with a line from BASIC since the monitor won't look for input during execution.]

Disk users will have to add a CALL 1002 after turning MacApple on or off.

Finally, if you use the front or back arrow immediately after a keyword, a @ will appear on the screen. This can be removed with the space bar.

I hope this program will save you some time and effort.

Table 2: Keyword table.

A	1000	--	C1	D3	C3	A8	A2	80	50	50
B	1008	--	80	50	50	50	50	50	50	50
C	1010	--	80	50	50	50	50	50	50	50
D	1018	--	80	50	50	50	50	50	50	50
E	1020	--	D0	C5	C5	CB	80	50	50	50
F	1028	--	D0	CF	CB	C5	80	50	50	50
G	1030	--	C7	CF	D4	CF	80	50	50	50
H	1038	--	80	50	50	50	50	50	50	50
I	1040	--	C9	CE	D0	D5	D4	80	50	50
J	1048	--	C7	CF	D3	D5	C2	80	50	50
K	1050	--	C3	C1	CC	CC	80	50	50	50
L	1058	--	CC	C5	CE	A8	80	50	50	50
M	1060	--	80	50	50	50	50	50	50	50
N	1068	--	CE	C5	D8	D4	80	50	50	50
O	1070	--	C3	CF	CC	CF	D2	BD	80	50
P	1078	--	D0	D2	C9	CE	D4	80	50	50
Q	1080	--	D0	CC	CF	D4	80	50	50	50
R	1088	--	D2	C5	D4	D5	D2	CE	80	50
S	1090	--	D3	C3	D2	CE	A8	80	50	50
T	1098	--	D4	C8	C5	CE	80	50	50	50
U	10A0	--	80	50	50	50	50	50	50	50
V	10A8	--	D6	CC	C9	CE	80	50	50	50
W	10B0	--	C8	CC	C9	CE	80	50	50	50
X	10B8	--	80	50	50	50	50	50	50	50
Y	10C0	--	D2	C5	CD	80	50	50	50	50
Z	10C8	--	C4	C9	CD	80	50	50	50	50

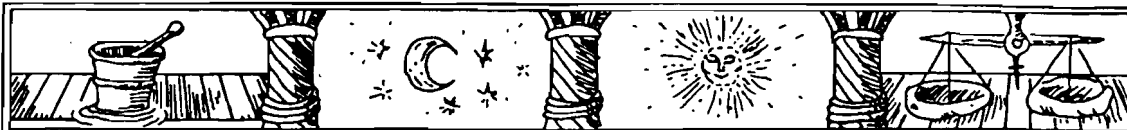
```

10 DIM A$(26),B$(10)
20 A$="ABCDEFGHIJKLMNOPQRSTUVWXYZ"
30 FOR I=0 TO 25
40 PRINT "ENTER KEYWORD FOR CONTROL ";A$(I+1,I+1)
50 INPUT B$
60 IF LEN(B$)>7 THEN 40
70 FOR J=0 TO LEN(B$)-1
80 IF LEN(B$)<1 THEN 110
90 POKE 4096+8*I+J, ASC(B$(J+1,J+1))
100 NEXT J
110 POKE 4096+8*I+J,128
120 NEXT I: PRINT "DONE": END

```

MICRO

DDJ



DR. DOBB'S JOURNAL of COMPUTER Calisthenics & Orthodontia

Running Light Without Overbyte

Twelve Times Per Year

\$21/1 Year — \$39/2 Years

Recent issues have included:

ZX65: Simulating a Micro

EXOS-6500 Software Development Tool Kit

6502 Assembler—Pet 8K-32K

A Note on 6502 Indirect Addressing

The C Programming Language

What you see is what you get.

To subscribe, send your name and address to *Dr. Dobb's Journal*,
Department V4, Post Office Box E, Menlo Park, CA 94025.
We'll bill you.



PCG

KIM/SYM

Home Accounting System

This program illustrates a very simple and basic application for a personal computer in the home that requires a minimum of hardware to implement.

Robert Baker
15 Windsor Drive
Atco, New Jersey 08004

This article was originally written for the KIM but will also run on the SYM with the included routines.

After acquiring a KIM-1 micro-computer, this simple program was written to help justify its existence in our home. The program was designed to do the bookkeeping for our family budget but could easily be used for many other applications.

My goal was to write a program that would not require any additional hardware besides the very basic system containing a KIM-1 module, power supply, and cassette recorder. Thus, the program uses the on-board keyboard for all input, and the 7-segment displays for all output by means of two of the monitor sub-routines in ROM. The positioning of the 7-segment displays makes them ideal for displaying monetary values with a small space between the "dollars" digits and the "cents" digits, but the program can also be used for various other applications, such as a parts inventory.

The program itself resides in page 2 (Loc. 0200-02FF) of RAM and uses the first locations of page zero for working storage as shown in the program listing. Page 3 (Loc. 0300-03FF) of RAM is used to store the balances of each account with three bytes per account. The first three locations of page 3 are reserved for account #0 which is the overall total of all existing accounts. The remaining space of page 3 may be

Table 1: Keyboard Commands (# = any decimal number 0-9)

Input	Operation/Display
##A	Set account number '##' Display reads 'AAAA ##'
{no input} A	Display current account number. Display reads 'AAAA ##'
{no input} +	Increment account number & display new number. If at last account "wrap" to account #0. Display reads 'AAAA ##'
B {any time}	Display balance of current account. Display reads '#### ##'
#### #C	Credit current account & display new balance. Display reads '#### ##'
#### #D	Debit current account & display new balance. Display reads '#### ##'
	If current balance is less than amount to be deducted, item will be disregarded and display will read 'EEEE EE'

```

0800      ;*****
0800      ;*
0800      ;*   KIM + 1 = ?   *
0800      ;*
0800      ;* BY ROBERT BAKER *
0800      ;*
0800      ;*****
0800      ;*
0800      ;*
0800      INH   EPZ $F9      ;LSD 7-SEGMENT DISPLAY
0800      POINTL EPZ $FA     ;MIDDLE 2 DIGITS IN DISPLAY
0800      POINTH EPZ $FB     ;MSD IN DISPLAY
0800      SCANS  EQU $1F1F   ;SCAN DIGITS DISPLAY & LOOK FOR INPUT
0800      GETKEY EQU $1F6A   ;READ KEYBOARD INPUT
0800      ;
0800      PTR   EPZ $00      ;TABLE POINTER
0800      ACCT  EPZ $01      ;CURRENT ACCT #
0800      INFLG EPZ $02      ;INPUT FLAG
0800      WORK  EPZ $03      ;WORKING STORAGE
0800      ;
0200      ;
0200      ORG  $200
0200      ;
0200      OBJ  $800
0200 A900  START LEA $500    ;INIT ACCT# = 0
0202 85F9  SETA  STA INH    ;SFT ACCT#
0204 8501  SFTPTF STA ACCT
0206 29F0  SFTPTF AND $5F0  ;SFT POINTER -
0208 4A    LSR             ;CONVERT ACCT#
0209 4A    LSR             ; TO BINARY &
020A 8503  STA WORK       ;MULTIPLY BY 3
020C 4A    LSR
020D 8500  STA PTR
020F A501  LDA ACCT
0211 38    SFC
0212 E503  SPC WORK
0214 F500  SFC PTR

```

used as required for up to 84 individual accounts. The maximum number of accounts is determined by the value in location 02BA, which should be one greater (decimal) than the highest account number desired.

After hand-loading the program the first time, be sure to clear all locations of page 3 that are to be used for storage of the account balances (Loc. 0300 to 0302 + (3 * #accounts)). Also, don't forget to set the highest account number plus one in location 02BA. The locations on page zero are initialized by the program so there is no need to set [or save] these.

When you are ready to run, load address 0200 and depress "GO". The display should read 'AAAA 00' to indicate proper initialization with the current account number equated to zero. You're now ready to use the program as desired. Table 1 gives a complete description of each of the keyboard controls; keys 0-9 are used for input values and keys A,B,C,D, and + are used for control.

Each time a value is added to (credit) or subtract from (debit) an individual account, it is also added/subtracted to account #0 to keep a running total of all account balances. This provides a simple method of comparing your checking and savings accounts with your budget balance. To keep account #0 valid, the program will not allow you to credit/debit account #0 directly. Also, if you try to debit an account with an amount greater than its current balance, the entry will be disregarded and the display will read 'EEEE EE' to indicate the error.

After each session of running the program simply store the program on cassette following the standard procedures, locations 0200 to (0302 + (3 * #accounts)). This will save the current account balances plus the program itself. Then, the next time you want to run the program, simply load from cassette and start at location 0200. Alternately, you can save pages 2 and 3 separately to conserve space on cassette, or to allow more program flexibility for specific applications.

```

0216 8500          STA PTR
0218 0A           ASL
0219 6500          ADC PTR
021B 8500          STA PTR
021D A9AA          LFA #SAA          ;DISPLAY A'S
021F 85FA          STA POINTL      ;LOAD DISPLAY SPECIAL CHARACTERS
0221 85FB          STA POINTH
0223 A901          CLFLG LFA #S01      ;CLEAR INPUT FLAG
0225 8502          STA INFLG
0227 201F1F        DSPLY JSR SCANES      ;DISPLAY DATA
022A              ;FOR SYM, SUBSTITUTE CODF AT $106
022A DOFB          BNE DSPLY
022C 201F1F        INPT JSR SCANES      ;WAIT FOR INPUT
022F              ;FOR SYM, SUBSTITUTE CODF AT $106
022F FOFB          BEQ INPT
0231 206A1F        JSR GETKEY      ;READ KEY
0234              ;FOR SYM, SUBSTITUTE CODF AT $133
0234 C90A          CMP #S0A          ;DIGIT?
0236 1018          BPL CNL      ;BRANCH IF CONTROL KEY
0238 A004          LTY #S04          ;SET NORMAL SHIFT COUNT
023A C602          DEC INFLG      ;FIRST INPUT?
023C D002          BNE SHFT
023E A018          LDY #S18          ;YES, SET SHIFT COUNT TO CLEAR DISPLAY
0240 06F9          SHFT ASL INH      ;SHIFT DIGITS
0242 26FA          RCL POINTL
0244 26FB          RCL POINTH
0246 8E           DFY
0247 D0F7          BNF SHFT
0249 45F9          EOR INH          ;ADD NEW DIGIT
024B 85F9          STA INH          ; TO DISPLAY REGISTER
024D 4C2702        JMP DSPLY          ;GET NEXT KEY
0250 C90B          CNL CMP #S0B          ;WANT BALANCE?
0252 D00F          ENF INCHK
0254 A400          BAL LDY PTR          ;YES, GET POINTER
0256 A202          LTX #S02
0258 B90203        MBAL LDA TELH,Y      ;MOVE BALANCE
025B 95F9          STA INH,X      ; TO DISPLAY
025C 88           DEY
025E CA           DEX
025F 10F7          BPL MBAL
0261 D0C0          LINK BNE CLFLG      ;CLEAR FLAG & WAIT
0263 C602          INCHK DEC INFLG      ;ANY INPUT?
0265 F03E          BEQ GETA
0267 C90A          CMP #S0A          ;YES, NEW ACCT#
0269 D004          BNE CRT
026B A5F9          LDA INH          ;YES, GET NEW #
026D 104A          BPL CHKA      ;CHECK IT
026F A000          CRDT LTY #S00
0271 C400          CPY PTR          ;ACCT #0?
0273 F0B2          BFQ DSPLY      ;ERROR, CANNOT CR/DB ACCT #0
0275 C90C          CMP #S0C          ;CREDIT ACCOUNT?
0277 D00A          BNE DBT
0279 20C002        JSR ADP          ;YES, ADD TO ACCOUNT 0
027C A400          LTY PTR
027E 20C002        JSR ADD          ;ADD TO ACCOUNT
0281 F0D1          BFQ BAL          ;SHOW BALANCE
0283 C90D          DBT CMP #S0D          ;DEBIT ACCOUNT?
0285 D0A0          BNE DSPLY      ;NO, DISCARD KEY
0287 20C902        JSR SUB          ;YES, SUB FROM ACCOUNT #0
028A B00B          BCS DBCK      ;NEGATIVE RESULT?
028C A000          DBERR LTY #S00      ;YES, ADD # BACK
028E 20C002        JSR ADP
0291 A9EE          LFA #SEF          ;PUT F'S IN DISPLAY
0293 85F9          STA INH
0295 D08E          BNF CHRS          ;SHOW ERROR (EEEE EE)
0297 A400          DBCK LTY PTR
0299 20C902        JSR SUB          ;SUB FROM ACCOUNT
029C B0B6          BCS BAL          ;SHOW BALANCE IF O.K.
029E A400          LDY PTR          ;ADD # BACK IF ERROR
02A0 20C002        JSR ADD
02A3 F0E7          BEQ DBERR      ;ADD BACK TO ACCT 0 & SHOW ERROR
02A5 C90A          GETA CMP #S0A          ;DISPLAY ACCOUNT #?
02A7 D005          BNE NEWA
02A9 A501          LDA ACCT          ;YES, GET #?
02AB 4C0202        SHOWA JMP SETA          ;SHOW IT
02AE C912          NEWA CMP #S12          ;INC ACCOUNT #?
02B0 D0AF          BNE LINK      ;NO, CLEAR INPUT FLAG & WAIT
02B2 A501          LDA ACCT          ;YES, GET ACCOUNT #
02B4 F8           SED          ;GC TO DECIMAL MODE
02B5 18           CLC          ;CLEAR CARRY
02B6 6901          ADC #S01          ;INC #
02B8 D8           CLD          ;BACK TO BINARY MODE
02B9 C9           CHKA BYT $C9      ;'CMP'--CHECK ACCOUNT #
02BA 00           BYT $00          ;LAST ACCOUNT #1 + 1 GOES HERE
02BB              ;

```


Dear Editor:

I'm cheating, really... with regard to your "Too Many Apples!" editorial.... You asked for feedback from readers, and I'm not a regular MICRO reader.

I bought your February issue just for the "In the Heart of Applesoft" article. MICRO is a quality publication, but there just isn't enough Apple-related material to justify my subscribing.

I think I might be fairly typical of the sort of reader who would buy MICRO (or its Apple edition) on a regular basis if there were enough Apple coverage.

Good grief—thousands of us have kilobucks invested in Apple (it's the kind of computer that attracts intelligent laymen—no wonder so many articles of high quality are submitted!). With Pascal, I can tackle projects that are light years beyond the scope of KIM, SYM, et. al. and I'm hungry for reading material!

As far as I'm concerned, you can publish just one more article for those bare-board, skinflint, kitchen table time-wasters—"How to Convert Your KIM-1 Into a Dedicated Coffee Percolator." That's it—the final article!

Given the choice of publishing a magazine for the relatively well-heeled and serious users of what may well be the "Model A" of computing versus publishing one for a minority of impoverished assembly language hobbyists, it surprises me that you resist the opportunity to publish more Apple-based articles.

Where will KIM, SYM, PET and AIM be in ten years? Apple, and its progeny, might well dominate the world of microcomputing by that time. Will MICRO ride the bandwagon or drag its heels? The choice is yours!

Please don't publish my name or address. I don't want my Apple stolen!

Editor's Note: Beginning in June, MICRO will be expanding to include more Apple articles each month. We thank all who responded to the "Too Many Apples!" editorial.

```

0800 SYMKEY EQU $8923 ;CHECK FOR ANY KEY DOWN
0800 SYMLRN EQU $892C ;DETERMINE KEY PRESSED
0800 SYMSEG EQU $8C29 ;LED SEGMENT CODES
;
; ORG $100 ;OUT OF THE WAY ON STACK PAGE
; OBJ $800
;
;SYM-1 VERSION OF KIM SCAND & SCANDS ROUTINES
;
SCAND LDY #$00 ;ENTER HERE TO GET BYTE
LDA (POINTL),Y ;ADDRESSED BY POINTL
STA INH ;AND MOVE IT TO INH AREA
;
SCANDS LDY #$00 ;ENTER HERE IF INH ALREADY STORED
LDA POINTH ;POINTH FIRST TO DISPLAY BUFFER
JSR SPLITP ;THEN DO POINTL
LDA POINTL
JSR SPLITP
LDA INH ;LAST, BUT NOT LEAST, DO INH
JSR SPLITP
JMP SYMSCA ;SET SYM MONITOR LIGHT & RETURN
;
SPLITP PHA ;SAVE ORIGINAL
LSR ; ON STACK FOR LATER
LSR ;SHIFT HI HALF TO LO HALF
LSR ;WHICH IS 4 BITS DOWN
TAX ;PUT INTO X AS AN INDEX
LDA SYMSEG,X ;GET APPROPRIATE SEGMENT CODE
STA SYMDIS,Y ; AND PUT INTO DISPLAY BUFFER
INY ;BUMP Y FOR NEXT BYTE
PLA ;NOW GET ORIGINAL VALUE BACK
AND #$0F ;KEEP ONLY LOW ORDER 4 BITS
TAX ;AND REPEAT SEGMENT PROCESS
LDA SYMSEG,X
STA SYMDIS,Y
INY ;INCLUDING BUMP FOR NEXT BYTE
RTS ; AND RETURN
;
;SYM-1 VERSION OF GETKEY SUBROUTINE
;
GETKEY JSR SYMLRN ;GET SYM VERSION OF THE KEY
BNE KEYDWN ;BRANCH IF ANY KEY IS DOWN
GKNONE LDA #$15 ;ELSE SET TO KIM NO KEY DOWN
RTS ;AND RETURN
KEYDWN TXA ;X HOLDS INDEX INTO ASCII TABLE
CMP #$11 ;NEED TO FUDGE KEY VALUE?
BCC GKRTS ;00-0F IS OK 10=AD(KIM)=CR(SYM)
CMP #$16 ;CHECK FOR OUT OF KIM RANGE
BCS GKNONE ;AND TREAT AS A 'NO KEY'
LDA TRANSO ;ELSE TRANSLATE THROUGH TABLE
GKRTS RTS ; AND RETURN
;
TRANST BYT $12 ; '+' (KIM) = '- / +' (SYM)
BYT $11 ; 'DA' (KIM) = NO KEY (KIM)
BYT $15 ; SHIFT (SYM) = NO KEY (KIM)
BYT $13 ; 'G' (KIM) = 'GO/LP' (SYM)
BYT $14 ; 'PC' (KIM) = 'REG/SP' (SYM)
;
;SYM-1 VERSION OF KIM KEYIN SUBROUTINE
;
KEYIN JSR SYMKEY ;GET KEYBOARD STATUS
BNE KEYIN2 ;REVERSE ZERO FLAG
LDX $FF ;KIM NOT ZERO--NO KEY--FF FOR LRNKEY
RTS
KEYIN2 LDX #$00 ;AND IS ZERO IF KEY IS DOWN
RTS
;
;SYM-1 VERSION OF KIM CONVD ROUTINES $1F48 & $1F4E
;
CONVD STY PZSCR ;SAVE Y IN SCRATCH AREA
TAY ;MOVE NIBBLE OF A TO INDEX REGISTER
LDA SYMSEG,Y ;GET HEX SEGMENT CODES FROM TABLE
DISPCH STX SYMPBD ;SELECT THE DIGIT
STA SYMPAD ;OUTPUT THE SEGMENT CODES
LDY #$10 ;KEEP IT LIT FOR A WHILE
LIGHT DEY
BNE LIGHT ;TURN ALL SEGMENTS OFF FOR NEXT ONE
STY SYMPAD ;BUMP X TO NEXT DIGIT
INX ;RESTORE THE Y REGISTER
LDY PZSCR ;AND RETURN
RTS

```



MICRO

Challenges

By Paul Geffen

Last month's column may have left the impression that Ohio Scientific provides very little documentation for its products. This has been the case until recently. The new management at OSI is making an effort to improve the quantity and quality of its documentation. This effort, started last year, is still underway. Here is a report on the results so far.

Last year OSI published a set of revised *User's Manuals* for the C1P, C4P, and C8P personal computers. These were a big improvement over the previous versions. The new *User's Manuals* look better, contain more information, and are much more reliable than the old ones. They also include illustrations and photographs which are valuable to the beginner. These manuals cover the middle ground because they assume a certain amount of knowledge about computers, but contain limited detail about the inner workings of the machines.

More recently, OSI has published manuals for the novice as well as for the more advanced user. For the novice, there is now a series of five *Introductory Manuals*, for the C1P, C4P, C4PMF, C4PDF and C8PDF. I have seen only the first of these; the rest should be available by the time you read this. These manuals will be included with the computers, along with the *User's Manual*. These *Introductory Manuals* assume very little and are designed for the beginner. They include many photographs and provide detailed instructions on how to set up the machine and save programs.

The manuals also provide very little general information about BASIC. They serve to de-mystify the machine and make it accessible to someone who knows next to nothing about computers. This approach is designed to make OSI personal computers appeal to the mass market, those people who now form the fastest growing part of the computer market.

Also for the beginner, OSI publishes two introductory BASIC texts. The first is *Understanding Your Ohio Scientific C1P and C4P, A Workbook of Programming Exercises in BASIC* by Keith

Russell and David Schultz. This book covers all the capabilities commands and keywords of OSI BASIC (with the exception of the `USR(X)` function). The book is limited to the BASIC language and avoids machine level information so as not to confuse the reader. It is also written specifically for OSI machines and contains information peculiar to these machines, like how to get started and what POKEs to use to change the screen format.

The second BASIC text is *BASIC and the Personal Computer* by T.A. Dwyer and M. Critchfield. This book is four times as long as the one by Russell and Schultz, and is much more detailed as well as broader in scope. While the former covers only the basics, the latter includes chapters on applications like word processing, games, art, simulation, data structures, sorting and files. In addition to providing an introduction to computers and the BASIC language, Dwyer and Critchfield cover many of the possible applications of personal computers. This book was not written for OSI computers. It is a general BASIC text for college courses published by Addison-Wesley with a special cover for OSI. Some of the material here applies to other versions of BASIC, but for the most part the book is written for users of any machine. Both texts assume very little initially, but the one by Dwyer and Critchfield goes further and faster.

For BASIC programmers who want to learn about machine language, OSI publishes the *65V Primer*, an introduction to machine code on the OSI personal computers. OS 65V is the machine level monitor program in ROM which provides the most fundamental support for other programs like BASIC. This book is also a good introduction to 6502 assembly or machine language programming. It describes all of the machine instructions and contains many examples and exercises. It does not assume any knowledge of computers, but it helps to be able to program in BASIC before reading this book.

OSI has completed two new reference manuals which I have not seen but which should be available shortly. One is a new and improved *BASIC Reference Manual* and the other is an *Assembler/Editor/Extended Monitor Reference Manual*. I plan to review these in a future column.

For the hardware expert or repairman, Ohio Scientific and Howard Sams publish three detailed *Servicing Manuals* for OSI personal (C1P and C4P) and business computers (CII and CIII). These contain block diagrams,

parts lists, schematics, photos of the boards, and very little text. They are essential for repairing the computers and helpful when modifying the circuitry. These manuals assume the reader has a good electronics background, the ability to read schematics, and a working knowledge of digital electronics. The only item missing from these manuals is a schematic of the power supply, which is represented as a "black box."

All of the above documentation is available from OSI dealers, separate from the computers. In addition to these publications, OSI has expanded its customer service department and its programming staff. If you own an OSI machine and have questions about hardware or software, write to the Customer Service department, 1333 South Chillicothe Road, Aurora, Ohio 44202, or call (216) 831-5600.

For the experienced assembly language programmer who wants to know everything about the internal operation of the OS65D V3.2 disk operating system, a complete commented disassembly of OS65D V3.2 is available from Software Consultants, 7053 Rose Trail, Memphis, Tennessee 38134. This book has been praised in all the newsletters. I have just received a copy and plan a full report in my next column.

The C2-4P

This is a relatively old OSI personal computer. It is no longer in production but since there are quite a few of them around it deserves mention. This model has since been upgraded to the C4P. The only differences between the two are that the C2-4P has an older (rev A) video display board without color, and an older version of the CPU board with fewer I/O lines. Other than that, the only difference is the enclosure. C2-4P software will run on the C4P and most C4P software will run on the C2-4P unless it requires the I/O ports on the new CPU board.

A used C2-4P can be a very inexpensive personal computer but it helps to know how to maintain it. Schematics for the older boards are not in the C4P servicing manual but can be obtained from OSI directly. (Write to Bill Conrad at Customer Service.) Despite its age, this model is not obsolete. Much software continues to be written on and for this computer. And it can be converted into a C4P by replacing the two boards mentioned above.

"NIBBLE® IS TERRIFIC" (For Your Apple)



NIBBLE IS: *The Reference for Apple computing!*

NIBBLE IS: One of the Fastest Growing new Magazines in the Personal Computing Field.

NIBBLE IS: Providing Comprehensive, Useful and Instructive Programs for the Home, Small Business, and Entertainment.

NIBBLE IS: A Reference to Graphics, Games, Systems Programming Tips, Product News and Reviews, Hardware Construction Projects, and a host of other features.

NIBBLE IS: A magazine suitable for both the Beginner and the Advanced Programmer.

Each issue of NIBBLE features significant new Programs of Commercial Quality. Here's what some of our Readers say:

- "Certainly the best magazine on the Apple II"
- "Programs remarkably easy to enter"
- "Stimulating and Informative; So much so that this is the first computer magazine I've subscribed to!"
- "Impressed with the quality and content."
- "NIBBLE IS TERRIFIC!"

In coming issues, look for:

- Stocks and Commodities Charting
- Assembly Language Programming Column
- Pascal Programming Column
- Data Base Programs for Home and Business
- Personal Investment Analysis
- Electronic Secretary for Time Management
- The GIZMO Business Simulation Game

And many many more!

NIBBLE is focused completely on the Apple Computer systems.

Buy NIBBLE through your local Apple Dealer or subscribe now with the coupon below.

Try a NIBBLE!

nibble

Box 325, Lincoln, MA. 01773 (617) 259-9710

I'll try nibble!

**Enclosed is my \$17.50 (for one year).
(Outside U.S., see special rates on this page.)**

check **money order**

Your subscription will begin with the next issue published after receipt of your check/money order.

Name _____

Address _____

City _____

State _____ Zip _____

NOTE:

First Class or Air Mail is required for all APO, FPO and all foreign addresses with the following additional amounts.

- | | |
|--------------------------------------|-----------------------------|
| - Europe \$32.00 | Africa: North \$32.00 |
| - Mexico and Central America \$21.00 | Central \$43.00 |
| - South America \$32.00 | South \$43.00 |
| - Middle East \$35.00 | Far East, Australia \$43.00 |
| | Canada \$18.00 |

All payments must be in U.S. funds drawn on a U.S. bank.

© 1980 by MICRO-SPARC, INC., Lincoln, Mass. 01773. All rights reserved.
*Apple II is a registered trademark of Apple Computer Company

More Output from your Micro

Here is a simple way to add extra output bits to your single board microcomputer. This technique will work on the AIM, SYM, KIM and OSI Superboard or C1P. The method is similar to that used on the Apple II for generating sound, and a "random beeper" program concludes this article.

H.H. Aumann
1262 Rubio Vista
Altadena, California 91001

The circuit in figure 1 provides an independent output bit which can be turned on and off under program

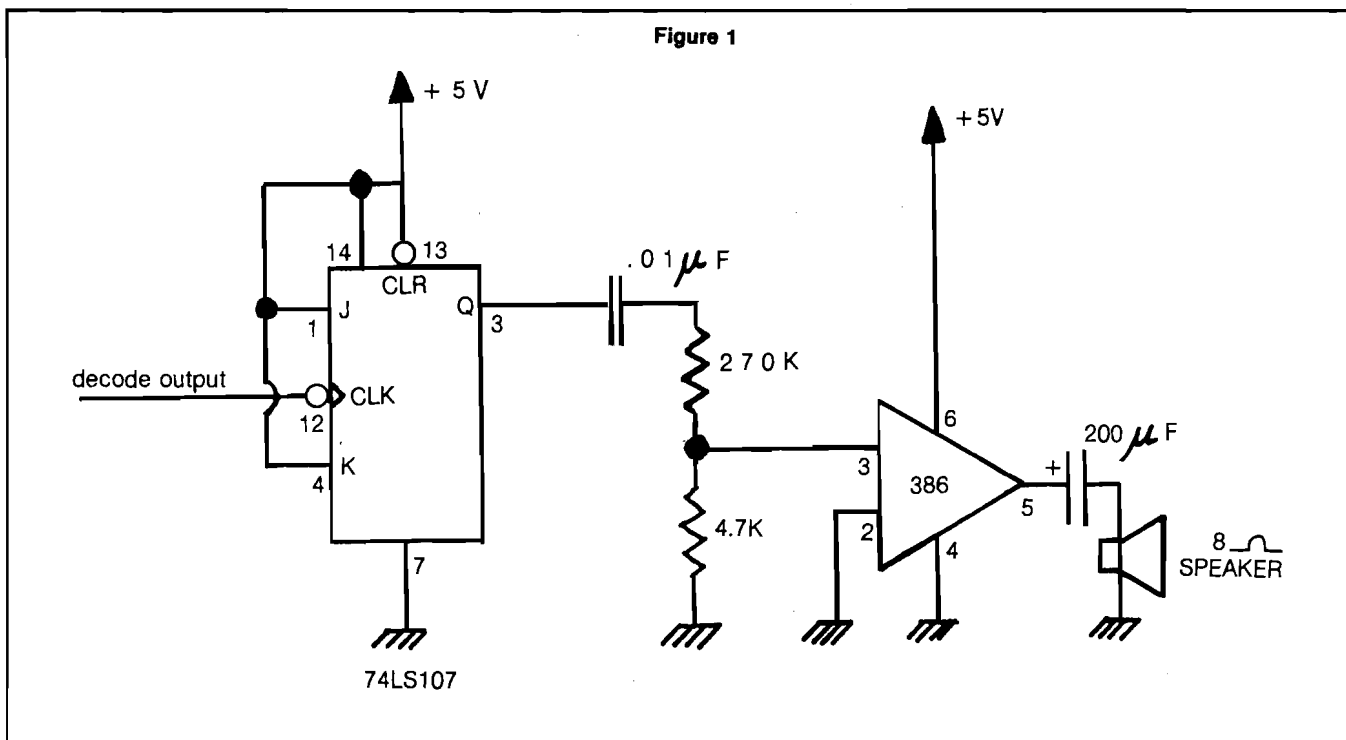
control with minimal effort and without tying up your VIA. It consists of one half of a 74LS107 dual JK flip-flop connected to unused address decode outputs from the computer. In this case, the flip-flop is used to drive an audio amplifier and a speaker to provide sound output, but many other applications are possible. For example, the output bit could be used to control a printer or other device.

This is how it works. All microcomputers use decoders to divide the 64K range of possible addresses into more manageable units. Some of the outputs from these decoders are not used on the board and are available for

other purposes. Table 1 shows what addresses are unused on each micro and where the corresponding signals may be found.

Two of these decoder outputs are used to set and reset the flip-flop in figure 2. Almost any flip-flop may be substituted, as shown. The output of the flip-flop is the new output bit. If all you want to do is toggle the output then the decode line may be connected to the clock input of the flip-flop. In this case only one address is needed for access but you may not be able to tell if the output bit is on or off. In the case of sound output this is not important.

Figure 1



If you use the circuit in figure 2, then a read from an address corresponding to decode output 1 will turn on the output bit, and a read from an address corresponding to decode output 2 will turn off the output bit.

The following BASIC program generates random sound output on the OSI Superboard. The machine language program is relocatable and will run on any machine by changing the byte at \$0226. The BASIC program must be changed to load the machine code in a convenient location. This is left to the reader as an exercise.

Table 1

AIM	SYM	KIM	OSI
\$8000 A-18	\$1000 A-F	\$0400 A-C	\$D400 write U20 - 9
\$9000 A-19	\$1400 A-H	\$0800 A-D	\$D800 write U20 - 10
\$A000 A-20	\$1800 E-16	\$0C00 A-E	\$D400 read U20 - 13
	\$1C00 A-J	\$1000 A-F	\$D800 read U20 - 14

A - Application Connector
E - Expansion Connector

Note: U20 is a 74138 decoder chip on the OSI model 600 CPU board.

```

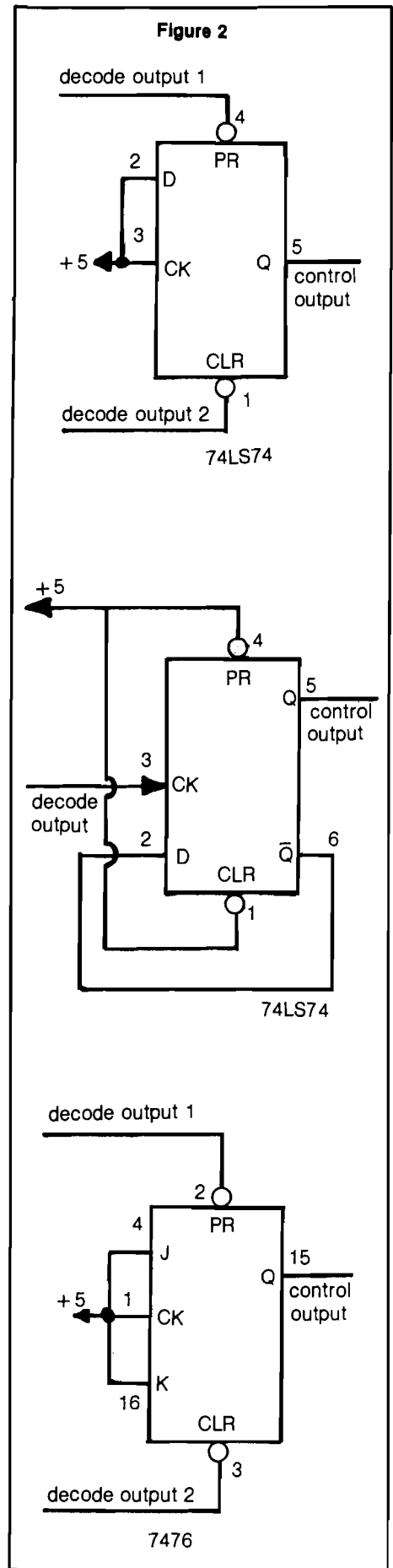
10 REM SIMPLE TONE GENERATOR DEMO
12 REM FOR OSI SUPERBOARD / C1P
20 FOR I = 548 TO 567: READ BI
25 POKE I,BI: NEXT : REM LOAD $224-237
30 POKE 11,36: POKE 12,2
35 REM SET ENTRY POINT FOR USR(1)
40 D = 256 * RND (1):P = 256 * RND (1)
50 POKE 546,P: POKE 547,D
55 REM SET PERIOD AND DURATION
60 X = USR (1): GOTO 40
70 DATA 173,0,216,136,208,5,206,35,2,240
80 DATA 8,202,208,245,174,34,2,208,237,96

```

```

                                ORG $0222
                                PERIOD DFS 1
                                DURATN DFS 1
AD00D8                          SOUND LDA $D800
D003                             LOOP BNE SKIP
CE2302                          DEC DURATN
F008                             SKIP BEQ DONE
CA                               DEX
D0F6                             BNE LOOP
AE2202                          LDX PERIOD
D0EE                             BNE SOUND
60                               DONE RTS

```

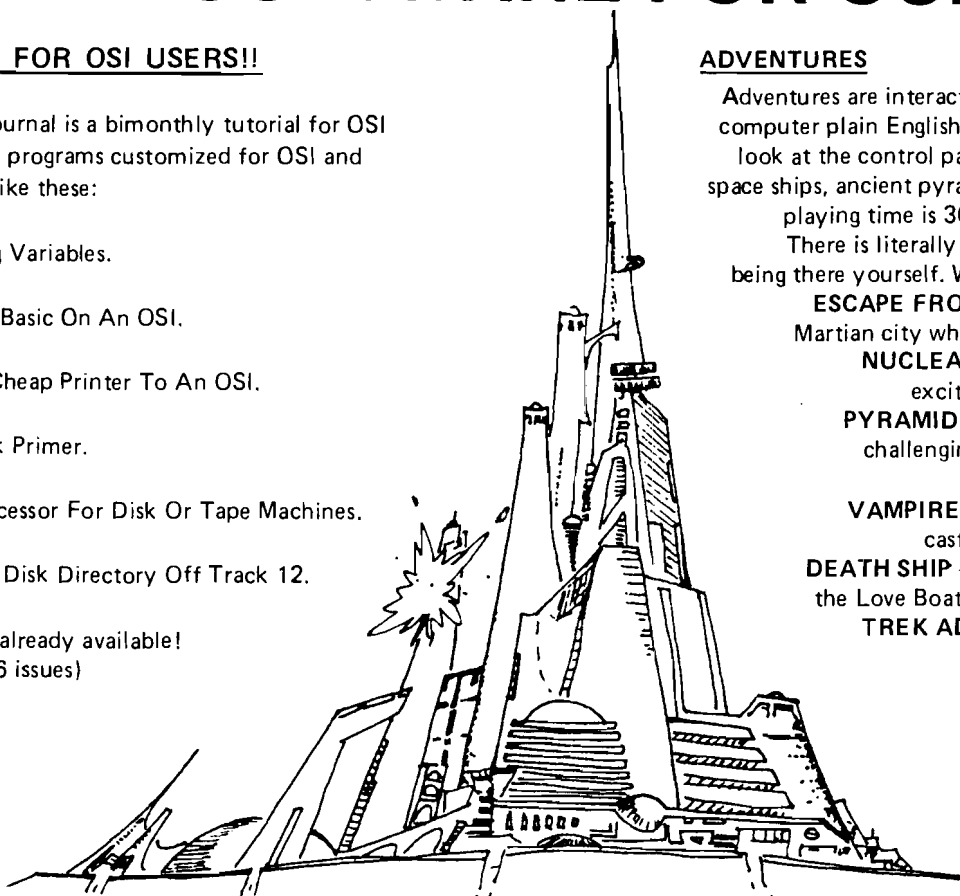


A JOURNAL FOR OSI USERS!!

The Aardvark Journal is a bimonthly tutorial for OSI users. It features programs customized for OSI and has run articles like these:

- 1) Using String Variables.
- 2) High Speed Basic On An OSI.
- 3) Hooking a Cheap Printer To An OSI.
- 4) An OSI Disk Primer.
- 5) A Word Processor For Disk Or Tape Machines.
- 6) Moving The Disk Directory Off Track 12.

First year issues already available!
\$9.00 per year (6 issues)

NEW SUPPORT ROMS FOR BASIC IN ROM MACHINES

C1S — for the C1P only, this ROM adds full screen edit functions (insert, delete, change characters in a basic line). Software selectable scroll windows, two instant screen clears (scroll window only and full screen), software chose of OSI or standard keyboard format, Bell support, 600 Baud cassette support, and a few other features. It plugs in in place of the OSI ROM. NOTE: this ROM also supports video conversions for 24, 32, 48 or 64 characters per line. Replaces video swap tape on C1P model 2. All that and it sells for a measly \$39.95.

C1E/C2E for C1/C2/C4/C8 Basic in ROM machines. This ROM adds full screen editing, software selectable scroll windows, keyboard correction (software selectable), and contains an extended machine code monitor. It has breakpoint utilities, machine code load and save, block memory move and hex dump utilities. A must for the machine code programmer replaces OSI support ROM. Requires installation of additional chip when installed in a C2 or C4. C1 installation requires only a jumper move. Specify system \$59.95.

DISK UTILITIES**SUPER COPY** — Single Disk Copier

This copy program makes multiple copies, copies track zero, and copies all the tracks that your memory can hold at one time — up to 12 tracks at a pass. It's almost as fast as dual disk copying. — \$15.95

MAXIPROSS (WORD PROCESSOR) — 65D polled keyboard only - has global and line edit, right and left margin justification, imbedded margin commands, choice of single, double or triple spacing, file access capabilities and all the features of a major word processor — and it's only \$39.95.

P.C. BOARDS

MEMORY BOARDS!! — for the C1P. — and they contain parallel ports!

Aardvark's new memory board supports 8K of 2114's and has provision for a PIA to give a parallel ports! It sells as a bare board for \$29.95. When assembled, the board plugs into the expansion connector on the 600 board. Available now!

PROM BURNER FOR THE C1P — Burns single supply 2716's. Bare board — \$24.95.

MOTHER BOARD — Expand your expansion connector from one to five connectors or use it to adapt our C1P boards to your C4/8P. — \$14.95.

ARCADE AND VIDEO GAMES

GALAXIA one of the fastest and finest arcade games ever written for the OSI, this one features rows of evasive, hardhitting, dogfighting aliens thirsty for your blood. For those who loved (and tired of) Alien Invaders. — P.S. The price is a giveaway. SPECIFY SYSTEM!
Cassette \$9.95 — Disk \$12.95

TIME TREK (8K) — real time Startrek action. See your torpedoes move across the screen! Real graphics — no more scrolling displays. \$9.95

INTERCEPTOR C1P ONLY! An all machine code program as fast and smooth as the arcades. You use your interceptor to protect your cities from hordes of enemy invaders. A pair of automatic cannons help out, but the action speeds up with each wave of incoming ships. The fastest and most exciting C1P game yet.
C1P Cassette \$19.95

MINOS — A game with amazing 3D graphics. You see a maze from the top, the screen blanks, and then you are in the maze at ground level, finding your way through on foot. Realistic enough to cause claustrophobia. — \$12.95

ADVENTURES

Adventures are interactive fantasies where you give the computer plain English commands (i.e. take the sword, look at the control panel.) as you explore alien cities, space ships, ancient pyramids and sunken subs. Average playing time is 30 to 40 hours in several sessions.

There is literally nothing else like them — except being there yourself. We have six adventures available.

ESCAPE FROM MARS — Explore an ancient Martian city while you prepare for your escape.

NUCLEAR SUBMARINE — Fast moving excitement at the bottom of the sea.

PYRAMID — Our most advanced and most challenging adventure. Takes place in our own special ancient pyramid.

VAMPIRE CASTLE — A day in old Drac's castle. But it's getting dark outside.

DEATH SHIP — It's a cruise ship — but it ain't the Love Boat and survival is far from certain.

TREK ADVENTURE — Takes place on a familiar starship. Almost as good as being there.

SINGLE STEPPER / MONITOR

This is probably the finest debugging tool for machine code ever offered for OSI systems. Its' trace function allows you to single step through a machine code program while it continuously displays the A, X, Y and status registers and the program and stack pointers. You can change any of the registers or pointers or any memory location at any time under program control. It takes well under 1k and can be relocated anywhere in free memory. It is a fine tool for all systems — and the best news of all is the extremely low price we put on it. — Tape \$19.95 — Disk \$24.95

FOR DISK SYSTEMS — (65D, polled keyboard and standard video only.)

SUPERDISK. Contains a basic text editor with functions similar to the above programs and also contains a renumberer, variable table maker, search and new BEXEC* programs. The BEXEC* provides a directory, create, delete, and change utilities on one track and is worth having by itself. — \$24.95 on 5" disk - \$26.95 on 8".

AARDVARK IS NOW AN OSI DEALER!

Now you can buy from people who can support your machine.

— THIS MONTH'S SPECIALS —

Superboard II	\$279
C1P Model II	429
C4P	749
8K 610 board for C1P	269
Epson MX-80 printer with RS232 installed	595

... and we'll include a free Text Editor Tape with each machine!

True 32X32 Video Mod Plans for C1P
(4 Chips \$3.00 Crystal Required)
\$7.95

This is only a partial listing of what we have to offer. We now offer over 100 programs, data sheets, ROMS, and boards for OSI systems. Our \$1.00 catalog lists it all and contains free program listings and programming hints to boot.



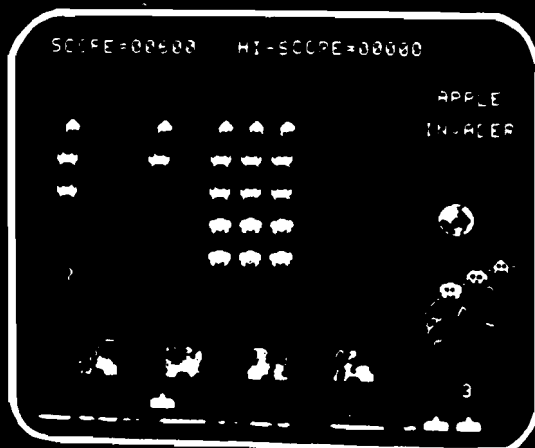
SPACE WAR

You're in command in **SPACE WAR!** Destroy your opponent's ship by forcing him to collide with the sun or to explode upon re-entry from hyperspace... or challenge him face to face with missile fire. You're in command of the speed and direction of your ship. You control the timing of your missiles. You select the game mode from five options, including Reverse Gravity, and the battle begins. Accelerate to place your shots--and escape into hyperspace before your opponent comes within range. But be wary...he (or she!) may circle out of sight and reappear on the opposite side of the galaxy! (This is the classic MIT game redesigned especially for the Apple.)



and SUPER INVASION

- **Super Invasion** is the original invasion game, with the original moon creatures and faster action than any other invasion game.
- Features superb high resolution graphics, nail-biting tension and hilarious antics by the moon creatures!
- Self-running "attract mode" of operation for easy learning and demonstrating of the game.
- As good in every way as the famous Invaders arcade game.
- High speed action! • Sound effects!
- Runs on the Apple II and the Apple II Plus



Fifty-five aliens advance and shower you with lethal writhing electric worms. As you pick off the aliens, one-by-one, they quicken their descent. They whiz across the screen wearing away your parapets, your only defense, coming closer and closer to your level. **Super Invasion** is the **original** invasion game with the original moon creatures and faster action than any other invasion game on the market.

Super Invasion is available for only \$19.95 on cassette (CS-4006) for a 32K Apple II. **Space War** is \$14.95 on cassette (CS-4009) for a 16K Apple II. **Space War** and **Super Invasion** are on one disk (CS-4508) for a 48K Apple II for only \$29.95.

Send payment plus \$1.00 shipping and handling to Creative Computing Software, P.O. Box 789-M, Morristown, NJ 07960. NJ residents add \$1.00 sales tax. Bankcard orders may be called in toll free to 800/631-8112. In NJ call 201/540-0445.

**sensational
software**

**creative
computing
software**

Note: This program is written to work with ROM Applesoft. To convert to disk or cassette would require considerable effort. If you want to convert it you will have to figure out the equivalent addresses in RAM Applesoft and change the equates in the listing.

I welcome any comments. Please send them to the address at the beginning of the article and include a S.A.S.E if you desire a reply.

LIST

```
10 AA = 98E + 05
20 R$ = "THIS IS A
    TEST"
30 N% = 2341
40 RR = 12345
50 FOR I = 0 TO 10:
    NEXT I
60 END
```

]RUN

```
]BRUN VARIABLE DUMP
AA 9800000
R $ THIS IS A TEST
N % 2341
RR 12345
I 11
```

]A=34567.98

]BB%=-32767

]CC\$="MOOSE"

]DD=12324+98

]N%=1232

```
]BRUN VARIABLE DUMP
A 34567.98
BB% -32767
CC$ MOOSE
DD 12422
N % 1232
```

```
4032 205CDB      JSR OUTDO
4035 2057DB      JSR OUTSPC
4038 A002        LDY #302
403A B106        LDA (POINTL),Y      ; GET THE INTEGER
403C AA          TAX
403D C8          INY
403E B106        LDA (POINTL),Y
4040 A8          TAY
4041 8A          TXA
4042 20F2E2      JSR GIVAYF          ; CONVERT TO FLOATING POINT
4045 202EED      JSR PRTFAC          ; PRINT IT.
4048 4C8540      JMP NXTSIM
404B             REAL:
404B 209940      JSR PRINTN          ; REAL HANDLING STARTS HERE.
404E 2057DB      JSR OUTSPC
4051 2057DB      JSR OUTSPC
4054 A407        LDY POINTH
4056 A506        LDA POINTL
4058 18          CLC
4059 6902        ADC #302
405B 9001        BCC CONT
405D C8          INY
405E 20F9EA      CONT JSR MOVEFM          ; USE APSOFT INTERNALS TO
4061 202EED      JSR PRTFAC          ; DO THE DIRTY WORK.
4064 4C8540      JMP NXTSIM
4067             STRING:
4067 209940      JSR PRINTN
406A A924        LDA #324
406C 205CDB      JSR OUTDO
406F 2057DB      JSR OUTSPC
4072 A002        LDY #302
4074 B106        LDA (POINTL),Y      ; SET UP THE POINTERS FOR
4076 85A0        STA LEN              ; STROUT SUBROUTINE.
4078 C8          INY
4079 B106        LDA (POINTL),Y
407B 859E        STA SPL
407D C8          INY
407E B106        LDA (POINTL),Y
4080 859F        STA SPH
4082 20AD40      ; JSR STROUT          ; PRINT THE STRING.
4085             ;
4085             ; NXTSIM SETS THE VARIABLE POINTER TO THE NEXT VARIABLE.
4085             ;
4085             NXTSIM:
4085 20FBDA      JSR CRDO
4088 18          CLC
4089 A907        LDA #307
408B 6506        ADC POINTL
408D 8506        STA POINTL
408F 9002        BCC CONT2
4091 E607        INC POINTH
4093 20BD40      CONT2 JSR WAIT
4096 4C0B40      JMP LOOP
4099             ;
4099             ; PRINTN PRINTS THE NAME OF THE CURRENT VARIABLE.
4099             ;
4099             PRINTN:
4099 A000        LDY #300          ; INDEX=0
409B B106        LDA (POINTL),Y      ; GET FIRST LETTER
409D 205CDB      JSR OUTDO
40A0 C8          INY          ; MOVE UP
40A1 B106        LDA (POINTL),Y      ; GET NEXT CHAR IN NAME
40A3 297F        AND #37F
40A5 D002        BNE CONT3          ; IF THIS IS A SINGLE CHARACTER
40A7 A9A0        LDA #3A0          ; NAME THEN PRINT A SPACE.
40A9 205CDB      CONT3 JSR OUTDO
40AC 60          RTS
40AD             ;
40AD             ;
40AD             ; STROUT PRINTS A STRING POINTED
40AD             ; TO BY SPL,SPH OF LENGTH LEN
40AD             ;
40AD             ;
40AD             ; STROUT:
40AD A000        LDY #300
40AF C4A0      LOOP1 CPY LEN
40B1 F009        BEQ RTS1
40B3 B19E        LDA (SPL),Y
40B5 205CDB      JSR OUTDO
40B8 C8          INY
40B9 4CAF40      JMP LOOP1
40BC 60          RTS1 RTS
40BD             ;
40BD             ;
40BD             ; WAIT LOOKS AT THE KEYBOARD
40BD             ; TO SEE IF A KEY WAS PRESSED.
40BD             ; IF SO, IT WAITS FOR A SECOND
40BD             ; KEY TO BE PRESSED BEFORE IT
40BD             ; RETURNS.
40BD             ;
40BD             ;
40BD             ;
40BD AD00C0      WAIT LDA KBOARD
40C0 10FA        BPL RTS1
40C2 AD10C0      LDA STROBE
40C5 AD00C0      WAIT1 LDA KBOARD
40C8 10FB        BPL WAIT1
40CA AD10C0      LDA STROBE
40CD 60          RTS
```


MICRO

Microprocessors in Medicine: The 6502

Jerry W. Froelich, M.D.
9 Brown Place
Woburn, Massachusetts 01801

Previously in this column we have not discussed applications where the computer helps in the "direct" management of patient therapy. In this issue we will describe a "PET" computer which keeps track of patients' data who are being treated with a blood anticoagulant called warfarin [coumadin].

Background

There are several medical conditions where retarding of the rate at which blood clots is necessary to prevent further complications of the disease. The drug warfarin is administered to "slow down" the blood clotting mechanism.

An example where anticoagulation is necessary is in a disease called "Deep Venous Thrombophlebitis" (DVT) where blood clots form in the deep veins of the legs. This condition is potentially life threatening in that a large blood clot may dislodge from the leg, travel through the heart, and lodge in the lung. If the clot is large enough it may totally block the flow of blood to the lungs resulting in shock or death, due to asphyxiation.

Warfarin is used for the long-term treatment of diseases, but warfarin itself may be life threatening if administered in incorrect dosages. If too much drug is administered, blood vessels may bleed spontaneously. If too little, the clotting process won't be "slowed down." Therefore, drug dosage and drug breakdown (in the patient) are critical to the use of warfarin. The predictable body metabolism of the drug makes it safe to use as long as the patient is monitored closely.

Model of Anticoagulation Therapy

Dr. William F. Powers at the University of Michigan has approached anticoagulation therapy from the systems point of view. Although he was not the first to apply computer

techniques to this problem, he was the first to use a microprocessor at the "bedside" to guide the physician and monitor anticoagulation therapy. The next several paragraphs (which refer to figure 1) discuss the algorithm used to model anticoagulation therapy, talk about the computerization of the algorithm, and show the equations for the dynamic model of anticoagulation therapy.

$$G = -G K_g + D(t) \quad (1)$$

$$Q = G K_g - K_e Q \quad (2)$$

$$P = S_m [1 - F / F + K_m] - K_p P \quad (3)$$

Where F approximately equal to 0.003 Q / Warfarin distribution volume which is approximately equal to aQ. Then,

$$P = S_m [1 - Q / Q + K] - K_p P \quad (4)$$

Where $K = K_m / a$

Figure 1: The basic equations for the dynamic model of anticoagulation therapy.

Equation (1) assumes that the rate of absorption of warfarin from the intestinal tract (the drug is taken by mouth) is proportional to the amount in the intestine, G.

The warfarin dosage schedule is described by the function D(+). Equation (2) states that the instantaneous rate of change of the amount of warfarin in the body, Q, is given by the difference between the rate of absorption from the intestinal tract and its rate of metabolism, the metabolism being a first order process.

Equation (3) states the rate of the prothrombin complex activity, (p), a blood clotting parameter that monitors the extrinsic pathway of the coagulation cascade and is equal to the difference between the rate of synthesis of the complex and its rate of degradation. The reduction in the rate of prothrombin complex synthesis below the normal value (S_m) due to inhibition by warfarin is given by a Michaelis-Menter formulation, in which [F] is the concentration of free warfarin and (K_m) the Michaelis constant.

The parameters K_g, K_e, K_p, and K are subject-dependent parameters. One of the goals of the program is to rapidly reach estimates of the subject-

dependent variables from the direct measurements of patient clotting time (prothrombin time tests). Once these parameters have been determined, the initial dosage may be optimized for a given patient. Subsequently, the program will be able to monitor changes of the patient's clotting time during the maintenance phase of drug administration (the daily dosage required to keep the clotting time in the therapeutic range).

Dr. Powers has analyzed three methods (1) to calculate the patient-dependent parameters from K_g, K_e, etc. He also settled on a "tuned" extended Kalman filter method because it was accurate and it could compute the results within a few minutes, whereas other techniques require between 30 and 60 minutes of computer time per case.

Overview of the Programs

Dr. Powers has written a series of programs. The early programs were mainly concerned about the simulation of the anticoagulation kinetics and estimation of the patient-dependent parameters. An example of the graph generated by the Kalman filter program is shown in figure 2. This graph displays both the measured and the computed prothrombin complex activities as a function of time. The patient-dependent parameters are also displayed at the top of the figure.

The most recently written program is appropriately titled MAINTENANCE. This program is menu-driven and interacts with the user promoting a better man-machine interaction. Figure 3 is an example of the menu. When the program is initially run the program requests various information about the patient being monitored: the patient's full name, phone number, parameters, and general comments. The program then requests the most recent laboratory values of the prothrombin complex activities. After entering the measurements the program can recommend a dosage and follow-up period (figure 4) from as few as four measurements.

After each patient's data has been entered it can be saved on a floppy disk so that when the patient returns for another visit, the data file can be recalled. The program also flags such conditions as an abnormal response to therapy. The data files can be backed up by copying the disk.

The program was not designed to replace the physician, but rather to assist the physician in the management of patients. All recommendations regarding therapy must be checked against the physician's understanding of the therapy. By managing the patients in this manner the computer and physician check each other and therefore yield higher quality medicine.

Summary

The previous sections have described how an inexpensive microcomputer such as the Commodore PET 2001 can be utilized with modern techniques to rapidly assess therapy and recommend dosages for patients receiving anticoagulation therapy. The programs have been written in BASIC and the use of the PET computer is straightforward. Therefore, it requires only a matter of hours to teach medical personnel how to use the computer and programs.

In the application of the anticoagulation control problem, this model appears to be adequate for rapidly identifying patients who become refractory to anticoagulation therapy, estimating the time to reach the therapeutic range, and determining the proper dosage schedules to maintain the desired prothrombin times. Furthermore, this approach gives a systematic method for dealing with hard-to-control patients, and alerting the physician early in the therapy course that a particular patient may be difficult to anticoagulate.

Acknowledgement

To William F. Powers for his assistance in supplying the programs.

References

1. W.F. Powers, P.H. Abbrecht and D.G. Covell, "Systems and Microcomputer Approach to Anticoagulation Therapy," *IEEE Trans Biomedical Engineering*, Vol. 27, pp. 520-523, 1980.
2. W.T. Sawyers and A.L. Finn, "Digital Computer Assisted Warfarin: Comparison of Two Models," *Computers and Biomedical Research*, Vol. 12, pp. 221-231, 1979.
3. H. Wiegman and A. Vossepel, *A Computer Program for Long Term Anticoagulation Control*, Vol. 5, p. 441, 1972.
4. W.F. Powers, *Microcomputer Approach to Anticoagulation Therapy*, University of Michigan, Ann Arbor, Rep. AE 80-1, February 1980.

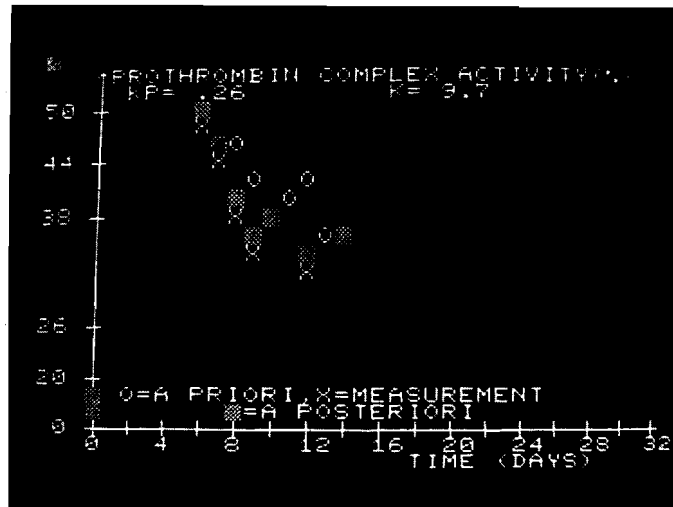


Figure 2: Graphical display of the extended Kalman Filter values.

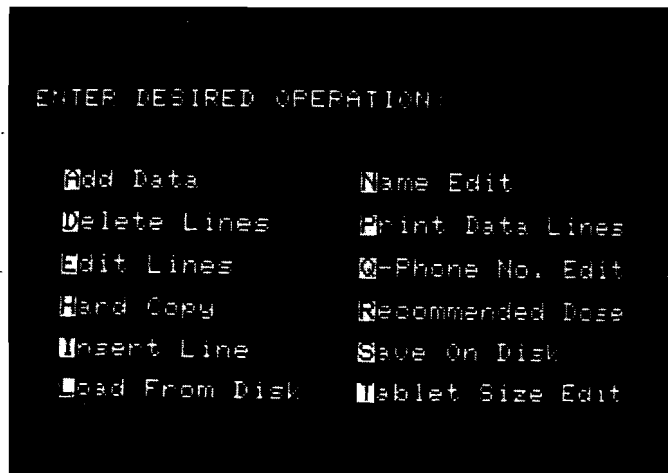


Figure 3: Example of the Menu selections from the MAINTENANCE program.

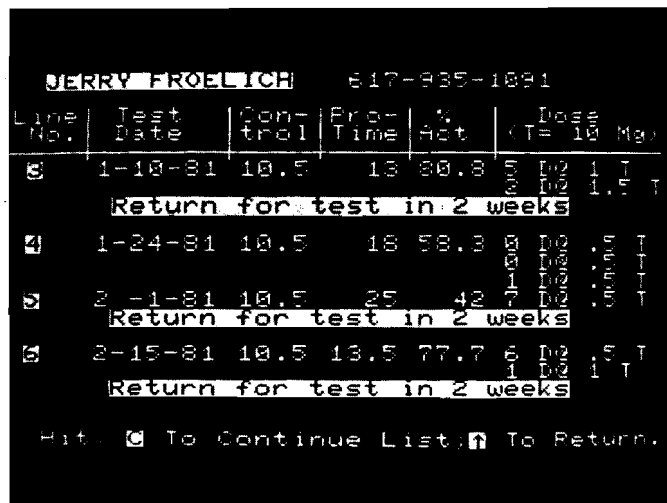
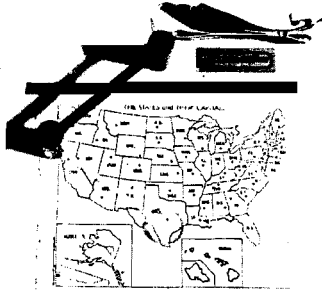
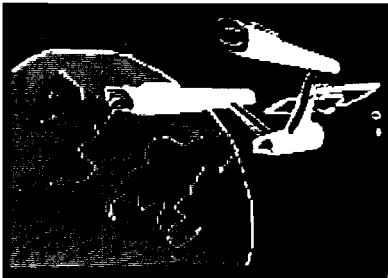


Figure 4: Example of screen display for recommended therapy from the maintenance program.

VersaWriter & APPLE II: The Keys to Unlimited Graphics

DRAWING TABLET

Although VersaWriter operates on a simple principle, it produces graphics which match or exceed those of other digitizers. Rugged construction, translucent base, easy to use — plugs directly into APPLE II.



GRAPHICS SOFTWARE

Easily the most capable and complete graphics software for the home computer available. Fast fill drawings in 100 colors. All text in five sizes, compile and display shapes, edit, move and much more!



UNIQUE OFFER

See VersaWriter at your local dealer and pick up a copy of our demonstration disk. The complete VersaWriter hardware and software package is a real bargain at \$249. For more information call or write:

Versa Computing, Inc. • 887 Conestoga Circle • Newbury Park, CA 91320 • (805) 498-1956

SPECIAL INTRODUCTORY OFFER

Programmable Character Generator Board \$89.95

You can use OSI's characters or you can make your own. Imagine you can now do true high resolution graphics 512 x 256 dots in the 64 x 32 screen format. And all under your control!

Other mods available — send for catalog.

SOFTWARE (with Documentation)

PC Chess V1.9 \$14.95

Play Chess against your computer!

Helicopter Pilot: (64 CHR Video Only) \$ 8.95

An Excellent Graphics Program!

Golf Challenger \$14.95

From 1 to 4 players. Play a round of golf on your 18 hole golf course. One of the best programs I have ever seen! You can even design your own course. Comes with full documentation (14 pages).

Two Very Intricate Simulations!

Wild Weasel II: You operate a Sam Missile base during a Nuclear War. Not as easy as you think! You must operate in a three dimensional environment.

Fallsafe II: The shoe is on the other foot! Here you are in the attacking bomber and you must penetrate deep into enemy territory. Can you survive? An extremely complex electronic warfare simulation! **SPECIAL:** both for 19.95

Hardware: C1P Video Mod: Makes your 600 Video every bit as good as the 4P and 8P. Gives 32/64 CHR/Line with guardbands 1 and 2 Mhz. CPU clock with 300, 600 and 1200 baud for Serial Port. Complete Plans \$19.95

KIT(Hardware and Software) \$39.95

Installed: 32CHR — \$79.95, 64CHR-\$89.95

Extra K of Video RAM for 64CHR not included!

Set of 3 ROMs available \$75.00

C1P Sound Effects Board: Completely programmable!

For the discriminating hobbyist, the best board on the market for creating sound and music. Can be interrupt driven so that you can use it for gaming purposes. Has on board audio amp, 16 bit interval timer, 128 Bytes of RAM and two 8 bit parallel I/O Ports.

Assembled and tested \$89.95 Bare Board \$39.95

Both include Prog. Manual and Sample Software.

C1P HI Speed Cassette Kit: Gives a reliable 300, 600, and 1200 Baud. No symmetry adjustments — the ideal fix for OSI's cassette interface. Easily implemented in 30 minutes. Will save you time and money even the first night you use it! \$12.95

Many, many more. Send for Catalog with free program (Hard Copy) and BASIC Memory Map. \$1.00. Two locations to serve you:

Progressive Computing
3336 Avondale Court, Windsor, Ontario
Canada, N9E 1X6
(519) 969-2500

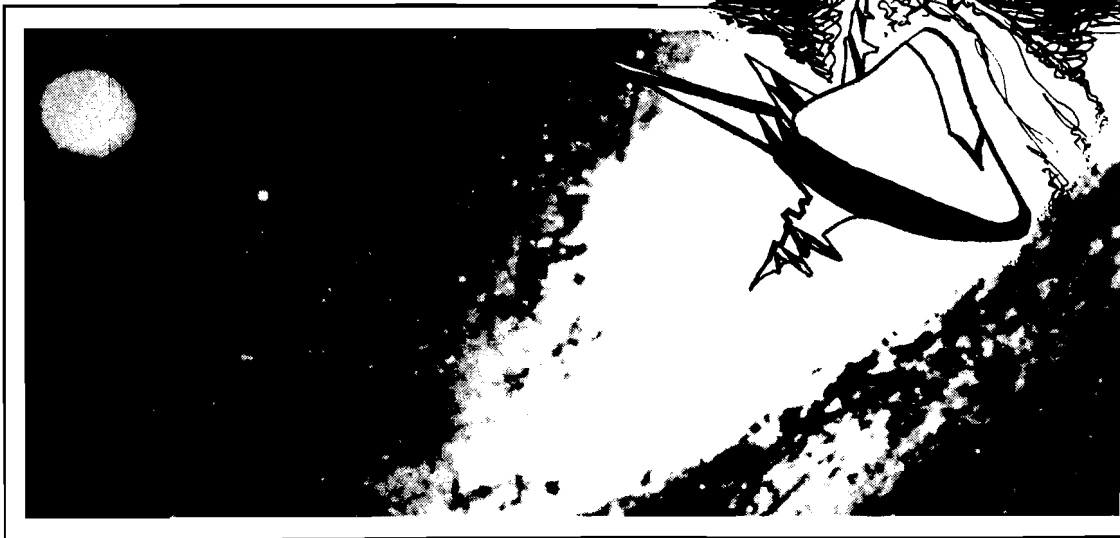
or

3281 Countryside Circle, Pontiac TWP, MI 48057
(313) 373-0468

VISA

MASTER CHARGE

**Hot pursuit
through space
and the
vortices
of time!**




RAITHUARE PRESENTS...

Time Lord

The fallen Time Lord, who presumptuously calls himself The Master, is at large. The elders of Waldrom have supplied you with the hyperspace-worthy vessel Tardus, and commissioned you to eliminate the evil "Master". Your resources include clones who will fight for you, the formidable CRASER weapons of the Tardus, and magic weapons such as Fusion Grenades and Borelian Matrix Crystals.

Traveling through hyperspace in search of the evil one, you will encounter Time Eaters, Neutron Storms, and other alien creatures and phenomena. Entering real space to search planets, you will encounter still other dangers. You will enter native settlements to buy food and supplies — or to fight for survival.

And once you find The Master can you destroy him?


TSE: HARDSIDE
6 South St., Milford, NH 03055 (603) 673-5144
TOLL FREE OUT-OF-STATE 1-800-258-1790

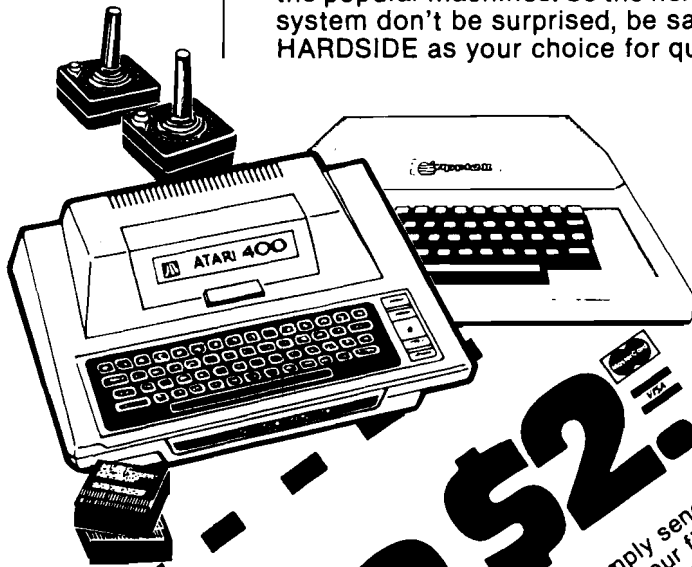
Based on Dr. Who of PBS fame.
Apple Integer Basic,
Disk, 48K ... \$29.95





TSE-HARDSIDE HAS IT ALL IN ONE!

How many times have you wished that there was a single source for your personal computer needs? Well look no further, TSE-HARDSIDE, located in pleasant New Hampshire, has virtually every conceivable item for your micro. Whether you're shopping for your Apple, Pet, TRS-80™ or Atari, TSE-HARDSIDE has it all. We stock hardware, software, books, magazines and specialty items for all of the popular machines. So the next time you're out shopping for your system don't be surprised, be satisfied. Remember the name TSE-HARDSIDE as your choice for quality, service and reliability.



TSE-HARDSIDE
 6 South St., Milford, NH 03055 (603) 673-5144
 TOLL FREE OUT-OF-STATE 1-800-258-1790

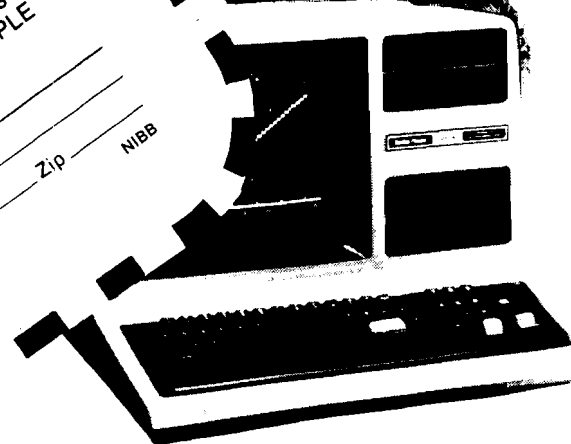


Save \$2.00

If you or a friend would like to receive our TSE/HARDSIDE Complete Computer Catalog, simply send \$1.00 and receive a \$2.00 certificate good toward your first purchase of software, hardware, books or specialty items from TSE/HARDSIDE. I have a (check one) TRS-80, APPLE, ATARI, PET. A photocopy of this coupon will be accepted.

Name _____
 Address _____
 City _____ State _____ Zip _____
 NIBB _____

MAIL TO: TSE/HARDSIDE
 6 South Street
 Milford, NH 03055



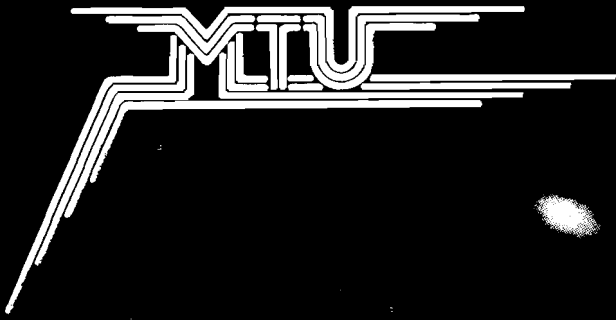


Photo credit: GREAT GALAXY IN ANDROMEDA: Palomar Observatory, California Institute of Technology

THE MTU FLOPPY DISK CONTROLLER WITH 16K RAM GIVES YOUR AIM-65 ION DRIVE POWER!

HARDWARE

- 16K 2 PORT RAM ONBOARD WITH WRITE PROTECT
- USES THE NEC-765 DISK CONTROLLER CHIP
- ROM BOOTSTRAP LOADER SPEEDS LOADING
- DMA OPERATION ALLOWS INTERRUPTS
- SUPPORTS 8 INCH DRIVES 1 OR 2 SIDED
- MAXIMUM STORAGE IS 4 MEGABYTES
- ANALOG PLL DATA SEPERATOR

SYSTEM FEATURES

- FORMAT UTILITY LOGS OUT DEFECTIVE SECTORS
- DISK/FILE COPY WITH WILDCARD SELECTION
- SYSTEM CUSTOMIZATION UTILITY
- VISIBLE MEMORY TERMINAL DRIVER PROVIDED
- INTERCHANGE CODOS SOFTWARE AMONG KIM, SYM, AIM, PET SYSTEMS
- IN FIELD USE FOR OVER 6 MONTHS

CODOS SOFTWARE

- CODOS DISK OPERATING SOFTWARE
- 8K RAM RESIDENT ALLOWS UPGRADES
- FINDS AND LOADS 32K BYTES IN 3 SECONDS
- STARTUP FILE EXECUTES AT BOOT-UP
- COMMAND FILE EXECUTION FROM DISK
- DYNAMIC DISK STORAGE ALLOCATION
- DEVICE-INDEPENDENT I/O
- TRUE RANDOM ACCESS TO RECORD IN ONE ACCESS
- MONITOR WITH 29 BUILT-IN COMMANDS
- FULL ENGLISH ERROR MESSAGES
- FILE NAMES 12 CHARACTERS + EXTENSIONS
- FILE SIZE UP TO 1 MEGABYTE
- UP TO 247 FILES PER DISK DRIVE
- INDIVIDUAL WRITE PROTECT ON FILES
- WORKS WITH AIM EDITOR, ASSEMBLER, BASIC AND MONITOR ROMS
- SUPERVISOR CALLS AVAILABLE TO USER PROGRAM

K-1013M Hardware Manual-\$10, K-1013-3M CODOS manual-\$25, K-1013-3D RAM/Disk controller with CODOS-\$595. Floppy drives, cables, power supply also available.

MASTERCARD & VISA accepted

WRITE OR CALL TODAY FOR OUR 48 PAGE FALL 1980 CATALOG DESCRIBING ALL MTU 6502 PRODUCTS, INCLUDING 320 BY 200 GRAPHICS, AIM GRAPHIC/TEXT PRINT SOFTWARE, BANK-SWITCHABLE RAM/ROM/I-O, AIM CARD FILE, POWER SUPPLY AND MORE!

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

How Microsoft BASIC Works

What is a variable? How are variables manipulated? This article gives the answers to both of these questions and discusses the similarity of FNx definitions to variables as well.

Greg Paris
625 Main St., #642
Roosevelt Island
New York, New York 10044

All computer languages are, to some extent, symbolic in nature. This means that addresses, constants and variables may be used throughout a program and be manipulated by their labels, instead of using absolute or true values. Although the use of symbols is often merely convenient — as in assembler texts — in many circumstances the concept permits manipulations which otherwise would be impossible. Algebraic variables in BASIC or FORTRAN are just one important case. For these reasons, how a computer language defines and manipulates symbols is fundamental to the structure and operation of whatever interfaces between the user and the opcodes — an interpreter, compiler, etc.

The varieties of symbol types allowed in any language determine, to a great extent, the power of that language to solve certain programming problems. The inherent accuracy of mathematical calculations is another example where the format of variable storage is critical.

For these reasons, a logical first step in dissecting the operation of the BASIC interpreter is to find out how it defines its symbols, and how it stores them. The actual arithmetic and string manipulation is more complicated, and will be left for a later article.

This article is organized as follows. First, I will make a few definitions. This will level out most readers'

backgrounds, and obviously may be skipped if you know the jargon. Next I will describe the actual formats of both numeric and string variables. Then I will give a brief discussion of how BASIC uses RAM. Finally, I will combine all of the above to describe variable storage formats, and explain their coding.

Definitions

I caution the more advanced reader that I am not a software development engineer, and may not use the approved industry-standard terminology.

Legal Variable Name: The BASIC manual defines a legal variable name to be "any alphabetic character, and [it] may be followed by any alphanumeric character... Any alphanumeric characters after the first two are ignored." In addition, one cannot embed reserved words into the variable name (A\$ and AAAAAA are legal variable names; %A is not, and neither is AGOTO).

Variable: To the interpreter, a variable is anything that is not an array (no joke!). Any time you need to refer to only one number, or one string, or one whatever, it will be called a variable. For example, X1 is a floating-point (or FP) variable, X1% is an integer variable, and X1\$ is a string variable. They are stored in different ways internally so the interpreter cannot be confused by these three identical variable symbols. You may be confused however, so use caution in such cases.

Array: An array is any group of variables which is referred to by a common legal variable name, followed by a list of subscripts — also called indices. The BASIC manual sometimes refers to arrays as "matrices." An array may contain either integer or FP numeric data or strings, but no more than one type per array. You are, in theory,

allowed 255 subscripts; the real restriction is the line length which limits you to twenty or so. For example, DIM X1(2) allots space for a singly-subscripted FP array, and has room for 3 numbers — X1(0), X1(1), and X1(2). Further, DIM X1%(20) allots space for an array of 21 integer variables, and DIM X1\$(10,3) partitions space for a doubly-subscripted array of 44 $[(10+1) \times (3+1)]$ different strings. [A technical note: if an array is not dimensioned before it is used, the interpreter will automatically execute a DIM command and thus assign each subscript the default value of 10.]

Header: I define a header as any information about a variable (how it is stored or referred to) that is stored along with the data to which it refers. For example, if the interpreter requires information about an array, including its size, how many subscripts, and the values of those subscripts, then the interpreter will group all this information, along with the variable name, into a header — the small block of "data" which immediately precedes the real data in the array. A header may be as short and simple as the 2 bytes of an encoded variable name, or as detailed as the example just given.

.WOR Address Format: When a 16-bit address is to be stored in an 8-bit machine, it can be stored first byte (MSB) first, second byte (LSB) second, or in the reverse order. In assembler notation, the MSB-first arrangement is often referred to as ".DBY" (for "Double BYte"), whereas the reversed order — LSB-first — is called ".WOR" order (for "WORD"). Almost all addresses handled by the BASIC interpreter are stored in .WOR format, including those that may be embedded in headers.

Numeric Variables

There are two types of numeric data allowed in BASIC: integer and floating-point (FP). An integer number is stored

in two bytes, and can represent any integer between +32,767 and -32,768. An FP number is stored in 5 bytes (4 bytes on OSI) and can represent numbers between $\pm 1.7 \times 10^{38}$ and $\pm 2.94 \times 10^{-39}$, and zero. This format for FP numbers allows at least 9 decimal digit accuracy at all times.

Since FP arithmetic as done by the BASIC interpreter is not germane, I will not detail its function in this article. Suffice it to say that there exists, in zero-page RAM, temporary storage areas for two FP numbers. The one most used is the floating-point accumulator (or FPA) and is located at the addresses shown in figure 1-A. The FPA is five to seven bytes long — the second byte of the FPA contains the sign of the mantissa, which is incorporated into the leftmost bit (MSB) of the mantissa whenever a number is removed from the FPA. (The use of this bit for the sign

need not confuse you, since in the FPA this bit is *defined* as being set, unless the number equals zero. Therefore, if it will *always* be 1, then it can be ignored during storage and used for another purpose, namely, to store the sign of the mantissa compactly.) In addition, there is a byte (see figure 1-A) which actually extends the FPA mantissa by 8 bits. It is used internally in all arithmetic operations, but is rounded off and stripped whenever a variable is removed from the FPA. The first byte of the FPA is the exponent of the number plus \$80. If the number equals zero, then this byte is zero.

Both types of variables, if referred to before being assigned a specific numeric value (i.e., if you use a previously undefined variable), will be filled with 0's — hence, the default value in each case is zero.

String Variables

The "value" of a string variable, and the information stored in a string variable (or array) in RAM, are two different things. The two items actually stored in the "variable" or "array" are a pointer (or a list of pointers) in .WOR format to the start of the string, and the length of the string. The string may be embedded in a program line, or stored in "top free space" (high RAM).

If the string is empty ("null"), then the byte for string length is set to zero, and although it will then be ignored, both bytes of the pointer are zeroed. The size of any string is limited to 255 characters because a single byte is used to indicate its length.

Figure 1-A: Locations of Floating-Point Accumulators.

Computer:	AIM 65	Applesoft	OSI (BASIC- in-ROM)	Old PET (1.0)	New PET (2.0, 4.0)
Length of FPA	6 bytes	7 bytes	5 bytes	6 bytes	6 bytes
Address of FPA	\$00A9-\$00AE	\$009D-\$00A3	\$00AC-\$00B0	\$00B0-\$00B5	\$005E-\$0063
FPA extension	\$00B8	[\$00A3]	\$00B2	\$00B7	\$0065

Figure 1-B: BASIC Utility Pointers.

Computer:	AIM 65	Apple	OSI (BASIC-in-ROM)	Old PET	New PET
Address of pointer to:					
Start of BASIC program (address:)	\$0073 (\$0212)	\$0067 (\$0801)	\$0079 (\$0301)	\$007A (\$0402)	\$0028 (\$0402)
Start of variable storage (\$PPPP)	\$0075	\$0069	\$007B	\$007C	\$002A
Start of array storage (\$RRRR)	\$0077	\$006B	\$007D	\$007E	\$002C
Start of free space (\$UNUN)	\$0079	\$006D	\$007F	\$0080	\$002E
Top (end) of free space (\$TTTT)	\$007B	\$006F	\$0081	\$0082	\$0030
Top of memory (\$NONO)	\$007F	\$004C	\$0085	\$0086	\$0034

User Functions

DEF and FNx are BASIC program statements which allow a user to define a unique function. Each FNx is labeled by a legal variable name, and this is why I discuss this statement in an article on variables. As detailed later, the BASIC interpreter stores a reference to each function definition in a complex header, filed under the variable name which is assigned to it by the user.

How BASIC Uses RAM

A memory map of how BASIC partitions space for its various needs is shown in figure 1-B. "Top free space" may be a new term to some readers. When BASIC is commanded to operate on strings, it designates an area in unused memory as work space (from \$UNUN to \$TTTT - 1), and then stores the result of any operation in "top free space" (from \$TTTT to \$NONO - 1).

Also listed in figure 1-B are the zero-page locations which are reserved by BASIC to store pointers to various addresses which are used frequently. These pointers are initialized upon entry into BASIC, and are updated any time the program is changed or run. All pointers are stored in .WOR format.

How Variable Names are Encoded

BASIC reserves 2 bytes for the variable name (symbol). However, since the same name could refer either to an integer, FP variable, or a string, it must distinguish between them. It does this by setting or clearing, in various combinations, the otherwise unused leftmost bit (MSB) of each of the two bytes in the name. All four possible permutations are used. The interpreter performs this encoding during a RUN whenever a new variable name is encountered, and uses the format described in table 1. If a variable name is only a single character, then the second character space allotted to it is filled with 0's, except for the MSB, which is set or cleared as needed.

Storage Formats

Most of the details of variable format and variable name encoding have been described. All that remains is to put the information together and describe what is actually found in memory from \$PPPP to \$UNUN - 1.

Table 1: Format of encoding different types of variable names.

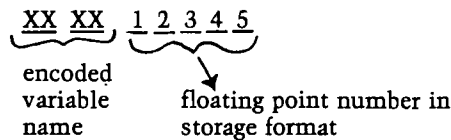
If the legal variable name is AC, then:

if the variable is	then the symbol is encoded as these two bytes:
a floating point numeric (no suffix)	\$41, \$43 (MSB each byte clear)
an integer numeric (suffix = %)	\$C1, \$C3 (MSB each byte set)
a string (suffix = \$)	\$41, \$C3 (MSB first byte clear, MSB second byte set)
an FNx definition variable	\$C1, \$43 (MSB first byte set, MSB second byte clear)

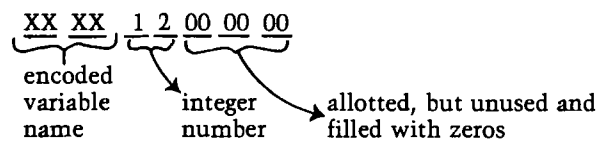
Figure 2: Variable and Array Storage Formats

VARIABLES:

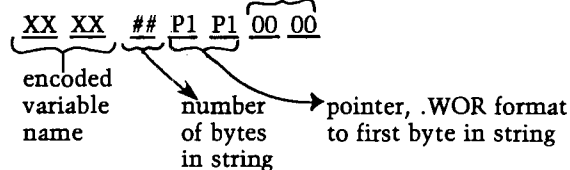
Floating Point Numeric



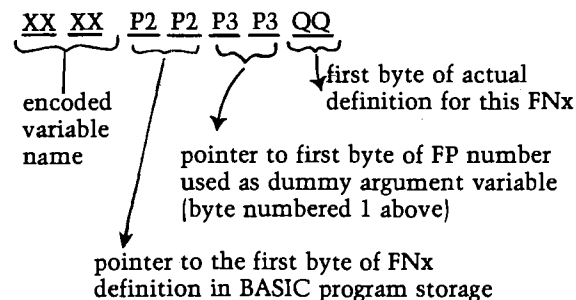
Integer Numeric



String Header



FNx Header



Variables are stored together, but separate from the arrays. However, integer numeric, FP numeric, string, and FNx definition variables are all intermixed. Arrays are stored in the next higher allocated RAM, and are also intermixed. In both cases, the jumbled order is actually a function of when they are defined during the RUNNING of a program. Each variable or array that is interpreted is assigned a space in the order in which it is encountered, with the variables and the arrays each shuttled off to their respective spaces.

There is a reason for separating variables from arrays. Each item stored as a variable takes up exactly 7 bytes. This makes searching for variables very easy, as the interpreter's variable pointer need only increment by 7 bytes to look for the next variable. Since arrays can vary greatly in size, this technique is not applicable, and scanning for individual array entries is somewhat more time consuming.

Each time the program begins RUNNING, it executes a CLEAR instruction, which erases any reference to any variables and arrays which may have previously been defined. This CLEAR instruction sets the pointers located at \$0075, \$0077, and \$0079 (on the AIM) to the same value — the address of the last byte of program storage, plus one. Similarly, the pointer at \$007B ("top free space") is set to equal the address in \$007F (top usable memory + 1).

The headers for variables and arrays, and the formats in which they are stored in RAM, are shown in figure 2.

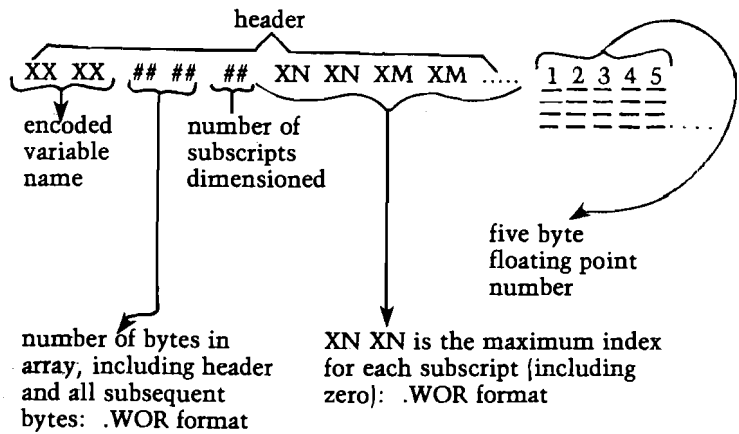
The definition of a header should be clearer now. In both types of numeric variables, the header is simply the 2 bytes of the encoded variable name. More complicated arrangements are seen in the FNx header and the various array headers.

Variables: For an FP variable, all 7 bytes are utilized. The last 5 bytes represent the FP number, in RAM storage form as described above.

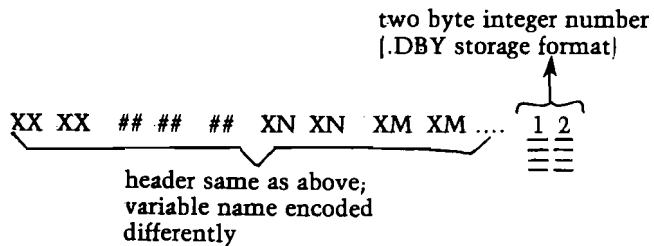
Figure 2 continued

ARRAYS:

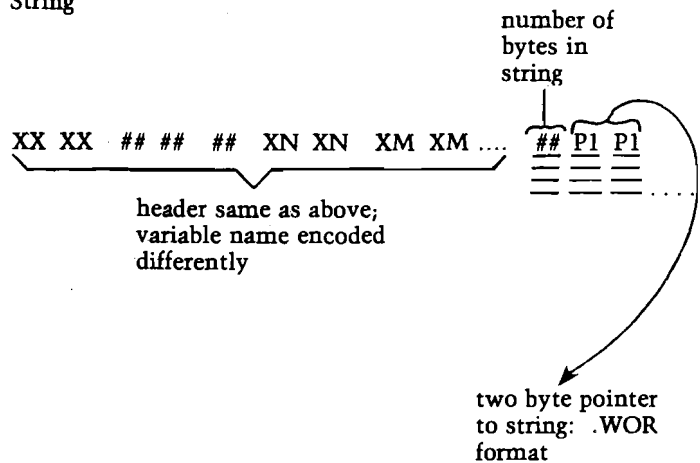
Floating Point Numeric



Integer Numeric



String



Legend for Figure 3

- A. Test program in BASIC.
- B. Zero page pointers to partitions in RAM (see figure 1-a).
- C. Dump of tokenized test program (partial).
Note that D\$ is found at \$025B, and the definition of FNQ at \$0241.
- D. Dump of variable and array storage.
Note that the order of space assignment is identical to the discovered order in the program.
- E. Contents of "top free space", includes 'value' of E2\$, found at \$0FF1.

A.

```

10 DIM AA(2),B%(2,3)
20 AA=2:B%=17
30 DEF FNQ(X)=X*AA
40 C=5.7207
50 D$="A STRING"
60 DIM C(2)
70 F%=-24
80 E2$="IS NOT "+D$
90 STOP

```

B.

```

<M>=0073 12 02 BASIC PROGRAM STARTS AT $0212
< > 0075 98 02 VARIABLES START AT $0298
< > 0077 D0 02 ARRAYS START AT $02D0
< > 0079 1D 03 FREE SPACE STARTS AS $031D
< > 007B F1 0F FREE SPACE ENDS AT $0FF1
< > 007F 00 10 TOP OF MEMORY IS $1000

```

C.

```

<M>=0212 26 02 NEXT LINE IS AT $0226
< > 0214 0A 00 THIS IS LINE 10
< > 0216 85 20 'DIM' TOKEN, SPACE
< > 0218 41 41 'AA'
< > 021A 28 32 '(2'
< > 021C 29 2C '), '
< > 021E 42 25 'B%'
< > 0220 28 32 '(2'
< > 0222 2C 33 ',3'
< > 0224 29 00 ')', END OF LINE
< > 0226 35 02 NEXT LINE IS AT $0235
< > 0228 14 00 THIS IS LINE 20
< > 022A 41 41 'AA'
< > 022C AC 32 '=' TOKEN, '2'
< > 022E 3A 42 ':B'
< > 0230 25 AC '%', '=' TOKEN
< > 0232 31 37 '17'
< > 0234 00 END OF LINE
< > 0235 46 02 NEXT LINE IS AT $0246
< > 0237 1E 00 THIS IS LINE 30
< > 0239 95 20 'DEF' TOKEN, SPACE
< > 023B 9F 51 'FN' TOKEN, 'Q'
< > 023D 28 58 '(X'

```

An integer variable only uses 4 of the 7 bytes allotted to it. Use of integer variables in your program therefore wastes some space, but could save time during interpretation.

The string "variable" has a 5-byte header, made to fill 7 bytes by tacking a bunch of zeros on the end.

The FNx header is very interesting. It is filed as a variable because it is defined with a variable name. Any legal variable name may be used as its label. In addition, any legal variable name may be used as the dummy argument variable, even one used elsewhere in the program, because before the interpreter evaluates an FNx statement, it saves the value which was originally stored in the dummy variable on the stack. If the dummy variable is a new variable, it is automatically created, allotted 7 bytes of space after the FNx header, and appropriately labeled as an FP variable. The FNx header is set up whenever a DEF FNx is performed. If this particular FNx is later redefined, only the original header is changed. The last byte in the header might not be used by the interpreter; it seems to be there only to clear the stack completely during the DEF FNx operation.

Arrays: Not only do arrays have longer headers, but they also utilize space more efficiently. There is no minimum allotment of space, and consequently, no filler bytes are necessary. FNx arrays are not supported in this version of BASIC.

The headers for each type of array are essentially identical in format and content. The first two bytes are the encoded array name (see table 1). The next pair of bytes is a 16-bit number (.WOR format), the total number of bytes in the array. This includes the header with all its subscripts spelled out, and all the space allotted for the variables or string pointers. The fifth byte represents the number of subscripts used. The remainder of the header is a list of subscripts — a series of 16-bit numbers in .WOR format, one for each subscript — in an order that is the REVERSE of the listed order in the DIM statement.

The actual storage format of the array contents is much the same as for a single variable. Each member of an FP array is allotted five bytes for storage, and each member of an integer array is allotted two bytes. Therefore, in contrast to an integer variable, using integer *arrays* not only saves interpreting time but also a tremendous amount of space as well. Each entry in a string array is allotted three bytes, as before.

Within the array, individual members are ordered in straightforward fashion, but not as simply as you'd expect. Just as in the array header, the individual members of an array are in a "reversed" ascending sequence. For example, if the statement DIM A(2,4) has been executed, then the order of members in the array is A(0,0), A(1,0), A(2,0), A(0,1), A(1,1), A(2,1), ..., A(1,4), A(2,4). By analogy, this can be extended to any number of subscripts.

An example is seen in figure 3. This program is intended only to demonstrate variable and array assignment. Note that all the pointers — FNQ and strings — point to the beginning of their respective referents. All the variables are ordered in the sequence in which they were interpreted; the arrays are similarly arranged in higher RAM. Note the encoded variable names for each assignment.

Summary

The following conclusions are of interest to anyone wishing to save execution time and/or memory space. 1) The use of an integer *variable* is generally a waste, for two reasons: the integer must be defined by a "%" each time it occurs (at the cost of 1 byte per occurrence), and, since it takes up 5 bytes anyway, even this doesn't save space. 2) An integer *array* really does save space, if it is of sufficient size. 3) You can save a few bytes, and shorten execution time slightly, by using as a dummy argument variable one that has already been used in the program. Its actual value will not be lost during the execution of an FNx.

These storage formats are not specific to one machine, and apply to those versions of Microsoft BASIC which are used on AIM, SYM, PET, OSI, Apple, etc.

```

< > 023F 29 AC ')', '=' TOKEN
< > 0241 58 A6 'X', '*' TOKEN
< > 0243 41 41 'AA'
< > 0245 00 END OF LINE
< > 0246 53 02 NEXT LINE IS AT $0253
< > 0248 28 00 THIS IS LINE 40
< > 024A 43 AC 'C', '=' TOKEN
< > 024C 35 2E '5.'
< > 024E 37 32 '72'
< > 0250 30 37 '07'
< > 0252 00 END OF LINE
< > 0253 65 02 NEXT LINE IS AT $0265
< > 0255 32 00 THIS IS LINE 50
< > 0257 44 24 'D$'
< > 0259 AC 22 '=' TOKEN, ''
< > 025B 41 20 'A '
< > 025D 53 54 'ST'
< > 025F 52 49 'RI'
< > 0261 4E 47 'NG'
< > 0263 22 00 ''', END OF LINE

```

D.

```

<M>=0298 41 41 FP VARIABLE 'AA'
< > 029A 82 00 VALUE IS 2
< > 029C 00 00
< > 029E 00
< > 029F C2 80 INTEGER VARIABLE 'B'
< > 02A1 00 11 VALUE IS 17
< > 02A3 00 00
< > 02A5 00
< > 02A6 D1 00 FN 'Q'
< > 02A8 41 02 DEFINED AT $0241
< > 02AA AF 02 DUMMY VARIABLE VALUE AT $02AF
< > 02AC 58
< > 02AD 58 00 FP VARIABLE 'X'
< > 02AF 00 00 VALUE IS 0
< > 02B1 00 00
< > 02B3 00
< > 02B4 43 00 FP VARIABLE 'C'
< > 02B6 83 37 VALUE IS 5.7207
< > 02B8 0F F9
< > 02BA 73
< > 02BB 44 80 STRING VARIABLE 'D' (D$)
< > 02BD 08 8 BYTES OF DATA
< > 02BE 5B 02 AT $025B
< > 02C0 00 00
< > 02C2 C6 80 INTEGER VARIABLE 'F' (F%)
< > 02C4 FF E8 VALUE IS -24
< > 02C6 00 00
< > 02C8 00

```



```

< > 02C9 45 B2 STRING VARIABLE '2' (E2$)
< > 02CB 0F 15 BYTES OF DATA
< > 02CC F1 0F AT $OFF1
< > 02CE 00 00

```

```

< > 02D0 41 41 FP ARRAY 'AA'
< > 02D2 16 00 USES 22 BYTES
< > 02D4 01 1 SUBSCRIPT
< > 02D5 00 03 SUBSCRIPT = 2
< > 02D7 00 00 ARRAY ELEMENTS ARE ALL 0

```

```

< > 02E6 C2 80 INTEGER ARRAY 'B' (B%)
< > 02E8 21 00 USES 33 BYTES
< > 02EA 02 2 SUBSCRIPTS
< > 02EB 00 04 SUBSCRIPT 2 = 3
< > 02ED 00 03 SUBSCRIPT 1 = 2
< > 02EF 00 00 ARRAY ELEMENTS ARE ALL 0

```

E.

```

<M>=OFF1 49 53 'IS'
< > OFF3 20 4E ' N'
< > OFF5 4F 54 'OT'
< > OFF7 20 41 ' A'
< > OFF9 20 53 ' S'
< > OFFB 54 52 'TR'
< > OFFD 49 4E 'IN'
< > OFFF 47 'G'

```

Ed. Note: Integer variables are not supported by OSI and SYM BASIC.

All you need to know about variables is here. Now you can design an UNDIM command, or figure out how to support FNx arrays. Or you can construct your own DATA SAVE and DATA LOAD routines for BASIC, linking them to the USR function, if necessary. What you will need in addition to this article is the knowledge of which BASIC subroutines handle the finding of specific variables, or of specific entries in arrays, and how these subroutines work. I plan to address these and other topics in subsequent articles.

Greg Paris has been doing postdoctoral research in neurobiology, and hopes to program microcomputer-based instrumentation for a living.

MICRO

32 K BYTE MEMORY RELIABLE AND COST EFFECTIVE RAM FOR 6502 & 6800 BASED MICROCOMPUTERS

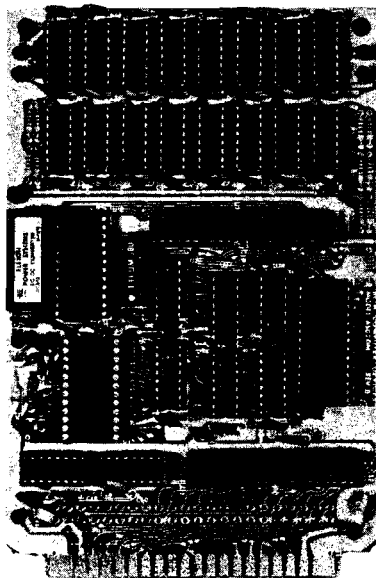
AIM 65-*KIM*SYM
PET*S44-BUS

- * PLUG COMPATIBLE WITH THE AIM-65/SYM EXPANSION CONNECTOR BY USING A RIGHT ANGLE CONNECTOR (SUPPLIED) MOUNTED ON THE BACK OF THE MEMORY BOARD.
- * MEMORY BOARD EDGE CONNECTOR PLUGS INTO THE 6800 S 44 BUS.
- * CONNECTS TO PET OR KIM USING AN ADAPTOR CABLE.
- * RELIABLE—DYNAMIC RAM WITH ON BOARD INVISIBLE REFRESH—LOOKS LIKE STATIC MEMORY BUT AT LOWER COST AND A FRACTION OF THE POWER REQUIRED FOR STATIC BOARDS.
- * USES -5V ONLY, SUPPLIED FROM HOST COMPUTER.
- * FULL DOCUMENTATION, ASSEMBLED AND TESTED BOARDS ARE GUARANTEED FOR ONE YEAR AND PURCHASE PRICE IS FULLY REFUNDABLE IF BOARD IS RETURNED UNDAMAGED WITHIN 14 DAYS.

ASSEMBLED WITH 32K RAM	\$395.00
& WITH 16K RAM	\$339.00
TESTED WITHOUT RAM CHIPS	\$279.00
HARD TO GET PARTS (NO RAM CHIPS)	
WITH BOARD AND MANUAL	\$109.00
BARE BOARD & MANUAL	\$49.00

PET INTERFACE KIT. CONNECTS THE 32K RAM BOARD TO A 4 OR 8K PET. CONTAINS INTERFACE CABLE, BOARD STANDOFFS, POWER SUPPLY MODIFICATION KIT AND COMPLETE INSTRUCTIONS. \$49.00

U.S. PRICES ONLY



16K MEMORY EXPANSION KIT ONLY \$58

FOR APPLE, TRS-80 KEYBOARD, EX10Y, AND ALL OTHER 16K DYNAMIC SYSTEMS USING MK4116-3 OR EQUIVALENT DEVICES.

- ★ 200 NSEC ACCESS, 375 NSEC CYCLE
- ★ BURNED-IN AND FULLY TESTED
- ★ 1 YR. PARTS REPLACEMENT GUARANTEE
- ★ QTY. DISCOUNTS AVAILABLE

ALL ASSEMBLED BOARDS AND MEMORY CHIPS CARRY A FULL ONE YEAR REPLACEMENT WARRANTY

BETA
COMPUTER DEVICES

1230 W. COLLINS AVE.
ORANGE, CA 92668
(714) 633-7280

Calif. residents please add 6% sales tax. Mastercharge & Visa accepted. Please allow 14 days for checks to clear bank. Phone orders welcome. Shipping charges will be added to all shipments.

SYM-1

Communications Interface

This program acts as a traffic cop in a three-way conversation between a SYM, a human at a CRT, and another computer via a modem. It directs messages to either the SYM or the modem on the request of the human operator, and makes sure the human gets to see both ends of the conversation.

Nicholas J. Vrtis
5863 Pinetree S.E.
Kentwood, Michigan 49508

It all started when I wanted to use an acoustic coupler to transfer programs to my SYM from another computer. At first glance, it looks easy enough. Hook the modem to the 20 mA TTY port, and connect the CRT to the RS 232 port. The problem with this arrangement is that although the two devices are electronically separate, the SYM monitor doesn't distinguish between data received from these two ports. There are status bits in TOUTFL (\$A654) which will allow me to control input and output to each of the devices separately, but the SYM still won't tell me where a character came from. There is also a bit in TECHO (\$A653) which can be used to control echoing characters to either the CRT or TTY. The real problem though, is that I only wanted *some* data from the modem to be handed to the SYM monitor. It wouldn't have the foggiest idea what to do with a sign-on request from the other computer. The same is true for data from the CRT, only worse. Some of that had to go to the monitor to tell it to expect a program, and some had to go to the modem to tell the other computer to start sending it. Finally, I wanted to see all the data from both the monitor and modem on the CRT, and didn't want the monitor data to be transmitted by the modem. For the same reason I couldn't give data from the modem to the monitor.

The solution turned out to be shorter than I first expected. About half of the work involved setting up the hardware and control bits the right way, and the other half was writing a short interface routine. Mechanically, the modem and the CRT have to be set up in full duplex mode, so they don't echo any characters. The SYM monitor will take care of that if we set the high order bit of TECHO on. The echo portion of the SYM monitor terminal input routine doesn't care where a bit comes from when it echoes it. Bits four and five of TOUTFL control which device the input byte is echoed to. It doesn't necessarily have to be the one it came from. If we set TOUTFL to enable input from both the CRT and the TTY, but only output to the CRT, then anything from either input device will be echoed to the CRT. Setting TOUTFL this way also means that any output from the monitor will only go to the CRT port, and not to the modem. It also means that anything entered at the CRT will not get transmitted to the modem, so we will have to use software later on to turn the TTY output bit back on when we want it. Finally, this requires that the modem and the CRT are both operating at the same baud rate.

Now that we have the system all wired up, and the bits set, we find that things are arranged so anybody can talk to the CRT, but nobody can talk to anybody else accidentally. Now for a little software to add some smarts to the thing, and we are all set. The SYM vectors all input via an address in System RAM at \$A661, called INVEC. By putting the address of our routine there, it will have a chance to look at all the input and decide what to do with it, to a certain extent. We won't bother with the output side; we just have to be careful not to go to the monitor with the TTY output enable bit on.

There are three characters which have special meaning to these input routines. The BELL character (hex \$07 — control G on my CRT) is used to indicate the start of a string to go to the monitor. The BELL is not sent to the monitor, but all characters following it, up to and including the next carriage return (hex \$0D), do get sent. The semicolon (hex \$3B) is very similar in meaning to the BELL, except that the semicolon itself is also returned to the monitor. This allows the transfer of paper-tape-format hex dumps to the SYM without requiring my program to precede each line with a BELL (and driving me crazy). I left four extra bytes in the program after the compare for the semicolon so you could change it to look for a range of characters. If you patch the following in, you can check for numerics as the key character instead of the semicolon.

CMP #'0'	Check if less than ASCII zero
BCC TRYP	Branch if less
CMP #':'	Colon is ASCII nine + 1
BCS TRYP	Branch if greater or equal

This arrangement would be useful when transferring a BASIC program or other data with line numbers. Watch out for the number of null characters which BASIC needs at the end of a line for timing. The final special input character is the DLE (hex \$10 — control P on my CRT). This performs the modem function corresponding to the BELL for the monitor. The method is different, though. Instead of returning a character via an RTS, the DLE routine causes the TTY output enable bit to be turned on in TOUTFL. When this bit is on, the SYM input routines will echo all characters from the CRT to both the CRT and the modem on the TTY port. As with the BELL, the 'to modem' mode is in effect up to and including

the next carriage return. Unfortunately, there isn't any way to implement a modem equivalent of the semicolon. Once a character has been received from the CRT, there just isn't enough time to turn around and transmit it via the TTY port. It technically could be done, but the person at the CRT would have to make sure that he waited at least one character time between each keystroke. If you type too fast, you end up transmitting garbage.

The special input characters are looked for only when the output hasn't already been directed to either the monitor or the modem. Similarly, the carriage return is only meaningful if one of the output modes is set. Be careful, though, because a carriage return from either the modem or the CRT will reset the flags to output to neither the monitor nor the modem. The special input characters don't have to be at the beginning of a line, so it is possible to have the 'to monitor mode' set accidentally by the other computer. If you know, and/or think that these characters might arrive unexpectedly from the other computer, you may want to change the character looked for in the comparisons to something you probably won't be getting. The routines are not set up to allow a 'to modem' and 'to monitor' mode at the same time. You can have one, or the other, or neither, but not both. If you want a program on the SYM to talk to the modem, simply have it turn on the TTY output bit in TOUTFL before outputting, and turn it off when done.

My final disclaimer is that these routines were not designed for long involved conversations between you and other computers. They were designed merely to transfer programs to the SYM. It can get rather tedious [not to mention noisy] preceding everything with BELL's and DLE.

SYM-1 Modem Communications Interfacer

Theory:

1. The terminal is connected to the CRT RS-232 port, the modem is connected to the TTY port.
2. TOUTFL is set to \$D0 — TTY and CRT input enabled, and TTY output disabled. Therefore any input from either TTY or CRT will appear on the CRT.

```

0800      1  ;*****
0800      2  ;*
0800      3  ;* SYM-1 MODEM COMMUNICATIONS *
0800      4  ;*   INTERFACE ROUTINE   *
0800      5  ;*
0800      6  ;*   BY NICHOLAS J. VRTIS   *
0800      7  ;*
0800      8  ;*****
0800      9  ;*
0800     10  ;*
0800     11  MODFLG EPZ $FA          ;SPARE SYM-1 P.Z. AREA
0800     12  TOUTFL EQU $A654       ;TERMINAL OUTPUT FLAG BYTE
0800     13  INTCHR EQU $8A58       ;TERMINAL INPUT ROUTINE
0800     14  ACCESS EQU $8B86
0800     15  ;*
0800     16          ORG $FC0        ;BACK OUT OF THE WAY
0800     17          OBJ $800
0800     18  ;*
0800     19  MODEM JSR INTCHR        ;GET AN INPUT CHARACTER
0800     20          AND #$7F        ;STRIP PARITY
0800     21          BIT MODFLG      ;CHECK CURRENT MODE
0800     22          BMI TOMON       ;BRANCH IF TO THE MONITOR
0800     23          BVS TOMODM      ;OR IF TO THE MODEM
0800     24  ;*
0800     25          CMP #' '        ;IS THIS FOR THE MONITOR
0800     26          BNE TRYP        ;NO
0800     27  ;*
0800     28          HEX EAEAEAEA    ;PATCH AREA FOR EXTRA COMPARE
0800     29  OFD2 EA
0800     30  OFD3 ;*
0800     31  OFD3 66FA              ROR MODFLG          ;ROLL CARRY INTO FLAG FOR 'TO MONITOR'
0800     32  OFD5 60                RTS                ;AND THIS WILL GIVE IT TO MONITOR
0800     33  OFD6 ;*
0800     34  OFD6 C907              TRYP CMP #$07          ;MONITOR SELECT CODE ??
0800     35  OFD8 D004              BNE TRYS          ;NO
0800     36  OFDA 66FA              ROR MODFLG          ;ROLL CARRY TO SET 'TO MONITOR' BIT
0800     37  OFDC D0E2              BNE MODEM         ;AND IGNORE THIS CHARACTER
0800     38  OFDE ;*
0800     39  OFDE C910              TRYS CMP #$10          ;MODEM SELECT CODE?
0800     40  OFE0 D0DE              BNE MODEM         ;NO—IGNORE THIS CHARACTER
0800     41  OFE2 ;*
0800     42  OFE2 A2F0              LDX #$F0          ;TURN ON TTY OUTPUT ALSO
0800     43  OFE4 A940              LDA #$40          ;YES—TURN ON 'TO MODEM' MODE
0800     44  OFE6 20868B            STFLAG JSR ACCESS ;MAKE SURE CAN UPDATE SYSTEM RAM
0800     45  OFE9 8E54A6            STX TOUTFL       ;STORE NEW FLAG SETTING
0800     46  OFEC 85FA              STA MODFLG       ;STORE NEW MODE SETTING
0800     47  OFEE D0D0              BNE MODEM         ;UNCONDITIONAL—IGNORE THIS ONE
0800     48  OFF0 ;*
0800     49  OFF0 C90D              TOMON CMP #$0D    ;IS THIS NEXT CARRIAGE RETURN?
0800     50  OFF2 D002              BNE **+4         ;NO—PASS IT ON
0800     51  OFF4 85FA              STA MODFLG       ;YES—SET MODE BITS OFF
0800     52  OFF6 60                RTS                ;AND RETURN TO THE MONITOR
0800     53  OFF7 ;*
0800     54  OFF7 C90D              TOMODM CMP #$0D   ;WAS IT A CARRIAGE RETURN?
0800     55  OFF9 D0C5              BNE MODEM         ;NO—IT IS ALREADY ECHOED TO THE TTY
0800     56  OFFB A2D0              LDX #$D0         ;YES—TURN OFF TTY ECHO BIT
0800     57  OFFD D0E7              BNE STFLAG       ;NEW FLAG, $0D TURNS OFF MODE SET
0800     58  OFFF ;*
0800     59  OFFF                    ZZZEND EQU *-1      ;LAST BYTE OF PROGRAM

```

3. TECHO must be set to \$80 so input is echoed to the CRT.
4. THE CRT must be in full duplex mode.
5. Address of 'MODEM' replaces address of 'INTCHR' in 'INVEC'.
6. 'MODEM' is normally waiting to return a character to the monitor via the RTS.
7. The CRT and TTY must be at the same speed.
8. To direct output to the modem from the CRT, the TTY echo bit is turned on in TOUTFL.
9. No direct provision is made for the CPU to talk to the modem.

Functions:

';' gets returned to the monitor and sets 'TO MONITOR' mode.

All following characters to next C/R also go to monitor.

Bell (\$07) does not go to monitor, but does set 'TO MONITOR' mode. All following characters to next C/R also go to the monitor.

DLE [\$10] does not go to modem, but does set 'TO MODEM' mode. All following characters to next C/R also go to the modem.



Annual Index

June 1980—May 1981

(Issues 25 - 36)

Articles

Title/Author	Issue/Page		
AIM			
Share Your AIM Programs <i>Jody Nells</i>	25:23	A Better Apple SEARCH/CHANGE <i>J.D. Childress</i>	32:17
AIM-65 File Operations <i>Christopher J. Flynn</i>	26:61	Make a Clear, Plastic Cover for your Apple <i>E.J. Nelburger</i>	32:53
Satellite Tracking with the AIM-65 <i>C.R. MacCluer</i>	27:13	Searching String Arrays <i>Gary B. Little</i>	33:57
Loading KIM-1 Tapes to AIM <i>Larry P. Gonzalez</i>	28:19	A Simple Securities Manager for the Apple <i>Ronald A. Guest</i>	33:7
Compact <i>Steve Bresson</i>	28:25	In the Heart of Applesoft <i>C. Bongers</i>	33:31
Tiny PILOT for the AIM <i>Larry Kollar and Carl Gutekunst</i>	28:59	UnwrApple <i>David Lubar</i>	34:11
An Improved Morse Code Receive Routine and Interface <i>Marvin L. DeJong</i>	29:23	Reset Protection for the Apple II <i>Joe Brady</i>	34:89
Biorhythm: An AIM BASIC Programming Exercise <i>P.E. Burcher</i>	29:51	S-C Assembled Modifications <i>Ned W. Rhodes</i>	35:7
AIM 65 File Operations: Writing Text Files with BASIC <i>Christopher J. Flynn</i>	30:65	Apple Memory Maps <i>Peter A. Cook</i>	35:27
A Random-Character Morse Code Teacher for the AIM 65 <i>Eugene V. Weiner, Marvin L. DeJong, Russell V. Lenth</i>	31:21	Integer Basic Internals (Apple) <i>Glenn R. Sogge</i>	35:65
AIM 65 File Operations <i>Christopher J. Flynn</i>	32:29	MacApple <i>David Lubar</i>	36:9
One-Dimensional Life on the AIM 65 <i>Larry Kollar</i>	33:50	Applesoft Variable Dump <i>Scott O. Schram</i>	36:23
A Relocating Loader for AIM Tape <i>Mel Evans</i>	34:25	Apple Memory Maps — Part 2 <i>Peter A. Cook</i>	36:45
MEMSEARCH for the AIM 65 <i>Bob Kovacs</i>	35:17	Protecting Memory from DOS <i>Glenn R. Sogge</i>	36:81
APPLE			
A Little Plus For Your Apple II <i>Craig Peterson</i>	25:7	ATARI	
APPLE II Integer BASIC Program List by Page <i>Dave Partyka</i>	25:37	Introducing the Atari 800 <i>William L. Colsher</i>	25:35
BASIC and Machine Language Transfers with Micromodem II <i>George Dombrowski</i>	25:47	Atari Notes <i>William L. Colsher</i>	27:57
TRACER: A Debugging Tool for the APPLE II <i>R. Kovacs</i>	25:59	A Versatile Hi-Res Function Plotter for the Atari 400 & 800 <i>David P. Allen</i>	30:47
Zoom and Squeeze <i>Gary B. Little</i>	26:37	Atari Bits <i>Len Lindsay</i>	31:57
Data Statements Revisited <i>Virginia Lee Brady</i>	27:7	Atari Real Time <i>Charlie and Mary Kozarski</i>	32:35
Better Utilization of Apple Computer Renumber and Merge Program <i>Frank D. Chipchase</i>	27:17	An Atari Assembler <i>William L. Colsher</i>	33:17
Solar System Simulation with or without an Apple II <i>David A. Partyka</i>	27:33	Atari Error Messages <i>David P. Allen</i>	35:69
Applesoft Floating Point Routines <i>R.M. Mottola</i>	27:53	The Atari Dulcimer <i>Mike Dougherty</i>	36:59
Business Dollars and Sense In Applesoft <i>Barton M. Bauers, Jr.</i>	27:65	KIM	
Creating Shape Tables, Improved! <i>Peter A. Cook</i>	28:7	VISA—KIM <i>Joel Swank</i>	26:47
A Versatile Hi-Res Function Plotter for the Apple II <i>David P. Allen</i>	28:49	A "Stop-on-Address" Routine for KIM <i>R. MacDonald</i>	29:30
Mean 14: A Pseudo-Machine Floating Point Processor for the Apple II <i>R.M. Mottola</i>	28:67	Full Disassembly Listing on Small Systems <i>Ralph Tenny</i>	32:37
PRINT USING for Applesoft <i>Gary A. Morris</i>	29:14	Increase KIM-1 Versatility at Low Cost <i>Ralph Tenny</i>	33:57
Paged Printer Output for the APPLE <i>Gary Little</i>	29:47	BASIC Program Converter Between SYM and KIM <i>Lee Chapel</i>	35:79
Cassette Label Program <i>Dawn E. Ellis</i>	29:65	KIM/SYM Home Accounting System <i>Robert Baker</i>	36:13
Step and Trace for the APPLE II Plus <i>Craig Peterson</i>	30:61	OHIO SCIENTIFIC	
Graphing Rational Functions <i>Ron Carlson</i>	31:7	Put Your Hooks Into OSI BASIC <i>Edward H. Carlson</i>	25:15
An Apple Flavored Lifesaver <i>Gregory L. Tibbetts</i>	31:25	Hypocycloids on the 540 <i>E.D. Morris</i>	25:57
Creating an Applesoft BASIC Subroutine Library <i>N.R. McBurney</i>	31:37	Challenger II Communications <i>Peter Koski</i>	26:53
		Interface of OSI C1P With Heath Printer <i>William L. Taylor</i>	27:47
		A C1P and H14 System, Part 2 <i>William L. Taylor</i>	28:30

An OSI Cheep Print <i>Thomas Berger</i>	29:7	SYM	
An Ultra-Fast Tape Storage System <i>John E. Hart</i>	30:11	SYM-1 BASIC Pack Program <i>George H. Wells, Jr.</i>	25:19
Ohio Scientific Users: Stop those S ERRORS <i>E.D. Morris, Jr. and Tim Finkbelner</i>	30:37	Slide Show for the SYM <i>David P. Kemp</i>	25:53
A C1P User's Notebook <i>Robert L. Elm</i>	31:11	SYM-1 Memory Search and Display <i>Nicholas Vrtls</i>	26:7
Relocating OSI ROM BASIC Programs <i>William L. Taylor</i>	31:61	SYM-Bell <i>Randy Sebra</i>	30:17
Vectors and the Challenger 1P <i>Mike Bassman</i>	32:21	Cassette I/O for SYM BASIC <i>Nicholas J. Vrtls</i>	31:65
Fun with OSI <i>Leo Cain</i>	32:75	SYM Bridge Trainer <i>Len Green</i>	32:41
Why WAIT? <i>Robert L. Elm</i>	33:15	Improved Dual Tape Drive for SYM-1 BASIC <i>George Wells</i>	33:23
A C1P Sound Idea <i>David A. Ell</i>	33:71	SYM-ple SYM-on <i>Len Green</i>	34:15
Joysticks for the OSI C4 <i>Charles Platt</i>	35:23	SYM Time-Remaining Timer <i>Ralph Orton</i>	35:37
Oh No—It's Garbage Collect <i>Gordon A. Campbell</i>	35:43	SYM-1 Communications Interface <i>Nicholas J. Vrtls</i>	36:39
Cursor Control for the C1P <i>Kerry V. Lourash</i>	36:75	Tiny Pilot Follow-Up <i>Nicholas J. Vrtls</i>	36:71
PET		GENERAL	
Lower Case Lister <i>James Strasma</i>	25:11	6502 Resource Update <i>Dr. William R. Dial</i>	25:65
PET-16 <i>James Strasma</i>	25:49	Sorting Revealed <i>Richard C. Vile, Jr.</i>	26:13
'Stop That PET' - Update <i>George R. Gaukel</i>	25:64	Variable Lister <i>Ray Cadmus</i>	27:19
Hello, World <i>John Sherburne</i>	26:31	Additions to Tiny Pilot <i>Bob Applegate</i>	27:21
Son of Screen Print <i>Kenneth Finn</i>	27:61	Nth Precision Add & Subtract With Adjusted Processor Status <i>Lawrence R. Golla</i>	27:27
Auto-Run-Save, Y-t Plotter, Canary for the PET <i>Werner Kolbe</i>	28:14	BCD Input to a 6502 Microprocessor <i>Richard Saltero</i>	27:68
Define Your Own Function Key on PET <i>Werner Kolbe</i>	29:19	XREFER <i>Joel Swank</i>	28:34
For Multiple File Tape Backups <i>G.R. Boynton</i>	29:36	Undedicating a dedicated Microcomputer <i>David N. Borton</i>	29:27
Self-modifying PET Programs <i>P. Kenneth Morse</i>	30:29	Tiny Pilot Complementary (Co-Pilot) <i>Robert Schultz</i>	29:32
Drawing a Line on PET's 80 x 80 Grid <i>Harvey S. Davis</i>	31:15	Hexadecimal Printer <i>LeRoy Moyer</i>	29:57
STUFFIT: A Time Saving Utility Program for PET BASIC Files <i>Roger C. Crtes</i>	31:45	Programming with Pascal <i>John P. Mulligan</i>	29:59
PET Symbolic Disassembler <i>Werner Kolbe</i>	32:23	How to Use the Hooks <i>Richard Williams</i>	30:7
PET String Flip <i>James Strasma</i>	33:65	John Conway's Game of Life Using Display Devices With Automatic Scrolling <i>Theodore E. Bridge</i>	30:53
A Second Cassette for PET <i>Jerry W. Froelich</i>	34:81	Multiplying on the 6502 <i>Brooke W. Boering</i>	31:71
PRINT USING for the PET <i>David Malmberg</i>	35:13	Keyboard Encoding <i>George Young</i>	32:7
An Inexpensive Word Processor <i>William F. Pytlak</i>	36:65		

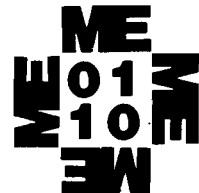
OHIO SCIENTIFIC USERS

SOFTWARE - GAME AND UTILITY PROGRAMS FOR AS LOW AS \$1.00. ALL WITH LISTINGS AND COMPLETE DOCUMENTATION.

KITS - UPDATE YOUR COMPUTER TO PLAY MUSIC, INCREASE OPERATING SPEED, HIGH RESOLUTION GRAPHICS AND MUCH MORE. KITS INCLUDE PARTS AND COMPLETE ASSEMBLY INSTRUCTIONS. LOW AS \$3.00.

OUR \$1.00 CATALOG INCLUDES OSI PROGRAMMING TIPS PLUS DESCRIPTIONS OF AVAILABLE PROGRAMS AND KITS.

MITTENDORF ENGINEERING 905 VILLA NUEVA DR. LITCHFIELD PARK, AZ 85340




Interfacing the 6522 Versatile Interface Adapter <i>Marvin L. DeJong</i>	32:65	Automatic Keyboard <i>Theo Schliff</i>	34:39
Turning USR(X) Routines Into BASIC DATA Statements <i>Thomas Cheng</i>	33:21	The 6502 Dream Machine <i>Randall Hyde</i>	34:67
Does Anyone Really Know What Time It Is? <i>Randy Sebra</i>	33:75	Add a Light Pen to your Micro <i>Peter Alan Koski</i>	35:57
A 6502 Assembler in BASIC <i>Edward H. Carlson</i>	34:7	More Output from Your Micro <i>H.H. Aumann</i>	36:19
Rapid Bubble Sort of Numerical Elements Using BASIC/ASL <i>L.S. Reich</i>	34:21	How MicroSoft BASIC Works <i>Greg Paris</i>	36:31
Encryption With RND and USR <i>Sherwood Hoyt</i>	34:35		

Departments

		Issue/Page		
The MICRO Software Catalog <i>Mike Rowe*</i>	25:71/26:71/27:71/28:73 29:73/30:72/31:79/32:87 33:67/34:93/35:83/36:88	Letterbox	26:60/27:56/29:6/31:59 32:6/33:6/34:6/35:6/36:6	
6502 Bibliography <i>Dr. William R. Dial</i>	25:75/26:75/27:75/28:76 29:76/30:76/31:89/32:90 33:90/34:97/35:89/36:92	Editorial <i>Robert M. Tripp</i>	25:5/26:5/27:5/28:5 29:5/30:5/31:5/32:5 33:5/34:5/35:5/36:5	
MICRO Club Circuit <i>Mike Rowe*</i>	25:68/26:68/27:25/28:47 29:34/30:72/31:75/32:61 34:44/35:15	New Publications <i>Mike Rowe*</i>	31:51/32:36/33:54/34:79/35:47	
MICROscope	27:31/28:57/29:49/30:33/31:43	Challenges <i>Paul Geffen</i>	34:46/35:77/36:17	
Microprocessors in Medicine: The 6502 <i>Jerry W. Froelich, M.D.</i>	29:56/30:36 31:53/34:63/36:25	Microbes	31:76/33:59/34:61/35:81/36:72	
Up From the Basements <i>Jeff Beamsley</i>	27:59/29:72/30:51/31:87	MICRO Dealers	35:51	
PET Vet <i>Loren Wright</i>	28:48/29:39/30:27/31:33 32:51/33:68/34:59/35:55 36:62			

*Mike Rowe is a pseudonym for material prepared by MICRO's staff.

*MICRO's volume year runs from June through May. Issue numbers span volumes consecutively, from MICRO's first bimonthly issue (Oct./Nov. 1977) to the current monthly issue (No. ??).



Also Available soon on Atari

EDIT 6502 T.M. LJK

Two Pass Assembler, Disassembler, and Editor Single Load Program

DOS 3.3., 40/80 Columns, for Apple II or Apple II Plus*

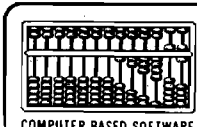
A MUST FOR THE MACHINE LANGUAGE PROGRAMMER. Edit 6502* is a two pass Assembler, Disassembler and text editor for the Apple computer. It is a single load program that only occupies 7K of memory. You can move freely between assembling and disassembling. Editing is both character and line orientated, the two pass disassemblies create editable source files. The program is so written so as to encompass combined disassemblies of 6502 Code, ASCII text, hex data and Sweet 16 code. Edit 6502 makes the user feel he has never left the environment of basic. It encompasses a large number of pseudo opcodes, allows linked assemblies, software stacking (single and multiple page) and complete control of printer (paganation and tab setting). User is free to move source, object and symbol table anywhere in memory. Requirements: 48K of RAM, and ONE DISK DRIVE. Optional use of 80 column M&R board, or lower case available with Paymar Lower Case Generator.

TAKE A LOOK AT JUST SOME OF THE EDITING COMMAND FEATURES. Insert at line # n Delete a character Insert a character Delete a line # n List line # n1, n2 to line # n3 Change line # n1 to n2 "stringl" Search line # n1 to n2 "stringl".


LOOK AT THESE KEY BOARD FUNCTIONS: Copy to the end of line and exit: Go to the beginning of the line: abort operation: delete a character at cursor location: go to end of line: find character after cursor location: non destructive backspace: insert a character at cursor location: shift lock: shift release: forward copy: delete line number: prefix special print characters. Complete cursor control: home and clear, right, left down up. Scroll a line at a time. **Never type a line number again.**

All this and much much more — Send for FREE Information.

Introductory Price \$50.00.



COMPUTER BASED SOFTWARE



ENTERPRISES

LJK Enterprises Inc. P.O. Box 10827 St. Louis, MO 63129 (314)846-6124
*Edit 6502 T.M. of LJK Ent. Inc. — *Apple T.M. of Apple Computer Inc.

consumer computers

formerly Computers 'A' Us

We accept these major credit cards:



mail order

OPEN EVERY DAY 9 to 6 PST

California, Alaska & Foreign orders (714) 698-8088
 Shipping Information or Backorders call (714) 698-0260
 Service Center and for Technical Information (714) 460-6302

ORDER TOLL FREE
800-854-6654



ALL EQUIPMENT IS
 FCC APPROVED.



APPLE II PLUS 16K 1049
 APPLE II PLUS 48K
 (APPLE Memory) 1189
 APPLE II Standard Models... CALL
 DISK II DRIVE & CONTROLLER. 529
 This model includes DOS 3.3 16 sector

TOP FIVE SELLERS
 Language System W/Pascal 425
 Silentyne Printer W/Interface 549
 Hayes Micromodem II 319
 Videx Videoterm 80 w/graphics 335
 Z-80 Microsoft Card 299

APPLE COMPUTER INC.
 Disk II Drive Only 445
 Integer or Applesoft II Firmware Card 155
 Graphics Tablet 649
 Parallel Printer Interface Card 155
 Hi-Speed Serial Interface Card 155
 Smarterm 80 Column Video Card 335

MOUNTAIN COMPUTER INC.
 Music System (16 Voices) 479
 A/D + D/A Interface 319
 Expansion Chassis 555
 Intral/X-10 System 249
 Clock/Calendar Card 239
 Supertalker SD-200 249
 Ramplus+ Card 135
 Romwriter Card 155

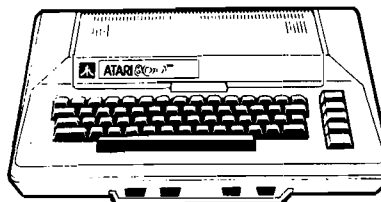
CALIFORNIA COMPUTER SYSTEMS
 Clock/Calendar Module 109
 GPIB IEEE-488 Card 259
 Asynchronous Serial Interface Card 129
 Centronics Parallel Interface Card 99
 We carry all CCS hardware. Please call

MISC. APPLE HARDWARE
 16K Ram Card Microsoft 189
 ADT Numeric Keypad (old or new hybrid) 115
 ALF 3 Voice Music Card 229
 Alpha Syntauri Keyboard System 1399
 Carvus 10MB Hard Disk CALL
 Lazer Lower Case Plus 50
 Micro-Sci Disk Drives CALL
 SSM AIO Serial/Parallel Card AGT 189
 Sup-R-Terminal 80 Cal. Card 339
 SVA 8 inch Floppy Disk Controller 345
 Versawriter Digitizer Pad 229

**WE HAVE MANY MORE ACCESSORIES
 FOR THE APPLE II IN STOCK—
 PLEASE CALL OR WRITE FOR A PRICE LIST.**



MODEL
800 16K
\$799



Atari 400 16K 499
 810 Disk Drive 499
 410 Program Recorder 69
 850 Interface Module 175
 822 Thermal Printer (40 col) 369
 825 Printer (80 col) 795
 Atari 16K Ram Module 155
 Atari Light Pen 65

**We stock all Atari accessories &
 software, please call for more info.**

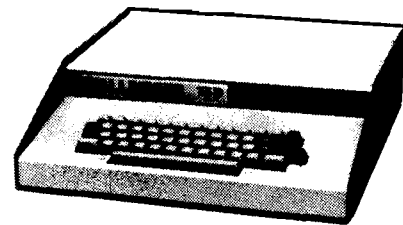
PRINTERS

Anodex DP-9500 W/2k Buffer 1375
 Anodex DP-9501 W/2k Buffer 1450
 C. Itoh Starwriter 25 CPS 1750
 C. Itoh Starwriter 45 CPS 2450
 Centronics 737 825
 Epson MX-70 W/Graphics 449
 Epson MX-80 132 Col. 620
 Paper Tiger IDS-445 W/Dat Plot 749
 Paper Tiger IDS-460 W/Dat Plot 1195
 Paper Tiger IDS-560 W/Dat Plot 1495
 Gume Sprint 5/45 Daisywheel 2550
 Silentyne w/Interface for Apple II 549
 Watanabe Digiplot 1295

VIDEO MONITORS

Amdex/Leedex Video-100 12" B&W 139
 Hitachi 13" Color 389
 NEC 12" P31 Green Phosphor CALL
 Panacolor 10" Color 375
 Sanyo 9" B&W 179
 Sanyo 12" B&W 255
 Sanyo 12" P31 Green Phosphor 295
 Sanyo 13" Color 445

OHIO SCIENTIFIC



Challenger 4P 699
 C4PMF (Mini Floppy System) 1599
 CIP Model II 449
 Sargon II (Disk or Cassette) 35
 Fig Forth (Disk Only) 69

APPLE SOFTWARE

DOS Toolkit 65
 Appleplot 60
 Tax Planner 99
 Apple Writer 65
 Apple Past 45
 D.J. Portfolio Evaluator 45
 D.J. News & Quotes Reporter 85
 Apple Fartran 165
 Apple Pilot 129
 DOS 3.3 Upgrade 49
 Music Theory 45
 The Controller Bus. Sys 519

MISC. APPLICATIONS PACKAGES

Visicalc 125
 Desktop Plan II 169
 CCA Data Management DMS 85
 Easywriter Word Processor 225
 ASCII Express 65
 Super Text II 139
 Programma Apple Pie 119
 The Landlord Apt. Mgmt. Pkg. 649
 Peachtree Business Software CALL
 Tax Preparer by HowardSoft 89
 Applebug Assem/Disassm/Editor 75
 3-D Graphics By Bill Budge 53

GAMES

Flight Simulator 34
 The Wizard and The Princess 32
 Cosmos Mission (Space Invaders) 24
 Sargon II Chess 32
 Hi-Res Football 39
 Adventure by Microsoft 27
 Phantoms Five 39
 Reversal (Othello) 34

**PLEASE CALL OR WRITE
 FOR A COMPLETE
 SOFTWARE LIST.**

ORDERING INFORMATION: Phone Orders invited using VISA, MASTERCARD, AMERICAN EXPRESS, DINERS CLUB, CARTE BLANCHE, or bank wire transfer. Credit cards subject to service charge, 2% for VISA & MC, 5% for AE, DC & CB. Mail Orders may send credit card account number (include expiration date), cashiers or certified check, money order, or personal check (allow 10 days to clear). Please include a telephone number with all orders. Foreign orders (excluding Military PO's) add 10% for shipping all funds must be in U.S. dollars. Shipping, handling and insurance in U.S. add 3% (minimum \$4.00). California residents add 6% sales tax. We accept COD's under \$500. OEM's, Institutions & Corporations please send for written quotation. All equipment is subject to price change and availability without notice. All equipment is new and complete with manufacturer warranty (usually 90 days). We cannot guarantee merchantability of any products. We ship most orders within 2 days.

**WE ARE A MEMBER OF THE BETTER BUSINESS BUREAU AND THE CHAMBER OF COMMERCE
 SHOWROOM PRICES MAY DIFFER FROM MAIL ORDER PRICES.**

PLEASE SEND ORDERS TO:

CONSUMER COMPUTERS MAIL ORDER 8314 PARKWAY DRIVE, GROSSMONT SHOPPING CENTER NORTH LA MESA CALIF. 92041

How? Use our ADVERTISING SOFTWARE! You put it in the APPLE and produce colorful, dynamic ads on the screens of TV sets in your shop window. Even if you are not a shop owner, you can use this software to broadcast messages on TV screens in schools, hospitals, factories, etc. The following message-making programs are available.

SUPER MESSAGE: Creates messages in full-page "chunks". Each name allows statements of mixed typestyles, typesizes and colors, in mixed upper & lower case. Five typestyles are available. They range from regular APPLE characters, up to double-size, double-width characters with a heavy, bold font. Six colors may be used for each different typestyle. Vertical & horizontal centering are available, and word-wrap is automatic. Users can chain pages together to make multi-page messages. Pages can be advanced manually or automatically. Multi-page messages can be stored to disc or recalled instantly.

REQUIRES 48K & ROM APPLESOFT \$ 50.

MULTI-MESSAGE with INTERLEAVED COLOR PATTERNS: Up to 10 messages can be run in sequence. Colorful, dynamic patterns (kaleidoscope or abstract art) can be interleaved between messages, at user option. Consists of 28 crisp, readable characters/lineX4 lines/pageX3 pages of text per message. Characters are 1/8 screen-height and "puff" onto the screen at comfortable reading speed.

REQUIRES 32K & INTEGER BASIC \$ 35.

HI-RES ALPHANUMERIC MESSAGE: Same as Multi-Message above, but has only one message/set and no interleaved color-patterns. Still very good general message-maker!

THE SCROLLING WONDER: 4 brief messages appear in APPLE uppercase characters by "floating" onto the screen from below. Messages enter in random sequence, with random 50% of messages "flash". A multiple-rainbow grand finale ends the program. Very good program to run at point of purchase.

GIANT LETTER: Brilliantly-colored letters, of full screen height, appear one-at-a-time, in sequence, to spell out messages. Successive words have different colors. A running summary of letters, in APPLE characters, appears in the bottom 4 lines of the screen, as the giant letters are presented. Very good program for shop windows.

ALL 3 ABOVE TOGETHER, ON DISK, FOR 32K, INTEGER BASIC \$ 30.

LET APPLE PLOT YOUR DATA AND KEEP YOUR RECORDS TOO!

APPLE DATA GRAPH 2.1: Plots up to 3 superimposed curves on the Hi-res Screen both the X & Y axes dimensioned. Each curve consists of up to 120 pieces of data. Graphs can be stored to disc and recalled immediately for updating. Up to 100 graphs can be stored on the same disc. Great for Stock-market Charting, Business Management, and Classroom Instruction!

REQUIRES 48K & ROM APPLESOFT \$ 40.

APPLE RECORD MANAGER: Allows complete files to be brought into memory so that record searches and manipulations are instantaneous. Records within any file can contain up to 20 fields, with user-defined headings. Information can be string or numeric. Users can browse thru files using page-forward, page-backward or random-search commands. Records can easily be searched, altered or sorted at will. Files can be stored on the same drive as the master program, or on another, if a second drive is available. Records or files can be printed, if desired. Additional modules coming are a STATISTICS INTERFACE, CHECKBOOK, MAILING LIST & DATA-ENTRY.

REQUIRES 48K & ROM APPLESOFT \$ 35.

* All Software above on Disk for APPLE DOS 3.2

How? Order any of the items below, and for each \$100 worth of merchandise ordered, we will give you one of the items at left for FREE!

APPLE ADD-ONS

HAYES MICROMODEM for APPLE	\$ 300.
Z80 SOFTCARD by MICROSOFT	\$ 275.
16K RAMCARD by MICROSOFT	\$ 159.
FORTRAN for APPLE by MICROSOFT	\$ 159.
COBOL for APPLE by MICROSOFT	\$ 599.
BASIC Compiler for APPLE by MICROSOFT	\$ 315.

PRINTERS

CENTRONICS 737 (3 mo. warranty)	\$ 795.
CENTRONICS 737 (15 mo. warranty)	\$ 915.
EPSON MX-70 with TRACTORS & GRAPHICS	\$ 400.
EPSON MX-80 with TRACTORS & 132 Columns	\$ 515.
PAPER TIGER 460G with GRAPHICS & 2K Buffer	\$ 1135.
PAPER TIGER 445G with GRAPHICS & 2K Buffer	\$ 749.

WORD PROCESSING

EZ WRITER PROFESSIONAL SYSTEM for APPLE	\$ 239.
EZ MAILER (Interfaces to EZ WRITER above)	\$ 65.
VIDEX VIDEOTERM (80-Column Card for APPLE)	\$ 295.
VIDEX VIDEOTERM (Same as above with GRAPHICS)	\$ 320.
SUPR'TERM (80-Column Card for APPLE)	\$ 320.

BUSINESS PROGRAMS for APPLE & TR-80 by SPECTRUM SOFTWARE

MICROACCOUNTANT: An ideal package for the very small business, based upon classic T-accounts & Double-Entry Bookkeeping. This efficient program records and produces reports on account balances, general ledger journals, revenues & expenses, 40-column or screen reports. Handles up to 1000 journal entries/month, for up to 300 accounts. Includes a short primer in Financial Accounting.

REQUIRES 48K & ROM APPLESOFT \$ 49.

BUSINESS CHECK-REGISTER with BUDGET: Unique system allows setting up pre-defined purpose & recipient accounts (50 each). Supports unique names too. Rapid-access to check-files with scrolling display & 40-col. printout, if desired. Up to 100 checks/mo. + reconciliation + AUTOMATIC BUDGET VARIANCES!

REQUIRES 48K & APPLESOFT ROM \$ 49.

STOCK MARKET

STOCK MARKET ANALYSIS for APPLE by GALAXY \$ 49.

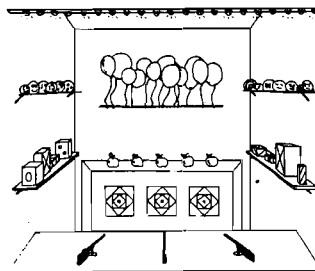
COD'S & Personal Checks are Welcome!

CONNECTICUT INFORMATION SYSTEMS CO.
218 Huntington Road, Bridgeport, CT 06608 (203) 579-0472



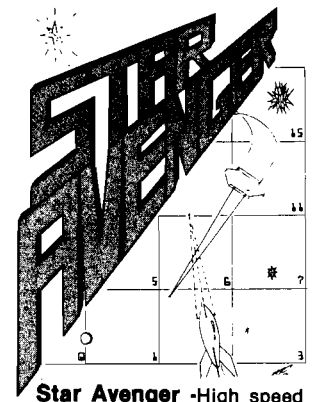
Asteron - The definitive hi-res implementation of the Asteroids arcade game. Features: Ship movement, hyperspace, alien saucers, sound effects, graphic routines allowing up to 25 objects to be displayed with real time response. Played from paddles or keyboard. **\$27.50**

APPLE ARCADE



Shooting Gallery

Shooting Gallery - A real time simulation of a midway arcade. Features: row targets, pop targets, different skill levels, and bonus time. May be played using either game paddles or joy sticks. **\$22.50**



Star Avenger -High speed guerrilla warfare in space pitting you against your Apple. Featuring a new universe each game and varying skill levels. Universe consists of 16 hi-res screens with instantaneous crossover. **\$27.50**

Available Now!
MultiBoot™ Upgrade

Have you not wished that your Basic software would work in both DOS 3.2 and DOS 3.3? Tired of spending hours "Muffin"ing your old programmes? Now your problems are solved with MultiBoot™ Upgrade. Upgrade a whole disc in just seconds and use your disc on any of DOS 3.3, DOS 3.2 and the Language Card. **\$50.00**

All Western MicroData game software is written in assembly language for maximum speed. All programmes require 48K and DISK DRIVE and will work on standard Apple II, Apple II plus, and Pascal systems, with either DOS 3.2 or DOS 3.3.

Western MicroData Enterprises Ltd.

P.O. Box G33, Postal Station G
Calgary, Alberta
Canada T3A 2W1
1-403-247-1621

For U.S. and foreign orders, prices are in U.S. dollars. For Canadian orders, prices are in Canadian dollars. Send cheque or Postal Money Order only. Allow 3-4 weeks for cheque to clear if not certified and allow up to 4 weeks for delivery.

Dealer and Computer Club enquiries invited.

Apple is a registered trademark of Apple Computer Inc.
Disk II is a registered trademark of Apple Computer Inc.

Apple Memory Maps, Part 2

Part 1 of this series (presented last month) gave several examples of memory maps which showed where the Apple stores its various program components. This concluding article contains a listing and description of the program which produced the maps.

Peter A. Cook
1443 N. 24th Street
Mesa, Arizona 85203

The Program

In order to draw a map of a BASIC program, two programs must be stored in memory at the same time. You must have the BASIC program which is to be mapped, and the mapping program itself. To achieve this, I first thought that an Integer BASIC and an Applesoft version of the mapping program would have to be constructed and appended onto the program which was to be observed. Another alternative was to write a machine language program which would work for either version of BASIC, and wouldn't have to be appended. Although this was obviously the better choice, it seemed a formidable task to me because I had never written a machine language program before. It turned out to be much easier than expected, however, and if you've never tried it yourself, it's a lot like programming a programmable calculator.

Since I don't have an assembler, I used the Apple's mini-assembler to do the job. The program is printed in listings 1, 2, and 3. Monitor routines were used wherever possible to keep the program short.

Several storage locations in page zero of memory had to be used in order to facilitate the indirect mode of addressing. Locations were selected which do not interfere with the

monitor, DOS, Integer BASIC, or Applesoft. They are listed in figure 18.

The program is entered from any version of BASIC by a CALL 13000. An even number inside the range of program lines was chosen because it is easy to remember. From this location it jumps to \$3200, the actual start of the program.

The program was assembled at \$3200 to allow it to be used in as small as a 16K machine, which ends at \$3FFF, and to permit RAM Applesoft to remain intact, which ends at \$3000. When MEMORY MAP is loaded it may overwrite part of the BASIC program, but since the program pointers will not have changed, MEMORY MAP will operate correctly. Hi-Res graphics page one, which extends from \$2000-4000, will definitely be overwritten. The only drawback is that if the BASIC program is overwritten, it will have to be loaded again following the use of MEMORY MAP.

For machines larger than 16K it would be more advantageous to move the program to a higher location, such as above Hi-Res graphics page two. Then you could jump back and forth between the BASIC program and MEMORY MAP without having to reload them, even if you have a very

long BASIC program. The changes required to do this would be many. Most of the JSR and JMP addresses would have to be changed, as well as the string addresses and text page locations. Also, the CALL instruction would have to be placed in a different location, such as 25000.

Pointers

The Apple remembers where it stores the various components of a program by placing their starting addresses in "pointers." The pointers are locations in page zero of memory which are set by the monitor in response to certain BASIC commands and control keys. Figure 19 lists the pointers and other reference locations used by the MEMORY MAP program.

Loading Instructions

1. Enter the hex values from listings 1, 2, and 3 into the computer using this format:

```
*3200:20 84 FE 20 2F FB ...
```

Up to 255 characters, or 85 hex pairs, can be entered following the colon. Then press the return key and start with another colon to continue.

\$1A	Language:	0 Integer BASIC 1 RAM Applesoft 2 ROM Applesoft
\$1B	DOS:	0 Not loaded 1 Loaded
\$1C, 1D	String starting address	
\$1E, 1F	Temporary usage	
\$FA, FB	Address of pointer's low byte	
\$FC, FD	Address of pointer's high byte	
\$FE, FF	Constants for non-pointer addresses	

Figure 18: Page zero usage by MEMORY MAP.

2. Save the program on disk using BSAVE MEMORY MAP, A\$3200, L\$6E0, or you can save it on cassette using *3200.38DFW.

3. To use the program, first load the BASIC program you wish to see mapped. Run it through to the end using as many different branches of the program as possible, to place all the variables, arrays, and strings into storage. If the program doesn't end automatically, terminate it with a Control C.

4. Load MEMORY MAP from disk using BLOAD MEMORY MAP, or from cassette using 3200.38DFR. Note: MEMORY MAP may overwrite part of the BASIC program. Be sure you have saved a good copy of it first.

5. CALL 13000. The memory map will now appear on the screen. If you wish to print it, press Y in response to the question at the bottom. If MEMORY MAP is stored on disk, you can use BRUN MEMORY MAP instead of the separate BLOAD and CALL commands.

6. If you wish to run the BASIC program over again, you may need to reload it first, depending on whether or not MEMORY MAP has overwritten part of it.

Integer BASIC Pointers

LOMEM	74, 75	\$4A, 4B
HIMEM	76, 77	\$4C, 4D
Program pointer	202, 203	\$CA, CB
Free space pointer	204, 205	\$CC, CD

Applesoft Pointers

Program pointer	103, 104	\$67, 68
Variable pointer (LOMEM)	105, 106	\$69, 6A
Array pointer	107, 108	\$6B, 6C
Free space pointer	109, 110	\$6D, 6E
String pointer	111, 112	\$6F, 70
HIMEM	115, 116	\$73, 74
End of program pointer	175, 176	\$AF, B0

Other Data

Language prompt	51	\$33
DOS slot number	1528	\$5F8
DOS file buffers (48K)	43607	\$AA57

Figure 19: Pointers and other reference locations used by MEMORY MAP.

References

1. *Apple II Reference Manual*, Apple Computer Inc., 1979 (new version).
2. *Applesoft II Basic Programming Reference Manual*, Apple Computer Inc., 1978.
3. *DOS Version 3.2 Instructional and Reference Manual*, Apple Computer Inc., 1979.
4. *The Apple II Monitor Peeled*, William E. Dougherty, 1979.
5. "What's Where in the Apple," William F. Luebbert, *MICRO*, August 1979, p. 29.
6. "Disassembling the DOS 3.2," William Reynolds, *MICRO*, October 1979, p. 7.

Program Remarks

The following remarks explain what the different sections of the program are for, and how they work.

Listing 1: Disassembled MEMORY MAP program.

Clears the screen. Selects text mode, normal characters, and the full text window.

3200-	20 84 FE	JSR	\$FE84
3203-	20 2F FB	JSR	\$FB2F
3206-	20 93 FE	JSR	\$FE93
3209-	20 89 FE	JSR	\$FE89
320C-	20 58 FC	JSR	\$FC58
320F-	08	CLO	
3210-	A9 00	LDA	#\$00
3212-	85 FB	STA	\$FB
3214-	85 FD	STA	\$FD
3216-	85 1B	STA	\$1B
3218-	85 1A	STA	\$1A
321A-	A8	TAY	

Checks the prompt character to see which language is in use. If it is Applesoft, it checks location \$E000 to see whether it contains a JMP instruction. If it does, the Applesoft ROM is in use. If not, the Integer BASIC ROM is connected, so you are using RAM Applesoft. If you entered the program from the monitor or the mini-assembler, it will assume you want Integer BASIC.

321B-	A5 33	LDA	\$33
321D-	C9 00	CMP	#\$00
321F-	00 0B	BNE	\$322C
3221-	E6 1A	INC	\$1A
3223-	A0 00 E0	LDA	\$E000
3226-	C9 4C	CMP	#\$4C
3228-	00 02	BNE	\$322C
322A-	E6 1A	INC	\$1A

Checks for maximum memory size by starting at 48K and trying to store a 0 and a 1 in that location. If it can't, it keeps decreasing the address by 4K until it can.

322C-	A9 FF	LDA	#\$FF
322E-	85 FE	STA	\$FE
3230-	A9 BF	LDA	#\$BF
3232-	85 FF	STA	\$FF
3234-	B1 FE	LDA	(\$FE),Y
3236-	85 1E	STA	\$1E
3238-	A9 00	LDA	#\$00
323A-	91 FE	STA	(\$FE),Y
323C-	D1 FE	CMP	(\$FE),Y
323E-	00 08	BNE	\$3248
3240-	A9 01	LDA	#\$01
3242-	91 FE	STA	(\$FE),Y
3244-	D1 FE	CMP	(\$FE),Y

```

3246- F0 09 BEQ $3251
3248- A5 FF LDA $FF
324A- 38 SEC
324B- E9 10 SBC #$10
324D- 85 FF STA $FF
324F- D0 E3 BNE $3234
3251- A5 1E LDA $1E
3253- 91 FE STA ($FE),Y
3255- E6 FE INC $FE
3257- E6 FF INC $FF

```

Checks location \$5F8 to see if DOS has been loaded. This location contains the slot number of the last DOS boot in the form \$n0, so it is checked to see if it falls in the range \$10 to \$70.

```

3259- A0 F8 05 LDA $05F8
325C- C9 10 CMP #$10
325E- 30 09 BMI $3269
3260- A0 F8 05 LDA $05F8
3263- C9 71 CMP #$71
3265- 10 02 BPL $3269
3267- E6 1B INC $1B
3269- 4C 86 32 JMP $3286

```

This subroutine selects and prints strings from the list at the end of the program, using the starting address of the string as a pointer. The first byte contains the horizontal tab, the second byte contains the number of characters, and the remaining bytes hold the characters themselves in reverse order.

```

326C- 85 1C STA $1C
326E- A9 38 LDA #$38
3270- 85 10 STA $10
3272- A0 00 LDY #$00
3274- B1 1C LDA ($1C),Y
3276- 85 24 STA $24
3278- E6 1C INC $1C
327A- B1 1C LDA ($1C),Y
327C- A8 TAY
327D- B1 1C LDA ($1C),Y
327F- 20 ED FD JSR $FDED
3282- 88 DEY
3283- D0 F8 BNE $327D
3285- 60 RTS

```

Prints the title and the language in use.

```

3286- A9 00 LDA #$00
3288- 20 5B FB JSR $FB5B
328B- A9 00 LDA #$00
328D- 20 6C 32 JSR $326C
3290- A5 1A LDA $1A
3292- F0 08 BEQ $329C
3294- A9 0F LDA #$0F
3296- 20 6C 32 JSR $326C
3299- 4C B4 32 JMP $32B4
329C- A9 1A LDA #$1A
329E- 20 6C 32 JSR $326C
32A1- 4C B4 32 JMP $32B4

```

Prints the starting and ending addresses of the Hi-Res graphics pages as a constant reminder of their location. Will not determine if Hi-Res is actually used, however.

```

32A4- 20 6C 32 JSR $326C
32A7- E6 25 INC $25
32A9- 20 22 FC JSR $FC22
32AC- A5 1C LDA $1C
32AE- 18 CLC
32AF- 69 06 ADC #$06
32B1- 85 1C STA $1C
32B3- 60 RTS
32B4- A9 07 LDA #$07
32B6- 20 5B FB JSR $FB5B
32B9- A9 29 LDA #$29
32BB- 20 A4 32 JSR $32A4
32BE- A2 03 LDX #$03
32C0- 20 CB 32 JSR $32CB
32C3- CA DEX
32C4- D0 FA BNE $32C0
32C6- F0 11 BEQ $32D9
32C8- 4C 00 32 JMP $3200
32C8- E6 25 INC $25
32CD- 20 22 FC JSR $FC22
32D0- A5 1C LDA $1C
32D2- 20 A4 32 JSR $32A4
32D5- 20 A4 32 JSR $32A4
32D8- 60 RTS

```

Draws two vertical lines to outline the memory map.

```

32D9- A2 02 LDX #$02
32DB- A9 09 LDA #$09
32DD- 85 24 STA $24
32DF- A0 14 LDY #$14
32E1- A9 02 LDA #$02
32E3- 85 25 STA $25
32E5- 20 22 FC JSR $FC22
32E8- A9 A1 LDA #$A1
32EA- 20 ED FD JSR $FDED
32ED- E6 25 INC $25
32EF- C6 24 DEC $24
32F1- 88 DEY
32F2- D0 F1 BNE $32E5
32F4- A9 16 LDA #$16
32F6- 85 24 STA $24
32F8- CA DEX
32F9- D0 E4 BNE $32DF
32FB- 4C AC 33 JMP $33AC

```

Subroutine for drawing horizontal lines on the map.

```

32FE- A9 0C LDA #$0C
3300- 85 1E STA $1E
3302- A9 0A LDA #$0A
3304- 85 24 STA $24
3306- A9 A0 LDA #$A0
3308- 20 ED FD JSR $FDED
330B- C6 1E DEC $1E
330D- D0 F7 BNE $3306
330F- 60 RTS

```

Subroutine for converting hexadecimal numbers to decimal numbers, and printing them. Divides by 10,000, 1000, 100, and 10 to obtain each digit, using the divide routine at \$3779. Leading zeros are not printed. Numbers are right-justified.

```

3310- A9 1B      LDA  #$1B
3312- 85 24      STA  $24
3314- A9 00      LDA  #$00
3316- 85 1E      STA  $1E
3318- 85 53      STA  $53
331A- 85 52      STA  $52
331C- A8         TAY
331D- B1 FC      LDA  ($FC),Y
331F- 85 51      STA  $51
3321- B1 FA      LDA  ($FA),Y
3323- 85 50      STA  $50
3325- A9 27      LDA  #$27
3327- 85 55      STA  $55
3329- A9 10      LDA  #$10
332B- 85 54      STA  $54
332D- 20 6E 33   JSR  $336E
3330- 20 5F 33   JSR  $335F
3333- A9 03      LDA  #$03
3335- 85 55      STA  $55
3337- A9 E8      LDA  #$E8
3339- 85 54      STA  $54
333B- 20 6E 33   JSR  $336E
333E- 20 5F 33   JSR  $335F
3341- A9 00      LDA  #$00
3343- 85 55      STA  $55
3345- A9 64      LDA  #$64
3347- 85 54      STA  $54
3349- 20 6E 33   JSR  $336E
334C- 20 5F 33   JSR  $335F
334F- A9 0A      LDA  #$0A
3351- 85 54      STA  $54
3353- 20 6E 33   JSR  $336E
3356- A5 52      LDA  $52
3358- 18         CLC
3359- 69 B0      ADC  #$B0
335B- 20 ED FD   JSR  $FDEE
335E- 60         RTS
335F- A5 53      LDA  $53
3361- 85 51      STA  $51
3363- A5 52      LDA  $52
3365- 85 50      STA  $50
3367- A9 00      LDA  #$00
3369- 85 53      STA  $53
336B- 85 52      STA  $52
336D- 60         RTS
336E- 20 79 37   JSR  $3779
3371- A5 50      LDA  $50
3373- 18         CLC
3374- 65 1E      ADC  $1E
3376- 85 1E      STA  $1E
3378- D0 04      BNE  $337E
337A- E6 24      INC  $24

```

```

337C- D0 08      BNE  $3386
337E- A5 50      LDA  $50
3380- 18         CLC
3381- 69 B0      ADC  #$B0
3383- 20 ED FD   JSR  $FDEE
3386- 60         RTS

```

Subroutine for printing the hex numbers used in \$3310.

```

3387- A9 22      LDA  #$22
3389- 85 24      STA  $24
338B- A9 A4      LDA  #$A4
338D- 20 ED FD   JSR  $FDEE
3390- A0 00      LDY  #$00
3392- B1 FC      LDA  ($FC),Y
3394- 20 DA FD   JSR  $FDDA
3397- B1 FA      LDA  ($FA),Y
3399- 20 DA FD   JSR  $FDDA
339C- 60         RTS

```

Calls the three preceding subroutines.

```

339D- 85 25      STA  $25
339F- 20 22 FC   JSR  $FC22
33A2- 20 FE 32   JSR  $32FE
33A5- 20 10 33   JSR  $3310
33A8- 20 87 33   JSR  $3387
33AB- 60         RTS

```

Prints the top horizontal line, and its decimal and hex addresses. Each horizontal line on the map will represent the starting address of the block above it. Thus, the top line for a 48K machine will be 49152, or \$C000. This is actually the first address of the ROM area. HIMEM is set to this value when the machine is first turned on.

```

33AC- A9 FE      LDA  #$FE
33AE- 85 FA      STA  $FH
33B0- A9 FF      LDA  #$FF
33B2- 85 FC      STA  $FC
33B4- A9 02      LDA  #$02
33B6- 20 9D 33   JSR  $339D

```

Checks location \$1B for a zero or a one to see if DOS is loaded.

```

33B9- A5 1B      LDA  $1B
33BB- D0 03      BNE  $33C0
33BD- 4C 56 34   JMP  $3456

```

If it is, prints "DOS, FILES" in the top block.

```

33C0- E6 25      INC  $25
33C2- 20 22 FC   JSR  $FC22
33C5- A9 5A      LDA  #$5A
33C7- 20 6C 32   JSR  $326C
33CA- E6 25      INC  $25
33CC- 20 22 FC   JSR  $FC22
33CF- A9 60      LDA  #$60
33D1- 20 6C 32   JSR  $326C

```

Checks location \$AA57 in a 48K machine for the number of DOS file buffers reserved. Three buffers are reserved when DOS is loaded, but the number can vary from 1 to 16 if changed by a MAXFILES command. To find the location for your memory size, subtract \$15A9 from the top of memory. DOS 3.1 uses a different location for this value, but I don't know what it is.

```

33D4-  A5 FE      LDA  $FE
33D6-  38         SEC
33D7-  E9 A9      SBC  #$A9
33D9-  85 1E      STA  $1E
33DB-  A5 FF      LDA  $FF
33DD-  E9 15      SBC  #$15
33DF-  85 1F      STA  $1F

```

Converts the hex number of buffers to a decimal number, and prints it.

```

33E1-  A0 00      LDY  #$00
33E3-  B1 1E      LDA  (<$1E),Y
33E5-  85 50      STA  $50
33E7-  98         TYA
33E8-  85 51      STA  $51
33EA-  85 52      STA  $52
33EC-  85 53      STA  $53
33EE-  85 55      STA  $55
33F0-  A9 0A      LDA  #$0A
33F2-  85 54      STA  $54
33F4-  20 79 37    JSR  $3779
33F7-  A9 04      LDA  #$04
33F9-  20 5B FB    JSR  $FB5B
33FC-  A9 12      LDA  #$12
33FE-  85 24      STA  $24
3400-  A5 50      LDA  $50
3402-  F0 06      BEQ  $340A
3404-  18         CLC
3405-  69 B0      ADC  #$B0
3407-  20 ED FD    JSR  $FD0D
340A-  A5 52      LDA  $52
340C-  18         CLC
340D-  69 B0      ADC  #$B0
340F-  20 ED FD    JSR  $FD0D
3412-  A9 A9      LDA  #$A9
3414-  20 ED FD    JSR  $FD0D

```

Multiplies the number of buffers by 595 to find their total length in bytes, using the multiply routine at \$375B.

```

3417-  A0 00      LDY  #$00
3419-  B1 1E      LDA  (<$1E),Y
341B-  85 50      STA  $50
341D-  98         TYA
341E-  85 51      STA  $51
3420-  85 52      STA  $52
3422-  85 53      STA  $53

```

```

3424-  A9 53      LDA  #$53
3426-  85 54      STA  $54
3428-  A9 02      LDA  #$02
342A-  85 55      STA  $55
342C-  20 5B 37    JSR  $375B

```

Subtracts the length of DOS, \$2307, from the top of memory to find the top of the buffers.

```

342F-  A5 FE      LDA  $FE
3431-  38         SEC
3432-  E9 07      SBC  #$07
3434-  85 FE      STA  $FE
3436-  A5 FF      LDA  $FF
3438-  E9 23      SBC  #$23
343A-  85 FF      STA  $FF

```

Subtracts the length of the buffers to find their starting address. Draws a horizontal line and prints the address. This is where HIMEM is set after a DOS boot. The table in figure 20 shows the values of HIMEM for different values of MAXFILES in a 48K machine.

```

343C-  A5 FE      LDA  $FE
343E-  38         SEC
343F-  E5 50      SBC  $50
3441-  85 FE      STA  $FE
3443-  A5 FF      LDA  $FF
3445-  E5 51      SBC  $51
3447-  85 FF      STA  $FF
3449-  A9 FE      LDA  #$FE
344B-  85 FA      STA  $FA
344D-  A9 FF      LDA  #$FF
344F-  85 FC      STA  $FC
3451-  A9 05      LDA  #$05
3453-  20 9D 33    JSR  $339D

```

FILES	HIMEM (DEC)	HIMEM (HEX)
1	39590	9AA6
2	38995	9853
3	38400	9600
4	37805	93AD
5	37210	915A
6	36615	8F07
7	36020	8CB4
8	35425	8A61
9	34830	880E
10	34235	85BB
11	33640	8368
12	33045	8115
13	32450	7EC2
14	31855	7C6F
15	31260	7A1C
16	30665	77C9

Figure 20: Values of HIMEM set by different MAXFILES.

Checks location \$1A to see which language is in use. If Integer BASIC, it branches to \$368A.

```
3456- A5 1A      LDA    $1A
3458- C9 00      CMP    #$00
345A- D0 03      BNE    $345F
345C- 4C 8A 36   JMP    $368A
```

Checks setting of HIMEM by looking at pointer address \$73,74. If same as bottom of DOS buffers, prints "HM". If a lower value has been set, draws another horizontal line and prints the new address.

```
345F- B1 FA      LDA    ($FA),Y
3461- C5 73      CMP    $73
3463- D0 0C      BNE    $3471
3465- B1 FC      LDA    ($FC),Y
3467- C5 74      CMP    $74
3469- D0 06      BNE    $3471
346B- 20 84 34   JSR    $3484
346E- 4C 93 34   JMP    $3493
3471- A9 73      LDA    #$73
3473- 85 FA      STA    $FA
3475- A9 74      LDA    #$74
3477- 85 FC      STA    $FC
3479- 20 7F 34   JSR    $347F
347C- 4C 93 34   JMP    $3493
```

Subroutine for printing "HM"

```
347F- E6 25      INC    $25
3481- 20 9F 33   JSR    $339F
3484- A9 18      LDA    #$18
3486- 85 24      STA    $24
3488- A9 C8      LDA    #$C8
348A- 20 ED FD   JSR    $FDED
348D- A9 CD      LDA    #$CD
348F- 20 ED FD   JSR    $FDED
3492- 60                RTS
```

Checks string pointer \$6F,70 to see if same as HIMEM.

```
3493- B1 FA      LDA    ($FA),Y
3495- C5 6F      CMP    $6F
3497- D0 06      BNE    $349F
3499- B1 FC      LDA    ($FC),Y
349B- C5 70      CMP    $70
349D- F0 17      BEQ    $3486
```

If not, prints "STRINGS". Draws horizontal line and prints address.

```
349F- E6 25      INC    $25
34A1- 20 22 FC   JSR    $FC22
34A4- A9 69      LDA    #$69
34A6- 20 6C 32   JSR    $326C
34A9- A9 6F      LDA    #$6F
34AB- 85 FA      STA    $FA
34AD- A9 70      LDA    #$70
34AF- 85 FC      STA    $FC
34B1- E6 25      INC    $25
34B3- 20 9F 33   JSR    $339F
```

Draws a horizontal line at the bottom of the map for address 2048. This is the bottom of usable memory for BASIC programs.

```
34B6- 20 BC 34   JSR    $34BC
34B9- 4C D2 34   JMP    $34D2
34BC- A9 08      LDA    #$08
34BE- 85 FF      STA    $FF
34C0- A9 00      LDA    #$00
34C2- 85 FE      STA    $FE
34C4- A9 FE      LDA    #$FE
34C6- 85 FA      STA    $FA
34C8- A9 FF      LDA    #$FF
34CA- 85 FC      STA    $FC
34CC- A9 15      LDA    #$15
34CE- 20 9D 33   JSR    $339D
34D1- 60                RTS
```

Checks location \$1A for a 1, to see if the language is RAM Applesoft. If it is, it prints "APPLESOFT."

```
34D2- A5 1A      LDA    $1A
34D4- C9 01      CMP    #$01
34D6- D0 0A      BNE    $34E2
34D8- C6 25      DEC    $25
34DA- 20 22 FC   JSR    $FC22
34DD- A9 72      LDA    #$72
34DF- 20 6C 32   JSR    $326C
```

Checks the program pointer, \$67,68. If RAM Applesoft is loaded, the program will start at 12289. If ROM Applesoft is used, the program will start at 2049. Draws a horizontal line and prints the address.

```
34E2- A9 67      LDA    #$67
34E4- 85 FA      STA    $FA
34E6- A9 68      LDA    #$68
34E8- 85 FC      STA    $FC
34EA- C6 25      DEC    $25
34EC- 20 9F 33   JSR    $339F
34EF- 20 F5 34   JSR    $34F5
34F2- 4C 14 35   JMP    $3514
```

Subroutine which checks the setting of LOMEM by looking at pointer \$69,6A. Prints "LM".

```
34F5- B1 FA      LDA    ($FA),Y
34F7- C5 69      CMP    $69
34F9- D0 09      BNE    $3504
34FB- B1 FC      LDA    ($FC),Y
34FD- C5 6A      CMP    $6A
34FF- D0 03      BNE    $3504
3501- 20 05 35   JSR    $3505
3504- 60                RTS
3505- A9 18      LDA    #$18
3507- 85 24      STA    $24
3509- A9 CC      LDA    #$CC
350B- 20 ED FD   JSR    $FDED
350E- A9 CD      LDA    #$CD
3510- 20 ED FD   JSR    $FDED
3513- 60                RTS
```




CONTINENTAL SOFTWARE THE APPLE SOURCE.

For Apple owners only. Thoroughly tested, well documented programs for business and pleasure. All written by professionals. Each checked out carefully by experts in its field.

**HYPERSPACE WARS
2 GAMES FOR THE PRICE OF 1 \$29.95**

48K Trek. Stardate 3421. The Terraunion is being attacked. You command United Starship Excalibur. Your mission: destroy the deadly Klepton invasion force. Four levels, Novice to Master.

3-D Space Battle. Use your on-board scanners to search for alien ships in hires three-dimensional space. Destroy as many aliens as you can before you run out of fuel or your ship is destroyed. Hi-res graphics. Req. 48K, Applesoft in Rom+1 disk drive. Dos. 3.2 or 3.3.

L.A. LAND MONOPOLY \$29.95

Bankrupt your opponents while becoming the richest player in the game. Buy, sell, rent and trade to accumulate the most cash and property. Two to six may play. Computer is banker. Create your own special version using streets in your own town.

Hi-res graphics. Req. 48K, Applesoft in Rom+1 disc drive. Dos. 3.2 or 3.3.

HOME MONEY MINDER \$34.95

Complete home financial system combines an excellent Home Checkbook Program with Budgeting. Transactions by month by budget category. Bank reconciliation. Budget for year. Total expenses compared monthly and year-to-date. Plus much more.

Req. 48K, Applesoft in Rom, 1 disk drive+printer. Avail. in Dos. 3.3.

THE MAILROOM \$34.95

Stores up to 750 names per disk. Prints master lists and labels 1, 2 or 3 across. Sorts in 5 seconds. Sort on any of 12 items, search any sorted item in 10-20 seconds maximum. Easy editing, customized inputs.

Req. 48K, Applesoft in Rom, 1 disk drive+printer (132 column capability needed to print Master List.) in Dos. 3.3.

THE COMPUTER PROGRAMMED ACCOUNTANT FOUR MODULES

Buy all four now—or add as you expand \$175 each (\$250 after 6/1/81)

The first programs for your Apple that your accountant will like as much as you do. Nobody makes it better—or easier to use—than Continental Software. Simple step-by-step instructions. Excellent error checking. Modules can be used individually, or integrated into a complete Accounting System. Manuals only: just \$15 each.

CPA1 GENERAL LEDGER.

True double entry bookkeeping with complete, accurate audit trails showing the source of each entry in the general ledger. Concise, meaningful reports generated include Balance Sheet, Profit & Loss Summary, Trial Balance and Complete Journal Activity Report. Reports show monthly, year-to-date and last year monthly+YTD for comparison. Custom charting feature includes hi-res plotting of one or more accounts.

CPA2 ACCOUNTS RECEIVABLE

Prints invoices on available custom forms or on plain paper. Back orders and extensions computed. Issues statements for all customers, one or more customers, or only those with current, 30-, 60-, 90- or 150-day balances. Maintain up to 300 customers. Customized journals. Allows simulation of manual special journal entries. Posts to General Ledger. Prints aging report to 150 days. Also prints customer lists and labels.

CPA3 ACCOUNTS PAYABLE

Prints checks to vendors and non-vendors on available pre-printed checks or plain paper. Each check stub shows invoice(s) paid, discounts taken, net paid, Prints Purchases and Cash

Disbursement Journals. Customized journals. Allows simulation of manual special journal entries. Prints Aging Report to 150 days, vendor list and labels and even a Cash Requirements Report. Posts to General Ledger.

CPA4 PAYROLL

Maintains personnel records for as many as 100 employees. Quarter-to-date and year-to-date earnings and deduction records. Employees are departmentalized and designated hourly or salaried. Prints complete Payroll Checks, 941 information, W-2s, State of California DE-3 information. Prints Payroll Journal and posts to General Ledger.

These are just some of the features of each CPA module. All require 48K, Applesoft in Rom, Dos. 3.3, 2 disk drives+printer.

At your local dealer or fill out and mail today. Phone for immediate delivery.

OK, I'LL BYTE.

Send me these revolutionary programs:

- Hyperspace Wars . . . \$ _____
- L.A. Land Monopoly. _____
- Home Money Minder _____
- The Mailroom _____
- CPA1 General Ledger _____
- CPA2 Accts. Rec. _____
- CPA3 Accts. Pay. _____
- CPA4 Payroll _____
- No. C.O.D.s Subtotal _____
- Cal. res. add 6% _____
- TOTAL _____

Name _____
Address _____
City _____ State _____ Zip _____
Card No. _____ Exp. _____

M15/81

CONTINENTAL SOFTWARE

12101 Jefferson Blvd.,
Culver City, CA 90230

(213) 371-5612



Checks the end-of-program pointer \$AF,B0. Prints "PROGRAM". Draws a horizontal line above it and prints the address. If no Applesoft program is loaded, the end-of-program will be one or two bytes higher than the starting pointer.

```

3514- B1 FA LDA ($FA),Y
3516- C5 AF CMP $AF
3518- D0 06 BNE $3520
351A- B1 FC LDA ($FC),Y
351C- C5 B0 CMP $B0
351E- F0 1A BEQ $353A
3520- C6 25 DEC $25
3522- 20 22 FC JSR $FC22
3525- A9 7D LDA #$7D
3527- 20 6C 32 JSR $326C
352A- A9 AF LDA #$AF
352C- 85 FA STA $FA
352E- A9 B0 LDA #$B0
3530- 85 FC STA $FC
3532- C6 25 DEC $25
3534- 20 9F 33 JSR $339F
3537- 20 F5 34 JSR $34F5

```

Checks \$69,6A for the setting of LOMEM. It should have been set automatically to the same position as the end-of-program pointer. If different, draws another line and labels it "LM" with the proper address. This is the starting location for variables.

```

353A- B1 FA LDA ($FA),Y
353C- C5 69 CMP $69
353E- D0 06 BNE $3546
3540- B1 FC LDA ($FC),Y
3542- C5 6A CMP $6A
3544- F0 10 BEQ $3556
3546- A9 69 LDA #$69
3548- 85 FA STA $FA
354A- A9 6A LDA #$6A
354C- 85 FC STA $FC
354E- C6 25 DEC $25
3550- 20 9F 33 JSR $339F
3553- 20 F5 34 JSR $34F5

```

Checks the array pointer \$6B,6C to see if different from LOMEM. If it is, prints "VARIABLES" and draws a line above it for the start of array space.

```

3556- B1 FA LDA ($FA),Y
3558- C5 6B CMP $6B
355A- D0 06 BNE $3562
355C- B1 FC LDA ($FC),Y
355E- C5 6C CMP $6C
3560- F0 17 BEQ $3579
3562- C6 25 DEC $25
3564- 20 22 FC JSR $FC22
3567- A9 86 LDA #$86
3569- 20 6C 32 JSR $326C
356C- A9 6B LDA #$6B
356E- 85 FA STA $FA

```

```

3570- A9 6C LDA #$6C
3572- 85 FC STA $FC
3574- C6 25 DEC $25
3576- 20 9F 33 JSR $339F

```

Checks the free space pointer \$6D,6E to see if it is the same as the start of array space. If not, it prints "ARRAYS" and draws a line above it.

```

3579- B1 FA LDA ($FA),Y
357B- C5 6D CMP $6D
357D- D0 06 BNE $3585
357F- B1 FC LDA ($FC),Y
3581- C5 6E CMP $6E
3583- F0 17 BEQ $359C
3585- C6 25 DEC $25
3587- 20 22 FC JSR $FC22
358A- A9 91 LDA #$91
358C- 20 6C 32 JSR $326C
358F- A9 6D LDA #$6D
3591- 85 FA STA $FA
3593- A9 6E LDA #$6E
3595- 85 FC STA $FC
3597- C6 25 DEC $25
3599- 20 9F 33 JSR $339F

```

Computes the amount of free space by subtracting the free space address from the string address. Prints the amount in decimal and hex. This completes the Applesoft map.

```

359C- 20 A2 35 JSR $35A2
359F- 4C A0 35 JMP $35A0
35A2- A9 0A LDA #$0A
35A4- 20 5B FB JSR $FB5B
35A7- A9 99 LDA #$99
35A9- 20 6C 32 JSR $326C
35AC- 60 RTS
35AD- A5 6F LDA $6F
35AF- 38 SEC
35B0- E5 6D SBC $6D
35B2- 85 FE STA $FE
35B4- A5 70 LDA $70
35B6- E5 6E SBC $6E
35B8- 85 FF STA $FF
35BA- A9 FE LDA #$FE
35BC- 85 FA STA $FA
35BE- A9 FF LDA $FF
35C0- 85 FC STA $FC
35C2- 20 10 33 JSR $3310
35C5- 20 87 33 JSR $3387

```

The next part of the program allows you to print the map on a printer, as was done for the illustrations in this article. Places the input line "PRINT (Y)?" at the bottom of the screen.

```

35C8- A9 17 LDA #$17
35CA- 20 5B FB JSR $FB5B
35CD- A9 A5 LDA #$A5
35CF- 20 6C 32 JSR $326C

```

If the response is not a "Y", erases the question, replaces it with the prompt character of the original BASIC program, and ends the MEMORY MAP program.

```

3502- 20 1B FD   JSR   $FD1B
3505- C9 09     CMP   #$09
3507- F0 1F     BEQ   $35F8
3509- A9 00     LDA   #$00
350B- 85 24     STA   $24
350D- 20 9C FC   JSR   $FC9C
350E- A9 16     LDA   #$16
3502- 20 5B FB   JSR   $FB5B
3505- A5 1A     LDA   $1A
3507- C9 01     CMP   #$01
3509- D0 03     BNE   $35EE
350B- 4C 3C 0C   JMP   $0C3C
350E- A5 1B     LDA   $1B
3510- F0 03     BEQ   $35F5
3512- 4C D0 03  JMP   $03D0
3515- 4C 03 E0   JMP   $E003

```

If the response is a "Y", the program continues. It was designed for use with a Trendcom 200 printer. One of the features of this printer's interface card is that it prints a line of characters on the screen before it prints them on the paper. In order to print only the memory map display from text page one, we have to move it first to another location before it becomes cluttered with extra characters from the printing process. The monitor MOVE routine is used here to move \$400.800 to \$3900.3D00. The MOVE routine transfers bytes from the addresses contained in \$3C,3D through \$3E,3F to the new address in \$42,43.

```

35F8- A9 00     LDA   #$00
35FA- 85 3C     STA   $3C
35FC- 85 3E     STA   $3E
35FE- 85 42     STA   $42
3600- A8       TAY
3601- A9 04     LDA   #$04
3603- 85 3D     STA   $3D
3605- A9 08     LDA   #$08
3607- 85 3F     STA   $3F
3609- A9 39     LDA   #$39
360B- 85 43     STA   $43
360D- 20 2C FE   JSR   $FE2C
3610- 4C 3E 36  JMP   $3E36

```

Subroutine for printing a horizontal border line on the finished map.

```

3613- A9 30     LDA   #$30
3615- 85 1E     STA   $1E
3617- A9 0A     LDA   #$0A
3619- 85 24     STA   $24
361B- A9 AD     LDA   #$AD
361D- 20 ED FD   JSR   $FDED
3620- C6 1E     DEC   $1E
3622- D0 F7     BNE   $361B
3624- 20 8E FD   JSR   $FD8E
3627- 60       RTS

```

Subroutine for printing a blank line within vertical border lines.

```

3628- A9 09     LDA   #$09
362A- 85 24     STA   $24
362C- A9 A1     LDA   #$A1
362E- 20 ED FD   JSR   $FDED
3631- A9 3A     LDA   #$3A
3633- 85 24     STA   $24
3635- A9 A1     LDA   #$A1
3637- 20 ED FD   JSR   $FDED
363A- 20 8E FD   JSR   $FD8E
363D- 60       RTS

```

Selects the printer slot number in the form \$Cn00. You will have to change location \$3643 to a different number if your printer is not in slot #2.

```

363E- A9 00     LDA   #$00
3640- 85 36     STA   $36
3642- A9 C2     LDA   #$C2
3644- 85 37     STA   $37

```

Prints a border around the outside of the map. Prints the moved text page line-by-line using the starting locations for each line stored at \$38B2.

```

3646- 20 13 36  JSR   $3613
3649- 20 28 36  JSR   $3628
364C- A2 2E     LDX   #$2E
364E- A9 09     LDA   #$09
3650- 85 24     STA   $24
3652- A9 A1     LDA   #$A1
3654- 20 ED FD   JSR   $FDED
3657- A9 0E     LDA   #$0E
3659- 85 24     STA   $24
365B- B0 B0 38    LDA   $38B0,X
365E- 85 1E     STA   $1E
3660- B0 B1 38    LDA   $38B1,X
3663- 85 1F     STA   $1F
3665- A0 00     LDY   #$00
3667- B1 1E     LDA   ($1E),Y
3669- 20 ED FD   JSR   $FDED
366C- C8       INY
366D- C0 27     CPY   #$27
366F- D0 F6     BNE   $3667
3671- A9 3A     LDA   #$3A
3673- 85 24     STA   $24
3675- A9 A1     LDA   #$A1
3677- 20 ED FD   JSR   $FDED
367A- 20 8E FD   JSR   $FD8E
367D- CA       DEX
367E- CA       DEX
367F- D0 C0     BNE   $364E
3681- 20 13 36  JSR   $3613

```

Restores normal screen output at the end of printing, and returns to BASIC to end the program.

```

3684- 20 93 FE   JSR   $FE93
3687- 4C E5 35  JMP   $35E5

```

MICRO

Classified

Programmer Fatigue?

SYM—BUG/MONEX adds 15 commands to SYM's repertoire including an interactive trace/debug. Cassette @ \$0200 or \$3800: \$19.95. EPROM (2716-5v) @ \$F000-\$F7FF: \$39.95. Commented source listing: \$9.95. RAE-1(1/2) FORMAT CASSETTE: \$35 (requires 8K). Custom assembly add \$2.00. Foreign add \$2.00. SASE for more information.

Jeff Holtzman
6820 Delmar-203
St. Louis, Missouri 63130

PET Machine Language Guide

Comprehensive manual to aid machine language programmer. More than 30 routines are fully detailed so that the reader can put them to immediate use. OLD or NEW ROMS. \$6.95 + .75 postage. VISA & Mastercharge accepted.

Abacus Software
P.O. Box 7211
Grand Rapids, Michigan 49510

AIM-65 Newsletter * * Target

Target provides hardware and software information useful for AIM-65 and 6502 users. The 1979 and 1980 back issues are available for \$12.00 while a continuing subscription costs \$6.00. Just write to:

Target
Donald Clem
Route 2
Spenserville, Ohio 45887

OSI SUPERB/C1P - New MonitorROM

You haven't seen a better utility ROM! Exchange MonitorROM and get: Screen-editor (insert, delete), cursor control, f. 24/32/64 chr/line; cassette-sys. w. file name handler (3-4 times faster), handles BASIC, Hexcode and variable arrays, and more. Further applications and program information \$1.00.

Gerwin Bleich
Boschstr. 1,3004
Isernhagen 1, West Germany

Quality Educational Courseware

Elementary educational courseware for the Apple II. All programs feature large lower-case letters, record keeping, and documentation. All require Applesoft 48K, disk. CLOCK: \$29.95, PRESCRIPTIVE MATH DRILL: \$79.95. Write for catalog.

Hartley Software
3268 Coach Lane #2A Dept. M
Kentwood, MI 49508

Spanish Hangman

2,000 SPANISH words and sentences taught in a fun way on the Apple. Send for your school's free 30-day evaluation diskette, from:

George Earl
1302 South General McMullen
San Antonio, Texas 78237

AIM/KIM/SYM

NBS Computing gives you time! A battery backed-up clock-calendar board that runs on the application bus. The clock will run for months without power and can generate interrupts on SYM systems. \$69.95 assembled, \$34.95 bare board. Both include drivers.

NBS Computing
1674 E. M-36
Pinckney, Michigan 48169

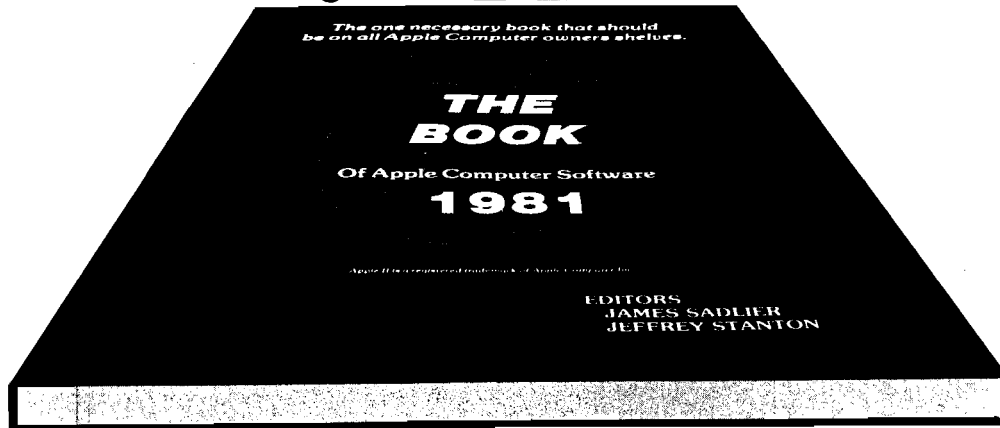
ASTEROIDS for OSI

Enjoy the arcade game in the comfort of your home. Exciting and habit forming. All in machine code. Specify system. Sorry NO 8" disks. \$10.95 tape or disk.

W.C. Software
1319 N. 16th
Grand Junction, CO 81501

(Continued on page 80)

Don't buy Apple Software



until you read this book.

First check The Book—the one complete critical analysis of most Apple Software available. Games, Educational, Business, Utility programs and more. Each comprehensively rated on 11 separate points. Each reviewed by an expert in its field. Just \$19.95.

Now you can compare and get more for your software dollar. Does the program you need exist? How good is it? Which software vendors offer the best support? Find out all this and much more.

MasterCard & Visa accepted. Fill out and mail today or call for shipment.

Calif. residents add 6%

16720 HAWTHORNE BLVD., LAWNSDALE, CA 90260. (213) 371-4012.

NAME _____
ADDRESS _____
CITY _____ STATE _____ ZIP _____
CARD NUMBER _____ EXP. _____

TheBookCompany

M15/'81

The following routines check the Integer BASIC pointers, which are different from the ones used for Applesoft. This one checks the setting of HIMEM, \$4C,4D.

```

368A- B1 FA LDA ($FA),Y
368C- C5 4C CMP $4C
368E- D0 0C BNE $369C
3690- B1 FC LDA ($FC),Y
3692- C5 4D CMP $4D
3694- D0 06 BNE $369C
3696- 20 84 34 JSR $3484
3699- 4C A7 36 JMP $36A7
369C- A9 4C LDA #$4C
369E- 85 FA STA $FA
36A0- A9 4D LDA #$4D
36A2- 85 FC STA $FC
36A4- 20 7F 34 JSR $347F

```

Checks the program pointer \$CA,CB.

```

36A7- B1 FA LDA ($FA),Y
36A9- C5 CA CMP $CA
36AB- D0 06 BNE $36B3
36AD- B1 FC LDA ($FC),Y
36AF- C5 CB CMP $CB
36B1- F0 17 BEQ $36CA
36B3- E6 25 INC $25
36B5- 20 22 FC JSR $FC22
36B8- A9 7D LDA #$7D
36BA- 20 6C 32 JSR $326C
36BD- A9 CA LDA #$CA
36BF- 85 FA STA $FA
36C1- A9 CB LDA #$CB
36C3- 85 FC STA $FC
36C5- E6 25 INC $25
36C7- 20 9F 33 JSR $339F

```

Draws the bottom line at 2048.

```

36CA- 20 BC 34 JSR $34BC
36CD- 20 D3 36 JSR $36D3
36D0- 4C E3 36 JMP $36E3

```

Checks the setting of LOMEM, \$4A,4B. This is the beginning of storage for variables, arrays, and strings, which are all stored in the same area in Integer BASIC.

```

36D3- B1 FA LDA ($FA),Y
36D5- C5 4A CMP $4A
36D7- D0 09 BNE $36E2
36D9- B1 FC LDA ($FC),Y
36DB- C5 4B CMP $4B
36DD- D0 03 BNE $36E2
36DF- 20 05 35 JSR $3505
36E2- 60 RTS

```

```

36E3- B1 FA LDA ($FA),Y
36E5- C5 4A CMP $4A
36E7- D0 06 BNE $36EF
36E9- B1 FC LDA ($FC),Y
36EB- C5 4B CMP $4B
36ED- F0 10 BEQ $36FF
36EF- A9 4A LDA #$4A
36F1- 85 FA STA $FA
36F3- A9 4B LDA #$4B
36F5- 85 FC STA $FC
36F7- C6 25 DEC $25
36F9- 20 9F 33 JSR $339F
36FC- 20 D3 36 JSR $36D3

```

Checks the free space pointer \$CC,CD to see where the variables end.

```

36FF- B1 FA LDA ($FA),Y
3701- C5 CC CMP $CC
3703- D0 06 BNE $370B
3705- B1 FC LDA ($FC),Y
3707- C5 CD CMP $CD
3709- F0 2F BEQ $373A
370B- C6 25 DEC $25
370D- 20 22 FC JSR $FC22
3710- A9 69 LDA #$69
3712- 20 6C 32 JSR $326C
3715- C6 25 DEC $25
3717- C6 25 DEC $25
3719- 20 22 FC JSR $FC22
371C- A9 91 LDA #$91
371E- 20 6C 32 JSR $326C
3721- C6 25 DEC $25
3723- C6 25 DEC $25
3725- 20 22 FC JSR $FC22
3728- A9 86 LDA #$86
372A- 20 6C 32 JSR $326C
372D- A9 CC LDA #$CC
372F- 85 FA STA $FA
3731- A9 CD LDA #$CD
3733- 85 FC STA $FC
3735- C6 25 DEC $25
3737- 20 9F 33 JSR $339F

```

Computes the amount of free space by subtracting the free space address from the program address. This completes the memory map for Integer BASIC. Jumps back to \$35C8 for the printer routine.

```

373A- 20 A2 35 JSR $35A2
373D- A5 CA LDA $CA
373F- 38 SEC
3740- E5 CC SBC $CC
3742- 85 FE STA $FE
3744- A5 CB LDA $CB
3746- E5 CD SBC $CD

```

```

3748- 85 FF      STA  $FF
374A- A9 FE      LDA  $$FE
374C- 85 FA      STA  $FA
374E- A9 FF      LDA  $$FF
3750- 85 FC      STA  $FC
3752- 20 10 33   JSR  $3310
3755- 20 87 33   JSR  $3387
3758- 4C C8 35   JMP  $35C8

```

Subroutine for multiplying integers. This is the MUL routine from the old monitor ROM, in case you have the autostart ROM installed. Multiplies number in \$50,51 by number in \$54,55 leaving 16-bit result in \$50,51,52,53.

```

3758- A0 10      LDY  $$10
375D- A5 50      LDA  $50
375F- 4A        LSR
3760- 90 0C      BCC  $376E
3762- 18        CLC
3763- A2 FE      LDX  $$FE
3765- 85 54      LDA  $54,X
3767- 75 56      ADC  $56,X
3769- 95 54      STA  $54,X
376B- E8        INX
376C- 00 F7      BNE  $3765
376E- A2 03      LDX  $$03
3770- 76 50      ROR  $50,X
3772- CA        DEX
3773- 10 FB      BPL  $3770
3775- 88        DEY
3776- D0 E5      BNE  $375D
3778- 60        RTS

```

Subroutine for dividing integers. This is the DIV routine from the old monitor ROM. Divides 16-bit number in \$50,51,52,53 by number in \$54,55, leaving quotient in \$50,51 and remainder in \$52,53.

```

3779- A0 10      LDY  $$10
377B- 06 50      ASL  $50
377D- 26 51      ROL  $51
377F- 26 52      ROL  $52
3781- 26 53      ROL  $53
3783- 38        SEC
3784- A5 52      LDA  $52
3786- E5 54      SBC  $54
3788- AA        TAX
3789- A5 53      LDA  $53
378B- E5 55      SBC  $55
378D- 90 06      BCC  $3795
378F- 86 52      STX  $52
3791- 85 53      STA  $53
3793- E6 50      INC  $50
3795- 88        DEY
3796- D0 E3      BNE  $377B
3798- 60        RTS

```

Listing 2: MEMORY MAP strings.

String data. All strings are stored with horizontal tab in first byte, length of string in second byte, and string characters in reverse order in the remaining bytes. Reverse order is used to allow simple decrementing of the counter instead of incrementing and comparing.

```

3800- 00 0D A0 A0 BA 00 C1 CD
3808- A0 09 D2 CF CD C5 CD 00
3810- 09 D4 C6 CF D3 C5 CC 00
3818- D0 C1 0D 0D C3 C9 D3 C1
3820- C2 A0 D2 C5 C7 C5 D4 CE
3828- C9 00 05 D3 C5 D2 C9 C8
3830- 00 05 B6 B7 B5 B4 B2 00
3838- 05 B0 B0 B0 B6 A4 00 05
3840- B4 B8 B3 B6 B1 00 05 B0
3848- B0 B0 B4 A4 00 05 B2 B9
3850- B1 B8 A0 00 05 B0 B0 B0
3858- B2 A4 0E 04 AC D3 CF C4
3860- 0B 07 A8 A0 D3 C5 CC C9
3868- C6 0C 07 D3 C7 CE C9 D2
3870- D4 D3 0B 09 D4 C6 CF D3
3878- C5 CC D0 D0 C1 0C 07 CD
3880- C1 D2 C7 CF D2 D0 0B 09
3888- D3 C5 CC C2 C1 C9 D2 C1
3890- D6 0D 06 D3 D9 C1 D2 D2
3898- C1 0B 0A C5 C3 C1 D0 D3
38A0- A0 C5 C5 D2 C6 00 0A BF
38A8- A9 D9 A8 A0 D4 CE C9 D2
38B0- D0 FF

```

Listing 3: Starting locations for printing the moved text page.

Left edge locations of the top 23 lines of the moved text page, in reverse order. The 24th line containing the "PRINT [Y]?" statement is not printed. As an example, the last two bytes in this section are \$00 and \$39, denoting the address \$3900. This location holds the byte moved from \$400, the leftmost character on the top line of text page one.

```

38B2- 50 3C D0 3B 50 3B
38B8- D0 3A 50 3A D0 39 50 39
38C0- A8 3C 28 3C A8 3B 28 3B
38C8- A8 3A 28 3A A8 39 28 39
38D0- 80 3C 00 3C 80 3B 00 3B
38D8- 80 3A 00 3A 80 39 00 39

```

This completes the description of the program. Use the loading instructions which follow, then try recreating the examples shown in Part 1 of this series. You will soon figure out many other ways to use memory maps as an aid in designing Integer BASIC and Applesoft programs.

MICRO



PET & APPLE II USERS

TINY PASCAL

Plus +
GRAPHICS



The TINY Pascal System turns your APPLE II micro into a 16-bit P-machine. You too can learn the language that is slated to become the successor to BASIC. TINY Pascal offers the following:

- LINE EDITOR to create, modify and maintain source
- COMPILER to produce P-code, the assembly language of the P-machine
- INTERPRETER to execute the compiled P-code (has TRACE)
- Structured programmed constructs: CASE-OF-ELSE, WHILE-DO, IF-THEN-ELSE, REPEAT-UNTIL, FOR-TO/DOWNTO-DO, BEGIN-END, MEM, CONST, VAR ARRAY

Our new TINY Pascal PLUS+ provides graphics and other built-in functions: GRAPHICS, PLOT, POINT, TEXT, INKEY, ABS AND SQR. The PET version supports double density plotting on 40 column screen giving 80 x 50 plot positions. The APPLE II version supports LORES and for ROM APPLESOFT owners the HIRES graphics plus other features with: COLOR, HGRAPHICS, HCOLOR, HPLOT, PDL and TONE. For those who do not require graphics capabilities, you may still order our original Tiny Pascal package.

TINY Pascal PLUS+ GRAPHICS VERSION-

PET 32K NEW Roms cassette.....\$55
 PET 32K NEW Roms diskette.....\$50
 APPLE II 32K/48K w/DOS 3.2 or 3.3.....\$50



TINY Pascal NON-GRAPHICS VERSIONS-

PET 16K/32K NEW Roms cassette.....\$40
 PET 16K/32K NEW Roms diskette.....\$35
 APPLE II w/ROM Applesoft 32K w/DOS.....\$35
 APPLE II w/RAM Applesoft 48K w/DOS.....\$35

USER'S Manual (refundable with software order).....\$10
 6502 Assembly Listing of INTERPRETER-graphics.....\$25
 6502 Assembly Listing of INTERPRETER-non graphics.....\$20



FREE postage in U.S. and CANADA. Orders may be prepaid by bankcard (include card number and expiration date). Michigan residents include 4% state sales tax. Orders accepted via THE SOURCE - CLO882.



ABACUS SOFTWARE

P. O. Box 7211
Grand Rapids, Michigan 49510

LOOK!!!

AVANT-GARDE CREATIONS has SOFTWARE:

EDUCATION
 ART/DESIGN
 GAMES
 BUSINESS
 UTILITIES

SELF TRANSFORMATION

LOOK AGAIN!!!

We have the following and MORE!:

5 Great Games! Animal Bingo, Jungle Safari, Space Defense, Sky Watcher, Air Traffic Controller \$29.95 (or \$9.95 each)
5 More Great Games! Deep Sea Treasure, Mystery Code, Depth Charge, The Mine Fields of Normalcy, Turn 'Em Loose \$29.95 (or \$9.95 each)

The Mailing Label & Filing System Filing, label-making, binary sort, dynamic sorting, directory, quick-find, more! \$24.95
Sentence Diagramming Educational, grades 6-12 \$19.95
Action Sounds & Hi-Res Scrolling Designed to give your program the excitement of **action & sound** \$15.95
Super Draw & Write Fonts, drawing, and useful utilities \$15.95

Super Shape Draw! The best system yet, it works!

...creates shape tables like a dream... \$19.95

The Creativity Package Draw, write poetry, music \$19.95

"...impressive...satisfying...interesting...fun!" Peellings (The Magazine of Software Reviews)

"Truly different...unique...the program is an enjoyable one...cute...very interesting...new...nice...a good value!" Apple Orchard (Winter)

Demo Disk I Some of our best stuff \$9.95

Demo Disk II More of our best \$9.95

All of our software is written in Applesoft*, 48K, disk

AVANT-GARDE CREATIONS

DEALER INQUIRIES INVITED

P.O. Box 30160

VISA/MASTERCARD

Eugene, OR 97403 Dept. mi

(503) 345-3043

*Apple is a trademark of Apple Computer, Inc.

[12pm-6pm 7 days a week]

Presenting.....

A-STAT™ 79

A Statistical Analysis and File Maintenance System for the Apple II™ Microcomputer*

A subset language of P-STAT™ 78 computes:

FREQUENCIES
 BI-VARIATE TABLES — CHI SQUARES
 CORRELATION MATRICES
 MULTIPLE REGRESSIONS
 APPLE FILE CABINET INTERFACE
 COMPLETE VARIABLE TRANSFORMATIONS

Uses Standard DOS Text Files and EXEC's

A-STAT™ 79 on disk with 80-page manual... \$125.00

48K version — All programs in Applesoft™

Available from:

Rosen Grandon Associates
 296 Peter Green Road
 Tolland, Connecticut 06084
 (203) 875-3541

* Apple II™ is a trademark of the Apple Computer, Inc.

P-STAT™ 78 is a trademark of P-STAT Inc., Princeton, N.J.

A-STAT™ 79 is copyrighted by Gary M. Grandon, Ph.D.

Decision Systems

Decision Systems
P.O. Box 13006
Denton, TX 76203

SOFTWARE FOR THE APPLE II*

ISAM-DS is an integrated set of Applesoft routines that gives indexed file capabilities to your BASIC programs. Retrieve by key, partial key or sequentially. Space from deleted records is automatically reused. Capabilities and performance that match products costing twice as much.
\$50 Disk, Applesoft.

PBASIC-DS is a sophisticated preprocessor for structured BASIC. Use advanced logic constructs such as IF...ELSE..., CASE, SELECT, and many more. Develop programs for Integer or Applesoft. Enjoy the power of structured logic at a fraction of the cost of PASCAL.
\$35. Disk, Applesoft (48K, ROM or Language Card).

DSA-DS is a dis-assembler for 6502 code. Now you can easily dis-assemble any machine language program for the Apple and use the dis-assembled code directly as input to your assembler. Dis-assembles instructions and data. Produces code compatible with the S-C Assembler (version 4.0), Apple's Toolkit assembler and others.
\$25 Disk, Applesoft (32K, ROM or Language Card).

FORM-DS is a complete system for the definition of input and output forms. FORM-DS supplies the automatic checking of numeric input for acceptable range of values, automatic formatting of numeric output, and many more features.
\$25 Disk, Applesoft (32K, ROM or Language Card).

UTIL-DS is a set of routines for use with Applesoft to format numeric output, selectively clear variables (Applesoft's CLEAR gets everything), improve error handling, and interface machine language with Applesoft programs. Includes a special load routine for placing machine language routines underneath Applesoft programs.
\$25 Disk, Applesoft.

SPEED-DS is a routine to modify the statement linkage in an Applesoft program to speed its execution. Improvements of 5-20% are common. As a bonus, SPEED-DS includes machine language routines to speed string handling and reduce the need for garbage clean-up. Author: Lee Meador.
\$15 Disk, Applesoft (32K, ROM or Language Card).

(Add \$4.00 for Foreign Mail)

*Apple II is a registered trademark of the Apple Computer Co.

SOFTWARE UNLIMITED

presenting the LARGEST SELECTION OF SOFTWARE EVER ASSEMBLED...

for ATARI® • APPLE® • PET® • and other Microcomputers
at SUPER DISCOUNT PRICES!

ATARI

<input type="checkbox"/> PHYSICS (AT)	24.50
<input type="checkbox"/> GREAT CLASSICS (AT)	24.50
<input type="checkbox"/> BASIC PHYSIOLOGY (AT)	24.50
<input type="checkbox"/> PRINCIPLES OF ECONOMICS (AT)	24.50
<input type="checkbox"/> SPELLING (AT)	25.50
<input type="checkbox"/> BASIC ELECTRICITY (AT)	24.50
<input type="checkbox"/> BASIC ALGEBRA (AT)	24.50
<input type="checkbox"/> 8K RAM MODULE (AT)	53.95
<input type="checkbox"/> 16K RAM MODULE (AT)	170.00
<input type="checkbox"/> KINGDOM (AT)	13.55
<input type="checkbox"/> LEMONADE (AT)	13.55
<input type="checkbox"/> STATISTICS I (AT)	17.95
<input type="checkbox"/> BLACKJACK (AT)	13.55
<input type="checkbox"/> BIORYTHM (AT)	13.55
<input type="checkbox"/> HANGMAN (AT)	13.55
<input type="checkbox"/> SPACE INVADERS (AT)	17.95
<input type="checkbox"/> EUROPEAN CAPITALS (AT)	13.55
<input type="checkbox"/> MORTGAGE LOAN (AT)	13.55
<input type="checkbox"/> STATES & CAPITALS (AT)	13.55
<input type="checkbox"/> EDUCATION SYSTEM (AT)	22.50
<input type="checkbox"/> ATARI BASIC (AT)	53.95
<input type="checkbox"/> ASSEMBLER DEBUG (AT)	53.95
<input type="checkbox"/> BASKETBALL (AT)	35.95
<input type="checkbox"/> VIDEO EASEL-LIFE (AT)	35.95
<input type="checkbox"/> SUPER BREAKOUT (AT)	35.95
<input type="checkbox"/> MUSIC COMPOSER (AT)	53.95
<input type="checkbox"/> COMPUTER CHESS (AT)	35.95
<input type="checkbox"/> 3-D TIC TAC TOE (AT)	35.95
<input type="checkbox"/> STAR RAIDERS (AT)	53.95
<input type="checkbox"/> TELELINK (AT)	22.50
<input type="checkbox"/> PADDLES (AT)	17.95
<input type="checkbox"/> JOYSTICKS (AT)	17.95
<input type="checkbox"/> U.S. HISTORY (AT)	24.50
<input type="checkbox"/> U.S. GOVERNMENT (AT)	24.50
<input type="checkbox"/> SUPERVISORY SKILLS (AT)	24.50
<input type="checkbox"/> WORLD HISTORY (AT)	24.50
<input type="checkbox"/> BASIC SOCIOLOGY (AT)	24.50

ADVENTURE INTERNATIONAL

<input type="checkbox"/> ADVENTURE HINT SHEET	7.95
<input type="checkbox"/> ADVENTURE (1,2,3) [D] (AP)	35.95
<input type="checkbox"/> ADVENTURE (4,5,6) [D] (AP)	35.95
<input type="checkbox"/> ADVENTURE (7,8,9) [D] (AP)	35.95
<input type="checkbox"/> ADVENTURE #10 [D]	18.95
<input type="checkbox"/> ADVENTURE (specify 1-10) (AP)	13.55
<input type="checkbox"/> PLANETIDS "ASTEROIDS" [D] (AP)	17.95
<input type="checkbox"/> PLANETIDS "ASTEROIDS" (AP)	13.55
<input type="checkbox"/> POKER (AP)	13.55
<input type="checkbox"/> POKER (AP) [D]	18.95
<input type="checkbox"/> KID VENTURE #1 (AP)	13.55

AVALON HILL

<input type="checkbox"/> MIDWAY (P,AP)	13.50
<input type="checkbox"/> NUKE WAR (P,AP)	13.50
<input type="checkbox"/> PLANET MINERS (P,AP)	13.50
<input type="checkbox"/> CONVOY RAIDER (P,AP)	13.50
<input type="checkbox"/> B1 BOMBER (P,AP)	13.50
<input type="checkbox"/> LORDS OF KARMA (P,AP)	18.00

AUTOMATED SIMULATION

<input type="checkbox"/> TUESDAY QUARTERBACK [D] (AP)	26.95
<input type="checkbox"/> STAR WARRIOR [C,D] (AP)	35.95
<input type="checkbox"/> THREE PACK [D] (AP,P)	45.00
<input type="checkbox"/> STARFLEET ORION [C,D] (AP)	22.50
<input type="checkbox"/> STARFLEET ORION [C] (P)	22.50
<input type="checkbox"/> INVASION ORION [C,D] (AP)	22.50
<input type="checkbox"/> INVASION ORION [C] (P)	22.50
<input type="checkbox"/> AP SHAI [D] (AP)	39.95
<input type="checkbox"/> AP SHAI [C] (P)	39.95
<input type="checkbox"/> RYN [D,C] (AP)	17.95
<input type="checkbox"/> RYN [C] (P)	17.95
<input type="checkbox"/> MORLOC [C,D] (AP)	17.95
<input type="checkbox"/> MORLOC [C] (P)	17.95
<input type="checkbox"/> RIGEL [C,D] (AP)	26.95
<input type="checkbox"/> RIGEL [C] (P)	26.95
<input type="checkbox"/> HELLFIRE [D] (AP)	35.95
<input type="checkbox"/> HELLFIRE [C] (P)	35.95

QUALITY SOFTWARE

<input type="checkbox"/> 6502 DISASSEMBLER (AT)	10.55
<input type="checkbox"/> ASTRO APPLE (AP)	13.55
<input type="checkbox"/> ASTRO APPLE (AP) [D]	17.95
<input type="checkbox"/> ASTEROIDS IN SPACE [D] (AP)	17.95
<input type="checkbox"/> ATARI ASSEMBLER (AT)	22.50
<input type="checkbox"/> BATTLE (AP)	13.55
<input type="checkbox"/> BATTLE (AP) [D]	17.95
<input type="checkbox"/> BATTLESHIP COMMANDER (AP)	13.55
<input type="checkbox"/> BATTLESHIP COMMANDER [D] (AP)	17.95
<input type="checkbox"/> BENEATH APPLE MANOR (AP)	13.55
<input type="checkbox"/> BENEATH APPLE MANOR (AP) [D]	17.95
<input type="checkbox"/> FASTGAMMON [D] (AP)	22.50
<input type="checkbox"/> FASTGAMMON (AP,AT)	17.95
<input type="checkbox"/> FORTH [D] (AT)	70.00
<input type="checkbox"/> FRACAS ADVENTURE (AP)	17.95
<input type="checkbox"/> FRACAS ADVENTURE [D] (AP)	22.50
<input type="checkbox"/> LINKER (AP) [D]	44.00
<input type="checkbox"/> TANK TRAP (AT)	10.55
<input type="checkbox"/> TANK TRAP (AT) [D]	13.55
<input type="checkbox"/> TARI TREK (AT)	10.55
<input type="checkbox"/> TARI TREK (AT) [D]	13.55

PERSONAL SOFTWARE

<input type="checkbox"/> CCA DATA MGMT [D] (AP)	85.00
<input type="checkbox"/> DESKTOP PLAN [D] (AP)	85.00
<input type="checkbox"/> GAMMON GAMBLER (AP)	17.95
<input type="checkbox"/> GAMMON GAMBLER [D] (AP)	22.50
<input type="checkbox"/> MONTY MONOPOLY [D] (AP)	31.55
<input type="checkbox"/> VISICALC [D] (AP)	125.00
<input type="checkbox"/> VISICALC [D] (AT,P)	170.00

INSTANT SOFTWARE

<input type="checkbox"/> AIR FLIGHT SIMULATION (AP)	8.95
<input type="checkbox"/> APPLE FUN [D] (AP)	17.95
<input type="checkbox"/> CASINO (P)	7.25
<input type="checkbox"/> MORTGAGE (P)	7.25
<input type="checkbox"/> PADDLE FUN [D] (AP)	17.95
<input type="checkbox"/> PENNY ARCADE (P)	7.25
<input type="checkbox"/> PET UTILITY (P)	8.95
<input type="checkbox"/> QUBIC 4/GOMOKU (P)	7.25
<input type="checkbox"/> SANTA PARAVIA FIUMACCIO (AP,P)	8.95
<input type="checkbox"/> SANTA PARAVIA FIUMACCIO (AP)[D]	17.95
<input type="checkbox"/> SAHARA WARRIOR (AP)	7.25
<input type="checkbox"/> SKY BOMBERS (AP) [D]	17.95
<input type="checkbox"/> SPACE WARS (AP)	7.25
<input type="checkbox"/> SUPERSHOOTERS (AP)	8.95
<input type="checkbox"/> TREK-X (P)	7.25

STRATEGIC SIMULATIONS

<input type="checkbox"/> COMPUTER AMBUSH [D] (AP)	51.50
<input type="checkbox"/> COMPUTER BISMARCK [D] (AP)	51.50
<input type="checkbox"/> COMPUTER CONFLICT [D] (AP)	35.00
<input type="checkbox"/> COMPUTER NAPOLEONICS [D] (AP)	51.50
<input type="checkbox"/> COMPUTER QUARTERBACK [D] (AP)	35.00
<input type="checkbox"/> AIR COMBAT [D] (AP)	51.50
<input type="checkbox"/> WARP FACTOR [D] (AP)	35.00

SUB-LOGIC

<input type="checkbox"/> 3D GRAPHICS (AP)	40.00
<input type="checkbox"/> 3D GRAPHICS [D] (AP)	48.00
<input type="checkbox"/> A2-FS1 FLIGHT SIMULATOR (AP)	22.00
<input type="checkbox"/> A2-FS1 FLIGHT [D] (AP)	29.00

MICROSOFT SOFTWARE

<input type="checkbox"/> ADVENTURE [D] (AP)	25.50
<input type="checkbox"/> OLYMPIC DECATHALON [D] (AP)	20.00
<input type="checkbox"/> TYPING TUTOR (AP) [D]	17.95
<input type="checkbox"/> TYPING TUTOR (AP)	13.55
<input type="checkbox"/> Z-80 SOFTCARD [D] (AP)	280.00
<input type="checkbox"/> 16K RAMBOARD	165.00

ON LINE SYSTEMS

<input type="checkbox"/> HI-RES ADVEN #0 (AP) [D]	17.95
<input type="checkbox"/> HI-RES ADVEN #1 [D] (AP)	22.50
<input type="checkbox"/> HI-RES ADVEN #2 [D] (AP)	29.00
<input type="checkbox"/> HI-RES FOOTBALL #1 [D] (AP)	36.00
<input type="checkbox"/> HI-RES CRIBAGGE [D] (AP)	22.50
<input type="checkbox"/> PADDLE GRAPHICS [D] (AP)	36.00
<input type="checkbox"/> TABLET GRAPHICS [D] (AP)	44.95

SIRIUS

<input type="checkbox"/> CYBER STRIKE [D] (AP)	36.00
<input type="checkbox"/> STAR CRUISER [D] (AP)	22.50
<input type="checkbox"/> BOTH BARRELS [D] (AP)	22.50
<input type="checkbox"/> PHANTOM FIVE [D] (AP)	36.00

SYNERGISTIC SOFTWARE

<input type="checkbox"/> DUNGEON & WILDERNESS [D] (AP)	29.00
<input type="checkbox"/> DUNGEON (AP)	13.50
<input type="checkbox"/> DUNGEON [D] (AP)	15.75
<input type="checkbox"/> ODYSSEY [D] (AP)	27.00
<input type="checkbox"/> HIGHER GRAPHICS [D] (AP)	31.50
<input type="checkbox"/> WILDERNESS (AP)	15.75
<input type="checkbox"/> WILDERNESS [D] (AP)	18.00

BORDERBUND

<input type="checkbox"/> EMPIRE GALACTIC (AP) [D]	22.50
<input type="checkbox"/> GALAXIAN (AP) [D]	22.50
<input type="checkbox"/> HYPER HEAD ON (AP) [D]	22.50
<input type="checkbox"/> REVOLUTION GALACTIC (AP) [D]	22.50
<input type="checkbox"/> TANK (AP) [D]	13.55
<input type="checkbox"/> TAWALA'S REDDOUBT (AP) [D]	26.95
<input type="checkbox"/> TRADER GALACTIC (AP) [D]	22.50

MUSE COMPANY

<input type="checkbox"/> ABM [D] (AP)	22.50
<input type="checkbox"/> ADDRESS BOOK (AP) [D]	44.50
<input type="checkbox"/> APPLOT II [D] (AP)	80.00
<input type="checkbox"/> BEST OF MUSE (AP) [D]	35.95
<input type="checkbox"/> GLOBAL WAR (AP) [D]	22.50
<input type="checkbox"/> MATH-APPLESOFT (AP) [D]	35.95
<input type="checkbox"/> SUPER TEXT II (AP) [D]	135.00
<input type="checkbox"/> THREE MILE ISLAND (AP) [D]	35.95
<input type="checkbox"/> U-DRAW II (AP) [D]	35.95
<input type="checkbox"/> THE VOICE (AP) [D]	35.95

IRIDIS

<input type="checkbox"/> IRIDIS 1 (AT)	8.95
<input type="checkbox"/> IRIDIS 1 (AT) [D]	11.75
<input type="checkbox"/> IRIDIS 2 (AT)	14.50
<input type="checkbox"/> IRIDIS 2 (AT) [D]	16.95

EDU-WARE

<input type="checkbox"/> COMPU READ (AP) [D]	22.50
<input type="checkbox"/> ESP (AP) [D]	14.50
<input type="checkbox"/> NETWORK (AP) [D]	17.95
<input type="checkbox"/> PRISONER (AP) [D]	26.95
<input type="checkbox"/> SPACE (AP) [D]	26.95
<input type="checkbox"/> SPACE II (AP) [D]	22.50
<input type="checkbox"/> TERRORIST (AP) [D]	26.95

PROGRAMMA

<input type="checkbox"/> MICRO INVADERS (AP)	14.50
<input type="checkbox"/> EXPAND-A-PORT (AP)	53.95
<input type="checkbox"/> JOYSTICK (AP)	35.95
<input type="checkbox"/> TINY PASCAL (AP) [D]	44.50
<input type="checkbox"/> SPACE WARS (AP,P)	8.95
<input type="checkbox"/> WPS STANDARD (AP) [D]	117.00

HAYDEN

<input type="checkbox"/> SARGON II (AP)	25.00
<input type="checkbox"/> SARGON II (AP) [D]	30.00
<input type="checkbox"/> REVERSAL (AP)	25.00

If you don't see it listed, write...we probably have it in stock!

Check program desired.
Complete ordering information
and mail entire ad.
Immediate Shipments from stock.

KEY:

AT-Atari
AP-Apple
P-Pet
D-on Disc.
C-Cassette

If not marked-Cassette

ATARI is a trademark of ATARI INC.
APPLE is a trademark of APPLE COMPUTER, INC.
PET is a trademark of COMMODORE BUSINESS MACHINES.

Prices subject to change without notice.

Ship the above programs as checked to:

Mr./Mrs. _____
Address _____
City _____
State _____ Zip _____
I have a _____ name of Computer _____
with _____ k memory

Number of Programs Ordered _____

Amount of order _____

N.Y. residents add Sales Tax _____

Add shipping anywhere in the U.S. **2.00**

Total amount enclosed _____

Charge my: Master Charge Visa

Signature _____

Card No. _____ Expires _____

Personal Checks please allow 3 weeks

Mail to:

DIGIBYTE SYSTEMS CORP.

31 East 31st Street, New York, N.Y. 10016

Phone: (212) 889-8975

The Atari Dulcimer

The Atari 800 comes with four musical voices under program control through BASIC. The following program uses these voices to simulate a three-string plucked dulcimer, played in real time.

Mike Dougherty
Box 230, Rt. 5
Kingston, Tennessee 37763

The Atari 800 personal computer has many outstanding features. The four musical voices caught my attention first for three reasons: I enjoy music, I had not used a computer with musical capability, and the sound voices were easy to control from BASIC. Having musical experience with a plucked dulcimer, it was natural for me to attempt to simulate this instrument with the Atari 800.

A traditional Appalachian-plucked dulcimer consists of a hollow, fretted fingerboard on top of a shallow sound box extending symmetrically on either side. The three-string dulcimer utilizes a single string to carry the melody, with the two remaining strings supplying a background harmonic "drone." The plucked dulcimer is typically played on the lap, the right hand strumming all of the strings with a pick, the left hand pressing the melody string to the frets with a "noter" stick. A background rhythm is impressed on the background drone and melody by strumming across the strings at different rates. Most dulcimers have a scale consisting of approximately sixteen notes with no sharps or flats. In general, the dulcimer notes range from the G below middle C to the A in the octave above middle C.

```

1 REM --- ATARI DULCIMER
2 REM ... BY MIKE DOUGHERTY
3 REM
10 DIM NT(255): REM KEY/NOTE TABLE
20 DIM LE(10): REM STRUM DURATION
100 GOSUB 10000: REM INITIALIZATION
1000 REM
1010 REM --- MAIN SOUND LOOP
1020 REM
1100 FOR LOOP = 0 TO 1 STEP 0
1200 FOR STRUM = 1 TO N
1300 FOR DUR = 6 TO 1 STEP - LE(STRUM)
1310 SOUND 0,163,10,DUR:SOUND 1,243,10,DUR:SOUND 2,161,10,DUR
1400 FOR WAIT = 0 TO TEMPO
1410 KEY = PEEK (764)
1420 SOUND 3,NT(KEY),10,DUR+3
1430 NEXT WAIT
1500 NEXT DUR
1510 SOUND 0,0,0,0:SOUND 2,0,0,0:SOUND 3,0,0,0
1600 NEXT STRUM
1610 IF PEEK (764) = 28 THEN GOTO 20000: REM RESTART PROGRAM
1700 NEXT LOOP
10000 REM
10010 REM --- INITIALIZE NOTES AND
10020 REM --- VARIABLES FOR DULCIMER
10040 REM
10050 GOSUB 30000: REM PRINT KEYBOARD
10100 KEY = 255: REM INITIAL NOTE = NULL
10130 FOR I = 0 TO 22: READ T1,T2:NT(T1) = T2: NEXT I
10200 DATA 47,173,63,162,46,153
10210 DATA 62,144,42,136,58,128
10220 DATA 56,121,45,114,61,108
10230 DATA 43,102,57,96,1,91
10240 DATA 13,85,5,81,8,76
10250 DATA 0,72,10,68,2,64
10260 DATA 6,60,15,57,7,53
10270 DATA 12,50,60,47
10300 PRINT "TEMPO ";: INPUT TEMPO
10400 PRINT "# OF STRUMS/LOOP (MAXIMUM 10) ";: INPUT N
10410 FOR I = 1 TO N
10420 PRINT "LENGTH OF STRUM # ";I;" ";: INPUT T1:LE(I) = T1
10430 NEXT I
10900 RETURN
20000 REM
20001 REM --- CLEAN UP AND RESTART
20002 REM
20010 SOUND 0,0,0,0:SOUND 1,0,0,0:SOUND 3,0,0,0
20020 RUN
30000 REM
30001 REM --- PRINT THE NOTE/KEY
30002 REM --- CORRESPONDENCE ON
30003 REM --- THE SCREEN
30004 RFM
30006 GRAPHICS 0:SETCOLOR 2,9,1:SETCOLOR 4,3,4
30007 PRINT "ATARI DULCIMER": PRINT : PRINT
30010 PRINT "ATARI Q W E R T Y U I O P - C SHARP"
30020 PRINT "KEYS: "
30025 PRINT " "
30030 PRINT " "
30040 PRINT " "
30050 PRINT "MUSIC"
30060 PRINT "NOTES: G A BC D EF G A BC D E "
30070 PRINT " "
30080 PRINT " - MIDDLE C "
30100 RETURN

```

Thus a simulation of a plucked dulcimer must contain at least the following elements:

1. a single voice melody,
2. a background drone of voices,
3. a method to impress the strumming rhythm,
4. the ability to do the above in real time.

The Atari 800 keyboard was chosen for the melody input. The Atari Dulcimer maps the "standard key row" of A, S, D, F, ..., +, *, (caps lower) onto the thirteen notes of G, A, B, middle C, ..., E above middle C. In addition, the Atari Dulcimer also maps the row of keys Q, W, E, ... =, (return) onto the sharp notes. The mapping information is maintained in the 256 element "NTE" array. The value of the current key pressed is determined by PEEK[764] and used as the index into "NTE". Each element of "NTE" contains either the proper pitch for that key, or a zero (which effectively turns off the melody voice). Thus the following two BASIC lines read the keyboard and play either a note or a "rest" on the melody, voice #3:

```
KEY = PEEK(764)
SOUND 3,NTE(KEY),10,-
```

Note that the keyboard space bar makes a very convenient "rest note" for the Atari Dulcimer. Although only 23 of the 256 "NTE" elements represent actual notes, this method allows direct table lookup of the pitch values for faster execution. Without this "wasteful" technique, real time playing of the Atari Dulcimer would be severely hampered.

The background drone is simulated with the three remaining Atari voices: voice #1 sounding C below middle C, and voices #0 and #2 combining to sound G below middle C. The base G was simulated by two voices, each voice one value off the "true" pitch. This small discord gives a proper "twang" for a string sound. To maintain the background nature of the drone, each background voice is played at a loudness of 3 levels below that of the melody voice.

A strumming effect is impressed upon the strings by allowing the loudness to decrease linearly with time. The length of the strum is determined by the step size of the loop:

```
FOR DUR = 6 TO 0 STEP
-LENGTH(STRUM)
.
.
. execute either
. SOUND --,DUR + 3 or
. SOUND --,DUR
NEXT DUR
```

Thus LENGTH[STRUM]=1 is the slowest possible strum while LENGTH[STRUM]=6 is the fastest strum. (In general, $1 < =$ LENGTH[STRUM] $< = 3$ gives the best results.) The step size was chosen to control the duration of the loop instead of the limit (fixed at 6) so that both short and long strums would start at the same loudness. The current Atari Dulcimer allows for the definition of up to 10 different length strums in a song.

The overall speed of the innermost delay loop is controlled by the "TEMPO" variable. The fastest possible tempo (speed) of the program is with a zero "TEMPO". To play a song with no strumming, simply use a large value for "TEMPO". At the end of each set of strums, the keyboard is checked for the escape key. If the dulcimer player has played the note "ESC", then the program stops all of the voices and restarts.

Table 1: Program Variables in Atari Dulcimer

KEY	last keyboard key pressed, stored in internal code
NT[255]	for each key pressed, as determined by PEEK[764], NTE[KEY] is the pitch for the sound command
N	number of strums in background harmonic drone
LE[10]	array containing the step increment that determines the duration of each strum: $1 < =$ LENGTH(STRUM) $< = 6$

TEMPO	overall speed of the Atari Dulcimer — the limit of the innermost loop
LOOP	outermost sound loop index. LOOP uses a step size of zero to form an infinite loop — this method is faster than the use of GOTOs
STRUM	loop index for each user defined strum — the duration of each strum is controlled by LENGTH[STRUM]
DUR	loop index for strum loop — DUR controls the overall loudness of the music voices
WAIT	loop index for delaying the innermost loop to the limit of TEMPO

Table 2: Sample Song Parameters for the Atari Dulcimer

Song Title	TEMPO	STRUMS	LENGTH(1)	LENGTH(2)	LENGTH(3)
"Wildwood Flower"	1	3	1	3	3
"The Battle Hymn of the Republic"	1	2	1	3	—
"O Come All Ye Faithful"	3	2	2	3	—
"Loch Lomand"	3	2	2	4	—

Mike Dougherty graduated from the University of Tennessee in 1977 with a M.S. in Computer Science, and has been employed by Union Carbide at the Oak Ridge National Laboratory since that time. He has worked on several projects involving computers from the VAX 11/780, down to single board microprocessors. His home-based system presently consists of an Atari 800 with 24K bytes of memory.



Big Savings On Atari & PET!

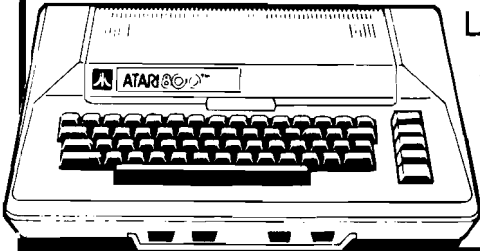
No Risk - No Deposit On Phone
Orders - Shipped Same Day You
Call - C.O.D. or Credit Card

* On all in stock units

Please Call Between 11AM & 6PM
(Eastern Standard Time)

(800) 233-8950

ATARI® 800™ PERSONAL COMPUTER

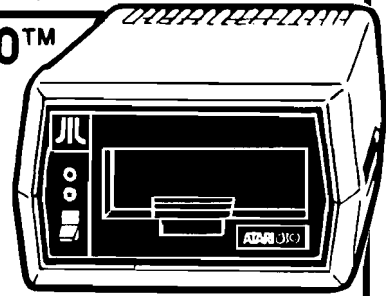


List \$1080
\$759



ATARI® 810™ DISC DRIVE

List \$599.95
**New Low Price
only \$489.00**



Maxell Disks 10 for \$34
Sycorn Disks 10 for 29
Atari Disks 5 for 22

400 8K	\$419
400 16K	449
410 Recorder.....	62
815 Disk	1199
822 Printer	359
825 Printer	779
830 Modem.....	159
850 Interface Module	179
CX852 8K RAM	94
CX853 16K RAM	149
CX70 Light Pen.....	64
CX30 Paddle	18
CX40 Joystick	18
CX86 Printer Cable	42
CO16345 822 Thermal Printer Paper	5
CAO16087 825 80-col. Printer Ribbon (3/box).....	17
CX4119 Conversational French	45
CX4118 Conversational German	45
CX4120 Conversational Spanish	45
CX4125 Conversational Italian	45
CXL4009 Chess	30
CXL4011 Star Raiders™	45
CXL4004 Basketball.....	30
CXL4006 Super Breakout™	30
CXL4010 3-D Tic-Tac-Toe	30
CXL4005 Video Easel™	30
CXL4007 Music Composer	45
CXL4015 TeleLink™	20
CXL4002 BASIC Computing Language	46
CXL4001 Education System Master	21
CXL4003 Assembler Editor	45

CX4115 Mortgage & Loan Analysis.....	\$13
CX4101 An Invitation to Programming 1	17
CX4106 An Invitation to Programming 2.....	20
CX4117 An Invitation to Programming 3.....	20
CX4107 Biorhythm.....	13
CX4103 Statistics I	17
CX4121 Energy Czar	13
CX4108 Hangman	13
CX4102 Kingdom.....	13
CX4112 States & Capitals	13
CX4114 European Countries & Captials.....	13
CX4105 Blackjack	13
CX4111 Space Invaders.....	18
CX8106 Bond Analysis	20
CX8107 Stock Analysis	20
CX8108 Stock Charting.....	20
CX4104 Mailing List	17
CX4110 Touch Typing	20
CX8102 Calculator	24
CX4109 Graph It.....	17
CX4120 Conversational Spanish	45
Talk & Teach Courseware; CX6001 thru CX6017	23

CX8104 Atari 810 Master Diskette II

New DOS 2 Operating
System Master

only \$21.00

Microtek RAM 16K or 32K

- Full 1 year warranty
- Compatible with 400 or 800
- Assembled and Tested

16K..... \$ 99
32K..... 189

Atari Specialists . . .

*We Carry
It All*

Combination Special!
825 Printer & 850 Interface
825 Lists for \$1000
Regular Mail Order Price: \$779
850 Lists for \$220.
Regular Mail Order Price: \$179
Save \$20.00
Buy Both For Only \$938

Software

WordPro 3 (40 col.)	\$186
WordPro 4 (80 col.)	279
WordPro 4 Plus (80 col.).....	339

Visicalc - Apple.....	\$122
Atari	163
PET	163

Printers

NEC 5530.....	\$2495
Diablo 630	2195
Trendcom 100	299
Trendcom 200	489
Paper Tiger 445G.....	769
Paper Tiger 460G.....	1219
Epson MX-80	539

commodore



Commodore Computers:

4032 N	\$1090
4032 B	1090
8032	1499

Commodore Peripherals:

CBM 4022 Printer	675
CBM 4040 Dual Drive Floppy Disk	1090
CBM 8050 Dual Drive Floppy Disk	1420
CBM C2N Cassette Drive	87
Tally 8024 Printer	1679

EBS Accounts Receivable Inventory System.....	\$695
Dr. Daley Mailing List	129
Dr. Daley Inventory	89
OZZ Information System	329
BPI General Ledger	329
Tax Package.....	399
Dow Jones Portfolio Management	129
Pascal	239
PET to IEEE Cable	37
IEEE to IEEE Cable	46

NEW -

8096	1890
VIC - 20	299

To Order:

Phone orders invited (800 number is for order desk only). Or send check or money order.
Equipment Shipped UPS collect. Pennsylvania residents add 6% sales tax. Add 3% for
Visa or MC. Equipment is subject to price change and availability without notice.

Computer Mail Order
501 E. Third St.
Williamsport, PA 17701
(717) 323-7921

MICRO

PET Vet

By Loren Wright

Upgrade Decisions

The decision to upgrade from 1.0 to 2.0 ROMs was very easy for many PET users. 2.0 corrected several bugs in the 1.0 ROMs, but the most significant reason to upgrade was to accommodate a disk drive. Other users (without disk drives) stayed with the 1.0 ROMs and have been putting up with the bugs.

The arrival of the 4.0 ROMs presents us with different kinds of decisions. To be sure, Commodore has stopped making 2.0 ROM machines, and these will eventually receive less support in hardware and software. That reason alone is not sufficient for an upgrade. After all, 1.0 ROM PETs are still alive and reasonably well.

The most important reason is the new DOS — 2.1 if you buy a 4040 disk drive or upgrade a 2040, 2.5 if you buy an 8050. BASIC 4.0 is built around DOS 2.1/2.5. All commands are handled directly by the DOS, without complicated secondary addresses or having to load a DOS program. A new, efficient, relative-record system has been added, and several other operations have been improved. Most commands require only a file name.

The other major change with BASIC 4.0 is the improvement of the garbage collection process. Every time memory gets tight, BASIC has to clear memory of old copies of dynamic strings. With older BASICs, this could take up to 20 minutes. 4.0 BASIC collects its garbage in less than one second.

Disk-O-Pro: An Alternative to a 4.0 Upgrade?

Disk-O-Pro combines the 4.0 BASIC disk commands (for the 2.1/2.5 DOS) with several other commands and features. It is a 4K ROM, addressed \$9000-\$9FFF, and works with 2.0 ROM (level III BASIC) PETs and CBMs. It is designed to be compatible with the Toolkit. In fact, initializing Disk-O-Pro

will also initialize the Toolkit, if it is present. Disk-O-Pro is available from Skyles Electric Works for \$75.

The SEW (Skyles Electric Works) group includes a number of commands not available in any Commodore BASIC. A few of the commands need more discussion than is presented in the table. The SCROLL command turns a BASIC program listing into a continuous cylinder, which can be moved through the screen, in either direction, with the cursor control keys. Also enabled by the SCROLL command are repeating keys and the "softkey."

The "softkey" is a user-defined sequence of characters, which is executed when the assigned key is hit. The maximum length of this sequence is 60 characters for Disk-O-Pro used with the Toolkit, and 80 characters without the Toolkit.

PRINT USING is a command for formatting output of strings and numbers. This is particularly useful when handling dollar and cent figures. Lining up decimal points, embedding commas, and adding trailing zeroes after the decimal point, can be automatically accomplished with a single PRINT USING statement.

BEEP controls a speaker connected to the CB2 line of the parallel user port. The STOP key acts like the DELETE key, except characters disappear to the right of the cursor.

Because most of Disk-O-Pro's commands work both in immediate and programmed modes, Disk-O-Pro has to intercept the PET's command input every time to check for its own commands. This means that program execution is slowed down—usually less than 20%—but sometimes a lot more. Fortunately, there is a KILL command, so that Disk-O-Pro can be disabled during those parts of the program where execution speed is important.

The disk commands are essentially the same, but there are minor differences, which could pop up unexpectedly. For instance, with Disk-O-Pro, specifying the disk unit [...ON U9] resets the default device number to the one specified. In BASIC 4.0 the default device number is always 8. There are also differences in when the error channel is checked, and whether a carriage return has to be output after each

PRINT# command. The Disk-O-Pro disk commands are recognized by BASIC 4.0, and *vice versa*. The slight differences in interpretation will not be a problem for the average user.

There is some incompatibility between programs written with and without Disk-O-Pro. When writing REM and DATA statements with Disk-O-Pro, REM must be followed with a quote and DATA must be tokenized as "\", otherwise these lines will be unreadable without Disk-O-Pro. Of course commands such as PRINT USING and BEEP will not be recognized without Disk-O-Pro.

The ROM occupies the same 4K block as the protection ROMs for Wordpro and VisiCalc, but these ROMs can be changed manually or by using a programmable ROM switch. Also, Disk-O-Pro does not speed up garbage collection—a major feature of BASIC 4.0.

Finally, I should point out that Disk-O-Pro is not BASIC 4.0, even if it behaves that way. Commercial software written specifically for BASIC 4.0 won't run with BASIC 2.0 and Disk-O-Pro.

Disk-O-Pro adds some really outstanding capabilities to your PET. However, you will have to make your decision based on your own circumstances. Disk-O-Pro offers compatibility with DOS 2.1/2.5, BASIC 4.0, and the Programmer's Toolkit, along with many other useful features. The price of Disk-O-Pro is slightly lower than that of a BASIC 4.0 upgrade. If you already have a Toolkit, then the price difference is more significant, since you would then have to replace your Toolkit with a 4.0 version. However, if you need faster garbage collection, full-speed operation, full compatibility with others' computers, and access to the latest commercial software, then you need BASIC 4.0.

Command-O: Enhancements for the 80-Column CBM

Command-O is also a 4K ROM, addressed \$9000 - \$9FFF, but it is only for 4.0 ROM machines—particularly the CBM 8016/8032. It is available from Skyles Electric Works for \$75. This ROM includes the SEW commands described above for Disk-O-Pro. The only difference is that the "softkey" may be SET to 190 characters. The rest

of the 4K ROM is occupied with the Editing/Debugging commands from the Programmer's Toolkit.

There is also a MOVE command that allows you to position the cursor at any row, column-specified point on the screen. The 'ESCAPE' key is converted to a 'CONTROL' key, enabling more convenient use of the 8016/8032 screen functions. As examples, 'ESCAPE-DELETE' deletes a text line, 'ESCAPE-G' sounds a beep, and 'ESCAPE-CLEAR' sets the upper left corner of the window.

Three of the Toolkit commands are included in improved versions. FIND and RENUMBER allow the operation to be restricted to a specified range of line numbers. TRACE displays each line before it is executed, and a STEP mode is included.

As with Disk-O-Pro there are potential problems with slower execution speed and incompatibility with non-Command-O systems. Since most of Command-O's commands apply only in the immediate mode, it is more convenient to turn the ROM off with the KILL command when execution speed is critical. The combination of the SEW commands and improved Toolkit commands makes Command-O a very significant addition to your system. Unlike Disk-O-Pro, it is not being sold as an alternative to a BASIC upgrade.

Programmer's Toolkit: The Old Standby

The Programmer's Toolkit is a 2K ROM, available in versions for all three Commodore BASICs. The price varies from \$40 for just the ROM, to \$60 for the ROM with an adaptor board that connects to the memory expansion port. The addressing is \$B000 - \$B7FF for 1.0 and 2.0 ROMs, and \$A000 - \$A7FF for 4.0. The Programmer's Toolkit is manufactured by Palo Alto Integrated Circuits (PAICS) and sold not only by them, but also by dealers throughout the country.

This product was reviewed in the August, 1980 MICRO (27:31) by James Strasma. Unlike Disk-O-Pro and Command-O, all its operations take place in the immediate mode. Therefore, there is no problem with incompatibility or slowed execution.

\$F000						
\$E000	2.0 BASIC	2.0 BASIC	2.0 BASIC	4.0 BASIC	4.0 BASIC	4.0 BASIC
\$D000						
\$C000						
\$B000	Toolkit		Toolkit			
\$A000					Toolkit	
\$9000		Disk-O-Pro	Disk-O-Pro			Command-O
Toolkit	\$40		\$40		\$40	
Disk-O-Pro		\$75	\$75			
Command-O						\$75
BASIC 4.0 Upgrade				\$89	\$89	\$89
	\$40	\$75	\$115	\$89	\$129	\$164

Command	Programmer's Toolkit (available 1.0, 2.0, 4.0)	Disk-O-Pro (2.0 only)	Command-O (4.0 only)	4.0 BASIC	Description
SEW Group					
INITIALIZE		X	X		Initialize disk(s).
MERGE		X	X		Disk append (similar to Toolkit "APPEND") or overlay.
EXECUTE		X	X		Load and run a program from disk.
SEND		X	X		Send a disk command.
SCROLL		X	X		Turn on enhanced screen editing [see text].
SET		X	X		Define softkey.
OUT		X	X		Turn off SCROLL functions.
PRINT USING		X	X		Formatted output of numbers and strings.
BEEP		X	X		Controls length and pitch of tone.
KILL		X	X		Remove Disk-O-Pro or Command-O from system.
DOS 2.1/2.5 Group					
CONCAT		X		X	Concatenate one file to another.
DOPEN		X		X	Open disk file.
DCLOSE		X		X	Close disk file.
RECORD		X		X	Position disk at desired relative record.
HEADER		X		X	Formats a disk.
COLLECT		X		X	Cleans up improperly closed files.
BACKUP		X		X	Duplicate one disk onto another.
COPY		X		X	Copies one disk to another without altering the second.
APPEND		X		X	Like DOPEN, but applies only to sequential files.
DSAVE		X		X	Save a BASIC text file on disk.
DLOAD		X		X	Load a BASIC text file from disk.
CATALOG		X		X	Display disk directory.
RENAME		X		X	Change the name of a file.
SCRATCH		X		X	Remove a file from disk.
DIRECTORY		X		X	Display disk directory.
Editing and Debugging Group					
AUTO	X		X		Automatic line numbering.
DUMP	X		X		List values of all non-array variables.
DELETE	X		X		Delete lines within range specified.
FIND	X		X		Find command or string in BASIC program.
HELP	X		X		Indicate errors in BASIC line.
TRACE	X		X		Display program line and execute through program.
OFF	X		X		Turn off trace.
RENUMBER	X		X		Renumber program or program segment.
STEP	X		(X)		Step through program (included in Command-O TRACE).
APPEND	X		(X)		Append program (included in Command-O MERGE).

People Who Know Quality and Need Speed, Flexibility and Reliability

Demand

JINSAMTM 8.0

Data manager for 32K 8000 series Commodore computers.

Want To Know Why?

- ★ Commodore approved software.
- ★ Unlimited definable categories.
- ★ Unlimited record length.
- ★ Unlimited data bases per disk.
- ★ Custom reports and labels.
- ★ Machine sorted by three categories at once.
- ★ Performs calculations and statistics.
- ★ Interface with Word Pro 4TM and/or Word Pro 4+TM
- ★ Recommended by Professional Software

Want To Know More?

See your local dealer, send for descriptive information, or send \$15.00 for your own demonstration disk.

Jini Micro Systems, Inc.

Box 274 M.8
Riverdale, NY 10463

Word Pro and Word Pro 4+ are Trademarks of Professional Software.

An Inexpensive Word Processor

This circuit interfaces an IBM 2740 Communications Terminal to an 8-bit parallel port. Software is included for a PET implementation, but is easily converted to other micro-computers.

William F. Pytlík
9012 Maritime Court
Springfield, VA 22153

Of the many uses of personal computers, one of the most useful and about which much has been written is word processing. For many, word processing is a "nice to have" feature, and in most cases well outside financial means. Daisy Wheel printers and the associated hardware/software needed to use the PET for word processing may cost in excess of \$4000.

Fortunately, an alternative exists for those who wish to use their PETs for limited word processing; that is, the occasional letter, technical report, or magazine article. The alternative is based on the use of the IBM 2740 Communications Terminal. These surplus terminals are available from a variety of sources and range in price from \$100 to \$600 (without interface electronics). All these terminals have one thing in common—they are heavy duty IBM Selectric typewriters, modified with solenoids, which activate the proper mechanical action of the typewriter. Therefore, to use the PET for simple word processing is conceptually simple. First an interface between the PET and the typewriter must be designed. Then appropriate software must be written that will permit creation of text and, via the PET user port, drive the proper Selectric solenoids.

Figure 1 presents a photograph of my system. Note the large keyboard in front of my PET. This keyboard is a standard replacement part available from the Commodore Service Department. The case must be fabricated separately. The keyboard simply plugs into the PET main circuit board in place of the small keyboard. The "black box" between the PET and the Selectric houses the interface electronics.

The interface converts the user port TTL voltage levels to voltage levels required to drive the Selectric solenoids. Figures 2 and 3 present the schematic of interface and power supplies. I made my own printed circuit board, but this circuit can be constructed using a general-purpose hobbyist PC board or by wire-wrapping. The interface converts the 5 volt TTL levels of the PET to the voltage level (35 to 55 VDC) required by the Selectric solenoids. The voltage is not critical — IBM uses 48 VDC. I used a 27 volt transformer which I had available, resulting in a DC voltage of 38 volts. The 38 VDC on my Selectric is applied to pin 2 of the "t" connector. Since other terminals may be different, the positive voltage should be applied

to the appropriate connector for that terminal. The solenoids are activated by grounding the proper solenoid. The proper activation of a combination of four rotate solenoids and two tilt solenoids results in a character being typed. Additionally, several other solenoids are required for control characters, etc. These include a check solenoid (required for all printable characters), space, backspace, index, shift, and carriage return solenoids. Thus, a total of twelve solenoids must be addressed by the eight output port lines. Consequently, a hardware decoding scheme is necessary. An analysis of the schematic (figure 2) readily reveals the decoding scheme.

IC2 and IC3 (74LS126) simply act as buffers between the PET and the interface. IC4, IC5, IC6 (74LS02) are used as decoders and drivers for the solenoid drivers (transistors Q1 to Q12). Thus a "0" at the output port energizes a solenoid by turning on one of the transistors. Transistors Q1 to Q12 (2N3904) are simple switching transistors. I used 2N3904's because they were readily available. They work well switching the 38 volts. If higher voltage levels are used, then a higher voltage transistor must be used (i.e. TIS95).

Figure 1



Two power supplies are required. A one amp unregulated 35 to 55 VDC is required to drive the solenoids while a regulated 5 VDC power supply is needed for the interface logic circuitry. All components (with the exception of the 7500 microfarad, 50 volt capacitor) are mounted on the single PC board. Figure 4 presents a photograph of the completed board mounted in the chassis.

The capability to drive the solenoids now exists. The remaining problem is to provide the proper timing and appropriate code to print the right character or effect proper operation. Table 1 presents the code conversion details. To pick a solenoid requires approximately 10 milliseconds (ms). To complete the mechanical action of printing a character requires an additional 60 ms. The time the carriage return requires is considerably longer. Since printing each character takes a minimum of 70 ms, there is no need for a machine language program. A BASIC program can adequately drive the typewriter at its maximum speed.

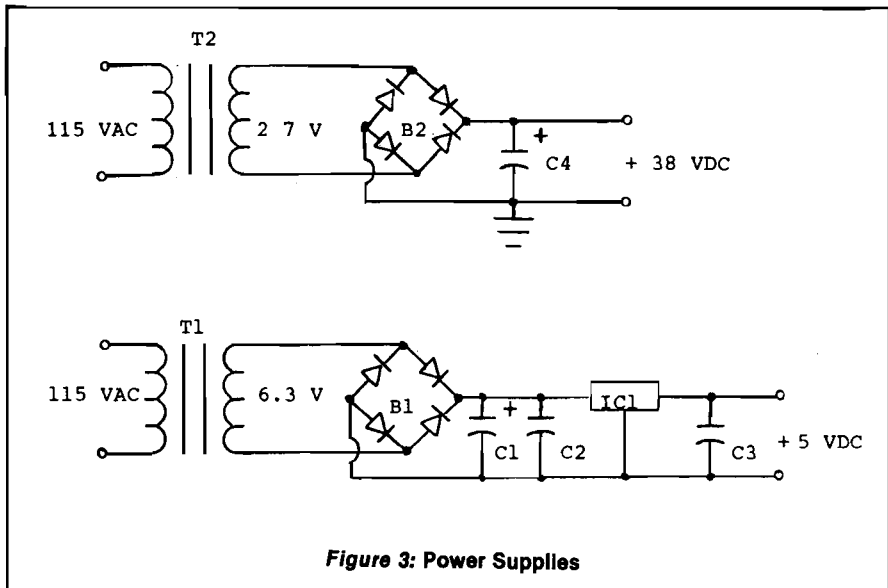
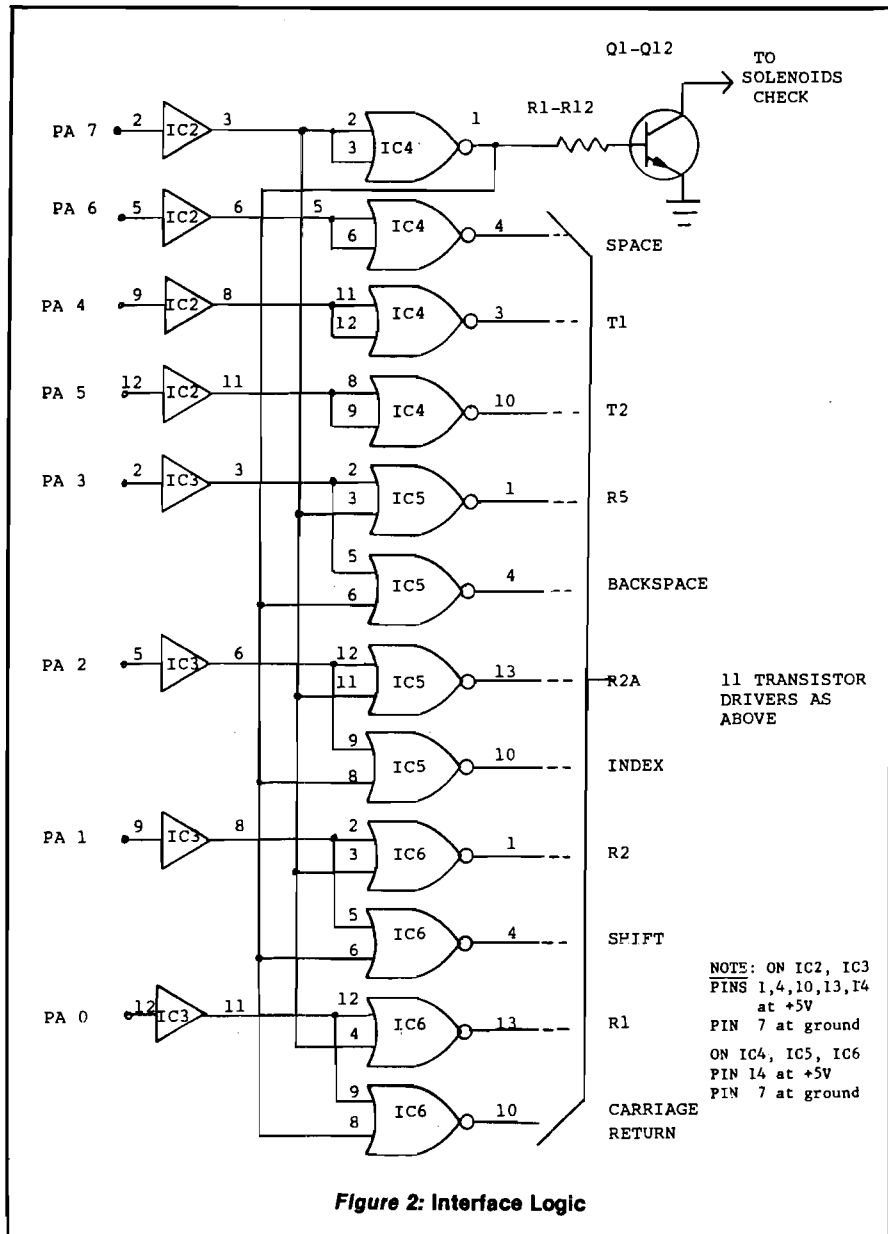
Table 1: Code Selection Chart

T2	0	0	1	1	C	R	R	R	R
T1	0	1	0	1	k	5	2	2	1
									A

U	#	,	\$.	0	1	1	1	1
N	9	z	r	i	0	1	1	1	0
S	6	w	o	f	0	1	0	1	1
H	4	u	m	d	0	1	0	1	0
I	2	s	k	b	0	1	0	0	1
F	0	@	.	&	0	1	0	0	0
T	8	y	q	h	0	0	1	1	0
E	7	x	p	g	0	0	0	1	1
D	5	v	n	e	0	0	0	1	0
	3	t	l	c	0	0	0	0	1
	1	/	j	a	0	0	0	0	0

S	"		!	^	0	1	1	1	1
H	{	Z	R	I	0	1	1	1	0
I	'	W	O	F	0	1	0	1	1
F	:	U	M	D	0	1	0	1	0
T	<	S	K	B	0	1	0	0	1
E	}	_	+	0	1	0	0	0	0
D	*	Y	Q	H	0	0	1	1	0
	>	X	P	G	0	0	0	1	1
	%	V	N	E	0	0	0	1	0
	;	T	L	C	0	0	0	0	1
	=	?	J	A	0	0	0	0	0

Note: The codes above must be sent to the interface for proper operation. To generate a space the NO-PRINT and CHECK solenoids must be picked.



Listing 1 presents the BASIC program required to read text from tape and type it on paper. The program is well documented and needs little explanation. Some items should be noted. First, the shift solenoids are latched; that is, once picked, the typewriter remains in that case until the solenoid is picked again. The program keeps track of case and appropriately picks the shift solenoid. The program assumes that the typewriter is in lower case when it is turned on. There is no guarantee of this. Consequently, the program asks you to check for case at the beginning of every page typed. This is simply done by manually typing a character. Secondly, the code used in this program is for the BCD type ball only. The code may be changed to accommodate other type elements, but the typewriter will no longer function in the manual mode. To determine the proper code, trial and error methods may be the best. Throughout the program a series of delays are introduced. These are required to give adequate time for solenoid activation. These may have to be adjusted for a given terminal.

Finally, a word about the look-up table. The code for shifted and unshifted characters is the same—the position of the shift solenoid determines case. Consequently, when the look-up table was developed, 64 was subtracted from the code of the shifted characters. This permits easy identification of shifted characters. Before the code is sent to the interface, 64 is added back.

The PRINT program assumes that data is written on a file. The text is then retrieved, one character at a time, and printed. Thus, a program is required to create the text. Listing 2 presents a simple approach to word processing. Features include update capability. The program requires the use of two cassette drives, but this can be changed if two cassettes are not available. The program is well documented. The following symbols/codes are used:

shifted &	end of text
←	backspace
shifted \$	underline
shifted "	can be used in lieu of space
cursor down	index
cursor left	deletes these characters
delete	deletes that line
return	carriage return/end of line

During Update only:

return	line of text OK
A	permits insertion of additional text
I	retains previous line permits insertion of additional text
shifted {	deletes previous line end of insertion
delete	deletes that line
home	deletes displayed line—a new line must be entered

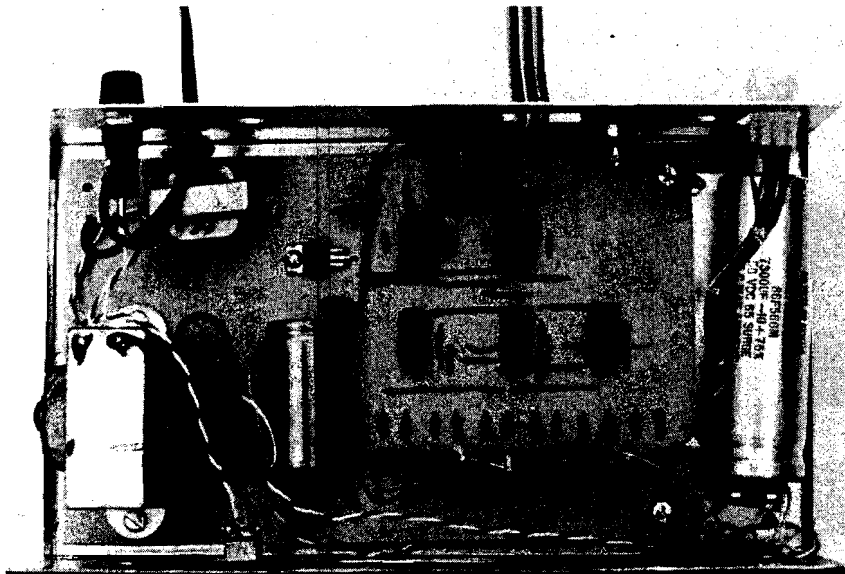
Both the CREATE and PRINT programs are slow because of the many REM statements. To speed up program execution I recommend removal of REM statements prior to use.

Using the Selectric to check your draft is slow. Listing 3 presents a short program which provides a quick listing of the text on my AXIOM printer. The program will only work with an AXIOM printer and must be modified for use on other high speed printers.

In conclusion, the word processing capability is limited, but it is low cost. Additional features can readily be added to the CREATE program, but, for the average user, like myself, the limited capability provided in this article is all that is really necessary.

Ed. Note: To convert these programs to other machines, the following information will be useful. Reverse field characters perform cursor control functions, such as clear screen, cursor right, and cursor home. Decimal address 59459 is the data direction register for the PET's parallel port and 59471 is the write address for the port.

Figure 4



Parts List

- T1 6.3 volt Transformer
- T2 27 volt Transformer
- IC1 7805 5 volt Voltage Regulator
- IC2, IC3 74LS126 Quad Buffers
- IC4-IC6 74LS02 Quad NOR Gates
- R1-R12 1K ¼ watt Resistor
- Q1-Q12 2N3904
- B1 50 PIV 1 amp Bridge
- B2 100 PIV 1 amp Bridge
- C1 5000 uf 12 volt Capacitor
- C2 .22 uf Capacitor
- C3 .1 uf Capacitor
- C4 7500 uf 50 volt Capacitor
- MISC Chassis, Wire, Sockets, etc.

```

10 REM *** SELECTRIC PRINT ROUTINE ***
20 INPUT "IS TYPEWRITER IN LOWER CASE -
-Y OR N";LC$
30 IF LC$="N" THEN GOSUB 840
40 REM IF THE TYPEWRITER IS IN UPPER
CASE THE SHIFT SOLENOID IS PICKED
50 POKE 59468,14
60 REM PLACE PET IN LOWER CASE MODE
70 CODE%=0
80 REM INITIALIZE SHIFT CODE--0=LOWER
CASE
90 INPUT "ENTER FILE NAME";TEXT$
100 PRINT
110 DIM A(220)
120 REM SELECTRIC CODE LOOK-UP TABLE
130 A(13)=254:A(17)=251:A(32)=61:A(33)=
47
140 A(34)= 15:A(35)= 79:A(36)=111
150 A(37)= 2:A(38)=120:A(39)= 11
160 A(40)= 14:A(41)= 8:A(42)= 6
170 A(43)= 56:A(44)= 95:A(45)=104
180 A(46)=127:A(47)= 80:A(48)= 72
190 A(49)= 64:A(50)= 73:A(51)= 65
200 A(52)= 74:A(53)= 66:A(54)= 75
210 A(55)= 67:A(56)= 70:A(57)= 78
220 A(58)= 10:A(59)= 1:A(60)= 9
230 A(61)= 0
240 A(62)= 3:A(63)= 16:A(64)= 80
250 A(65)= 48:A(66)= 57:A(67)= 49
260 A(68)= 58:A(69)= 50:A(70)= 59
270 A(71)= 51:A(72)= 54:A(73)= 62
280 A(74)= 32:A(75)= 41:A(76)= 33
290 A(77)= 42:A(78)= 34:A(79)= 43
300 A(80)= 35:A(81)= 38:A(82)= 46
310 A(83)= 25:A(84)= 17:A(85)= 26
320 A(86)= 18:A(87)= 27:A(88)= 19
330 A(89)= 22:A(90)= 30:A(91)= 4
340 A(163)= 40:A(93)= 63:A(94)= 31
350 A(95)=247:A(193)=112:A(194)=121
360 A(195)=113:A(196)=122:A(197)=114
370 A(198)=123:A(199)=115:A(200)=118
380 A(201)=126:A(202)= 96:A(203)=105
390 A(204)= 97:A(205)=106:A(206)= 98
400 A(207)=107:A(208)= 99:A(209)=102
410 A(210)=110:A(211)= 89:A(212)= 81
420 A(213)= 90:A(214)= 82:A(215)= 91
430 A(216)= 83:A(217)= 86:A(218)= 94
440 A(123)= 88:A(164)=40
450 REM PROGRAM OUTPUT PORT FOR WRITE
460 POKE 59459,255
470 POKE 59471,255
480 REM OPEN FILE WHICH CONTAINS TEXT
490 OPEN 1,1,0,TEXT$
500 PRINT "INSERT PAPER AND PRESS RETUR
N WHEN READY"

```

```

510 GET D$:IF D$="" THEN 510
520 GET#1,A$
530 PRINTA$;
540 REM CHECK FOR END OF FILE
550 IF ST>0 THEN 770
560 REM IF CHARACTER IS RETURN THEN
ACTIVATE CARRIAGE RETURN
570 IF ASC(A$)=13 THEN POKE 59471,254:P
OKE 59471,255:GOSUB810:GOTO520
580 REM IF CHARACTER IS A SPACE THEN
PRINT THE SPACE. THIS IS DONE UNIQUELY
590 REM BECAUSE A(ASC(A$)) IS LESS THEN
64 BUT IS NOT A SHIFTED CHARACTER
600 IF ASC(A$)=32 THEN 690
610 REM THE NEXT FEW LINES OF CODE
CHECK FOR UPPER/LOWER CASE AND SEND
620 REM APPROPRIATE CODE TO USER PORT
630 IF A(ASC(A$))<64 THEN 660
640 IF CODE%=1 THEN CODE%=0:GOSUB 840
650 GOTO 680
660 IF CODE%=1 THEN 680
670 CODE%=1:GOSUB 840
680 IF CODE%=1 THEN POKE 59471,A(ASC(A$
))+64:GOTO 710
690 POKE 59471,A(ASC(A$))
700 FOR I=1 TO 2:NEXT I
710 POKE 59471,255
720 REM DELAY TO PERMIT SOLENOIDS AND
PRINT MECHANISM TO REACT
730 FOR I=1 TO 8:NEXT I
740 REM CHECK FOR END OF PAGE
750 IF A$="␣" THEN POKE 59471,255:GOTO
780
760 GOTO 520
770 POKE 59471,255
780 CLOSE 1
790 END
800 REM DELAY NEEDED FOR CARRIAGE
RETURN
810 FOR I=1 TO 500:NEXT I
820 RETURN
830 REM SHIFT
840 POKE 59471,253:FOR I=1 TO 5 :NEXT I:
POKE 59471,255:FOR I=1 TO 10:NEXT I
850 RETURN

```

```

10 REM ***CREATE TEXT***
20 DIM TEXT$(60)
30 INPUT "ENTER LENGTH OF LINE";LN
40 INPUT "ENTER NUMBER OF LINES PER PAGE
";PL
50 X=0:X1=1

```

```

55 REM PLACE PET IN LOWER CASE MODE
60 POKE 59468,14
70 INPUT"ENTER FILE NAME";FI$
80 INPUT"UPDATE Y OR N";UD$
100 PRINT"␣"
105 REM OPEN CASSETTE 2 FOR WRITE
110 OPEN 2,2,1,FI$
130 IF UD$="N" THEN 310
135 INPUT"INSERT ADDITIONAL TEXT BEFORE
MAIN TEXT--IF YES ENTER A";U$
136 REM OPEN CASSETTE 1 FOR READ
140 OPEN 1,1,0,FI$
145 IF U$="A" THEN 310
150 PRINT"␣"
155 REM GET ONE LINE OF TEXT ONE
CHARACTER AT A TIME
160 B$=""
170 GET#1,A$
180 IF ASC(A$)=13 THEN TEXT$(X1)=B$:GOT
0210
190 B$=B$+A$
200 GOTO170
210 IF ST=64 AND UD$="Y" THEN X=1:GOTO
310
215 REM PRINT LINE OF TEXT ON SCREEN
220 PRINT TEXT$(X1)
225 REM NEXT LINES DETERMINE WHAT IS TO
BE DONE WITH LINE OF TEXT
230 GET U$:IF U$=""THEN 230
240 IF ASC(U$)=13 THEN X1=X1+1:GOTO 160
250 IF ASC(U$)=20 THEN 160
260 IF ASC(U$)=19 THEN 320
270 IF U$="A" THEN X1=X1+1:GOTO 320
280 IF U$="I" THEN 320
290 PRINT"WRONG CODE--REENTER":GOTO 230
300 GOTO160
310 PRINT"ENTER TEXT"
320 TEXT$(X1)="
325 REM SET RIGHT HAND MARGIN
330 FOR I=1 TO LN:PRINT"␣";:NEXTI:PRINT
"␣";:FOR I=1 TO (LN+1):PRINT"␣";:NEXTI
335 REM GET CHARACTERS FROM KEYBOARD
USE "␣" FOR END OF TEXT
340 GET A$:IF A$="" THEN 340
345 REM INDICATION FOR END OF INSERTION
350 IF A$="␣" THEN 160
360 IF A$="␣" THEN 490
370 IF ASC(A$)=20 THEN PRINTCHR$(13):GO
TO 320
375 REM SUBROUTINE 540 MAKES PET KEYBOA
RD LOOK LIKE A TYPEWRITER KEYBOAR
D
380 GOSUB 540
390 PRINTA$;
400 IF ASC(A$)=13 THEN 450

```

```

405 REM IF CURSOR LEFT THEN PREVIOUS
CHARACTER(S) IS DELETED
410 IF ASC(A$)=157 THEN TEXT$(X1)=LEFT$
(TEXT$(X1),LEN(TEXT$(X1))-1):GOTO 340
420 IF ASC(A$)=162 THEN A$=CHR$(32)
425 REM CREATE A LINE OF TEXT BY CON-
CATENATING INDIVIDUAL LETTERS
430 TEXT$(X1)=TEXT$(X1)+A$
440 GOTO 340
445 REM END OF TEXT--SAVES DATA
450 X1=X1+1:IF X1=PL+1 THEN PRINT"*****
*****":GOTO 490
460 IF UD$="Y" AND (U$="A" OR U$="I") T
HEN 320
470 IF UD$="Y" AND X=0 THEN 160
480 GOTO 320
490 FOR I=1 TO X1
500 PRINT#2,TEXT$(I)
510 NEXT I
520 CLOSE1:CLOSE2
530 END
540 IF ASC(A$)>64 AND ASC(A$)<91 THEN A
$=CHR$(ASC(A$)+128):RETURN
550 IF ASC(A$)>192 AND ASC(A$)<219 THEN
A$=CHR$(ASC(A$)-128)
560 RETURN

```

```

10 REM *** PRINT ROUTINE FOR AXIOM 801P
PRINTER ***
20 PRINT"␣"
30 DIM B$(100)
40 INPUT"FILE NAME";FILE$
50 POKE 59468,14
60 I=1
70 OPEN 1,1,0,FILE$
80 B$=""
90 GET#1,A$
100 PRINTA$;
110 IF ST>0 THEN 180
120 IF ASC(A$)=13 THEN 150
130 B$=B$+A$
140 GOTO 90
150 B$(I)=B$
160 I=I+1
170 GOTO80
180 CLOSE1
190 OPEN 4,4:CMD4:PRINTCHR$(8):PRINTCHR
$(11):PRINTCHR$(14)
200 FOR A=1 TO I
210 PRINT B$(A)
220 NEXT A
230 PRINT#4:CLOSE4
240 END

```

MORE SOFTWARE TOOLS FROM HES FOR YOUR 8K PET™

by Jay Balakrishnan



HESEDIT: change 22 lines of data by merely over-typing and insert, delete, and even duplicate lines—all at once! Scroll forwards or backwards by any amount — it's also easy to edit files bigger than your memory. Why code a program to maintain each file? Use HESEDIT for mailing lists, notes or prepare assembler source for HESBAL. All keys repeat. FAST - written in BASIC and assembler. ONLY \$12.95

6502 ASSEMBLER PACKAGE: HESBAL, a full-featured assembler with over 1200 bytes free (8K) & HESEDIT; for less than \$25! HESBAL is THE best 8K assembler available: it uses only 1 tape or disk, yet includes variable symbol sizes, pseudo-opcodes, over 25 error messages and more than 70 pages of documentation. ONLY \$23.95

HESLISTER: formats multi-statement line BASIC programs, shows logic structure (disk reqd.) \$9.95

GUARANTEED to load or replaced FREE
Order from your dealer or direct from us
Plus \$1.50 Postage (our doc. is heavy!)
Disk - Add \$3 . Calif Res. - 6% Sales Tax

HES

Human Engineered Software
3748 Inglewood Blvd. Room 11
Los Angeles, California 90066

24 HOURS — (213) 398-7259



Dealer inquiries welcomed



K I M A I M S Y M T I M

END FRUSTRATION!!

FROM CASSETTE FAILURES
PERRY PERIPHERALS HAS
THE HDE SOLUTION
OMNIDISK SYSTEMS (5" and 8")

ACCLAIMED HDE SOFTWARE

- Assembler, Dynamic Debugging Tool, Text Output Processor, Comprehensive Memory Test
- HDE DISK BASIC NOW AVAILABLE
PERRY PERIPHERALS S-100 PACKAGE

Adds Omidisk (5") to
Your KIM/S-100 System

- Construction Manual—No Parts
- FODS & TED Diskette
- \$20. +\$2. postage & handling. (NY residents add 7% tax) (specify for 1 or 2 drive system)

Place your order with:
PERRY PERIPHERALS
P.O. Box 924
Miller Place, N.Y. 11764
(516) 744-6462

Your Full-Line HDE Distributor/Exporter

NIKROM TECHNICAL PRODUCTS PRESENTS A DIAGNOSTIC PACKAGE FOR THE APPLE II AND APPLE II+ COMPUTER.

"THE BRAIN SURGEON"

All major computer systems are checked for functional hardware analysis on a regular basis for logical as well as some practical reasons. Finding what is exactly wrong can account for most of the money consuming down-time.

Apple Computer Co. has provided you with the best equipment available to date. The Diagnostic's Package was designed to check every major area of your computer, detect errors, and report any malfunctions. *The Brain Surgeon* will put your system through exhaustive, thorough procedures, testing and reporting all findings.

The Tests Include:

- MOTHERBOARD ROM TEST FOR BOTH APPLE II AND APPLE II+
- APPLESOFT CARD TEST • INTEGER CARD TEST • MEMORY RAM TEST
- DISK DRIVE ANALYSIS • MONITOR ALIGNMENT
- DC HAYES MICRODODEM II TEST

System Diagnosis is an invaluable aid to your program library even if your system is working fine. Hours have been wasted trying to track down a "program bug" when actually hardware could be the blame!

The Brain Surgeon allows you to be confident of your system. This can be critical when file handling, sorts or backups are involved. You *must* depend on your computer during all these critical times. Running *The Brain Surgeon* prior to these important functions helps to insure that your system is operating at peak performance.

The Brain Surgeon is easy to use and supplied on diskette with complete documentation.

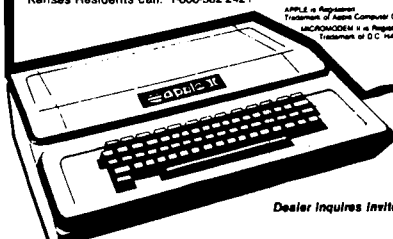
PRICE: \$45.00
REQUIRES: 48K
APPLESOFT in ROM, 1 Disk Drive
DOS 3.2 or 3.3



© Nikrom Technical Products
25 PROSPECT STREET • LEOMINSTER, MA 01453

Order Toll-Free Anytime
Master Charge & VISA users call: 1-800-835-2246
Kansas Residents call: 1-800-362-2421

APPLE is a Registered
Trademark of Apple Computer Co.
MICRODODEM is a Registered
Trademark of DC HAYES



Dealer Inquiries Invited

APPLE & PET

MAE

The Most Powerful Disk-Based
Macro Assembler/Text Editor
Available at ANY Price

Now includes the Simplified Text Processor (STP)

For 32K PET, disk 3.0 or 4.0 ROMS or 8032 (specify) — OR — 48K APPLE II or APPLE II+ and DISK II

MAE FEATURES

- Control Files for Assembling Multiple named source files from disk
- Sorted Symbol table — Up to 31 chars./label
- 27 Commands, 26 Pseudo-ops, 39 Error Codes
- Macros, Conditional Assembly, and a new feature we developed called Interactive Assembly
- Relocatable Object Code
- String search and replace, move, copy, automatic line numbering, etc.

STP FEATURES

- 17 text processing macros
- Right and left justification
- Variable page lengths and widths
- Document size limited only by disk capacity
- Software lower case provision for APPLE II without lower case modification

ALSO INCLUDED

- Relocating Loader
- Sweet 16 macro library for APPLE and PET
- Machine Language macro library
- Sample files for Assembly and text processing
- Separate manuals for both APPLE and PET

PRICE

- MAE, STP, Relocating Loader, Library files, 50 page manual, diskette — \$169.95

SEND FOR FREE DETAILED SPEC SHEET

EASTERN HOUSE SOFTWARE
3239 LINDA DRIVE
WINSTON-SALEM, N. C. 27106

(919) 924-2889

(919) 748-8446

Tiny Pilot Follow Up

MICRO has presented Tiny Pilot for the SYM, KIM and AIM in previous articles.* Here is additional information about "Tiny" and a programming example.

Nicholas J. Vrtis
5863 Pinetree S.E.
Kentwood, Michigan 49508

Here is the sample Tiny Pilot program a number of you have asked for since the original Tiny Pilot in the September 1979 issue of MICRO [16:41]. It is not necessarily an example of "good" Pilot programming (unless you consider anything that works "good"). It was written for two reasons. First, as a simple, practical example of what to do with Pilot. Second, as a demonstration of most of the features of Tiny Pilot. Since it uses most of the statement types and features, it is also a good test program.

The purpose of the demo program is a simple math drill. It asks the user for his name and then proceeds to ask addition problems until he types in QUIT. At that point, it tells him how many answers he got right, and how many were incorrect. Sounds simple enough, doesn't it? It really is! I added some things mainly to demonstrate some statement types. Variables used are:

A — first half of the addition problem
B — second half of the problem
D — a work variable
W — count of wrong answers
R — count of right answers
X — alternates between 0 and 1

Labels used are:

Q — start of the addition question
O — jumped to when answer is correct
D — program wrap-up [done]
N — start of subroutine to get next numbers for next problem
B — jumped to, to add to B instead of A in subroutine

Note that there are **no spaces** in the compute statements. The "?:" statement gets the operator's name so that it can be output in the T: statements with the \$? to personalize the whole thing. Further down in the program, the A: accepts the operator's answer to the addition problem as a character string. The M:QU statement looks for any answer starting with these two letters. This is done to avoid problems with spelling. If you wanted to get really fancy, you could put M:QU,I QU. This would match on either "QUIT" or "I QUIT." If the match statement is true, the program jumps to label D (for done), and wraps up. If this isn't the case, the TP program computes the correct answer and puts it into variable D. The following Match statement compares the value in variable D with the string just entered. Note that leading zeros are ignored from D, but not from the input string. Thus, the answer 02 would not match with the value 2. This is not much of a problem, since very few people put leading zeros in their answers.

The next statement which needs explanation is C:\$ = X. This is another way of matching a variable value. Setting \$ equal to X puts the character string for X into the answer area, so that the M:1 that follows will see if X was equal to 1. The purpose of X in the program is to add variety to the process. If it is equal to 1, the subroutine N adds 2 to variable A and sets X=0 for the next time. When X is equal to 0, B gets 1 added to it. Also, if X is equal to 1 and the answer is correct, the program types out "VERY GOOD!"

Finally, here are a couple of comments about the published version of the program. MICRO did a very faithful job of reassembling my source. The only problem I have heard about is that the at sign (@) did not print in the comments. This is the character used to start the execution of the Tiny Pilot program. More than one person has gotten a little confused about this. The 16-bit checksum for the program is \$6278. This was found by keying in the published code, and it agrees with my

* "Tiny Pilot: An Educational Language for the 6502" by Nick Vrtis [16:41].

"Tiny Pilot for KIM" by Bob Applegate [21:41].

"Tiny Pilot for the AIM" by Larry Kollar and Carl Gutekunst [28:59].

"Tiny Pilot Complementary [Co-Pilot]" by Robert Schultz [29:32].

version. I know of one bug in the version published; it has to do with entering a line longer than 126 characters. The comments say you can go up to 127, but don't believe everything you read. The problem is that the end-of-line character never gets put into the Tiny Pilot program. This, in turn, eventually causes the subroutine FWD1 to branch to SETBGN, which, in turn, resets CURAD back to the beginning of the Tiny Pilot address space. The easiest solution is to limit your lines to less than 126 characters. If you want to patch and/or re-assemble, the solution is to add a BNE \$243 at location \$24D. This will force an end-of-line to be inserted into location 127 and should keep everybody happy. Note that I have not bothered to try this. My CRT is only 80 characters wide, so I never run into the problem. (The only other problem I have heard about is that people with older KIMs don't have the rotate instructions.)

Remember that after the S: statement is entered, you end up back in the editor, with the current address pointing to the beginning of the Tiny Pilot program, so anything you type in will overlay the program. There is no easy way to find the end of your program in order to save it on tape. You must display the whole program, stop the program and look at CURAD.

R:TINY PILOT MATH DRILL PROGRAM
 R:CHANGE THE FOLLOWING TO CHANGE THE SERIES
 C:A = 5
 C:B = A + 3
 T:HI THERE, PLEASE ENTER YOUR NAME
 ?:
 T:WELCOME TO THE MATH DRILL \$?, I HOPE YOU DO WELL
 T:WHEN YOU HAVE HAD ENOUGH, ENTER QUIT INSTEAD OF THE
 T:ANSWER, AND I WILL TELL YOU YOUR SCORE.
 R:HERE IS THE START OF EACH QUESTION
 *QT:

T:HOW MUCH IS \$A + \$B
 A:
 M:QU
 YJ:D
 C:D = A + B
 M:\$D
 YJ:0
 R:HERE THE ANSWER IS WRONG

T:I AM SORRY, THE ANSWER IS \$D
 C:W = W + 1
 U:N
 J:Q
 R:HERE, THE ANSWER IS CORRECT
 *OT:THAT IS CORRECT \$?
 C:\$ = X
 M:1
 YT:VERY GOOD !

C:R = R + 1
 U:N
 J:Q
 *DR:HE ASKED TO QUIT, TELL THE SCORE

T:
 C:D = R + W
 T:I ASKED YOU A TOTAL OF \$D QUESTIONS
 T:YOU ANSWERED \$R CORRECTLY, AND \$W INCORRECTLY.

T:I HOPE YOU ENJOYED YOURSELF \$?, I SURE DID. THANK YOU.

S:
 R:SUBROUTINE TO GET THE NEXT SET OF NUMBERS
 *NC:\$ = X
 M:1
 R:X GIVES VARIETY BY ALTERNATING WHICH GETS ADDED TO

YJ:B
 C:A = A + 2
 C:X = 1
 E:
 *BC:B = B + 1
 C:X = 0
 E:

Microbes

Mike Rowe
 Microbes
 P.O. Box 6502
 Chelmsford, MA 01824

Len Green of Haifa, Israel informed us of some one-byte errors.

In SYM Bridge Trainer (32:44) location 02FB must be changed to C9ED CMP#\$ED or the program will halt after every North bid, including "Pass."

In SYM-ple Sym-on (34:18) location 02AF should be A200 PLAYON LDX#\$00, otherwise the program goes bananas every time you run it.

David Lubar, of Edison, New Jersey, spotted this microbe in his article "UnwzApple" (34:11):

At the end of the listing, in the section following the comment ;CALL FROM BASIC GOES HERE, only half the output hook is established. The lines LDA #START, STA CSWL, should be followed by LDA /START, STA CSWH.

MUSICAL COMPUTER I AND II

Learn How to Read Music!

Written by an M.A. educator with over 20 years of music experience. This two-program cassette provides an *alternative* to music education.

- Treble & Bass Note Reading
- Telling Time
- Notes and Rests
- Sharps and Flats
- Signs and Symbols
- Tempo Definitions
- **CHALLENGING** Practice Testing Opportunities

\$34.95 + \$1 for postage and handling
 Check or Money Order Please
 (MI residents add 4% sales tax)

SPECIAL one-time introductory sample price available to dealers only. Please request on your dealer letterhead.

Check: Apple II 32K
 TRS-80 Level II 16K
 ATARI 800 32K

Name:

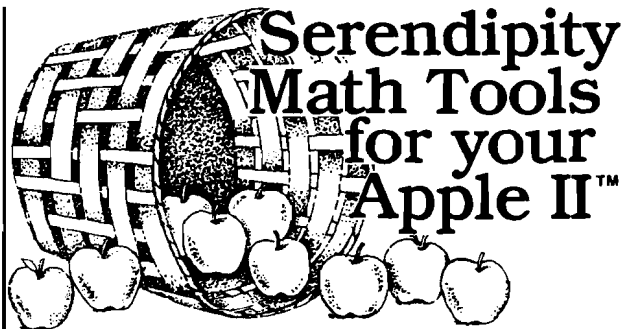
Address:

.....

COMPUTER APPLICATIONS TOMORROW

P.O. Box 605
 Birmingham
 MI 48012





INTER-STAT™ offers you a full range of interactive statistical analysis techniques, from averages and medians to binomial and poisson distributions, correlation coefficients and one- and two-way analysis of variance. \$169.

ADVANCED MATH ROUTINES is the mathematical tool kit for common, yet complex numerical problems. Routines include: linear regression, matrix operations, numerical calculus, differential equations and data set recall for iterative calculations. \$169.

Thoroughly tested, well documented and easy to master, each package includes a 30+ page self-teaching manual. Serendipity's complete line of software solutions for business, education and professional applications are available at your local Computerland or Apple dealer.

For a free brochure, or to order direct contact Serendipity Systems, 225 Elmira Road, Ithaca, NY 14850. Phone 607-277-4889. Visa and MC accepted.

*Apple Computer

SERENDIPITY SYSTEMS

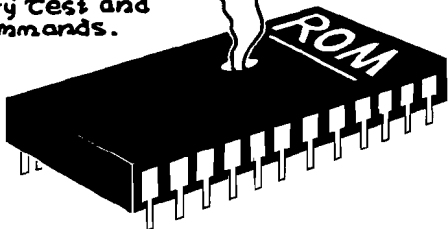
"The ROM Rabbit"

"Your Wish... ..Is My Command"
It's all in a 24-pin Rom at EHS!!



**Pet Owners
Granted 3 Wishes:**

1. Faster Cassette Load
2. Auto-repeat on all keys
3. Memory Test and 12 Commands.



ROM and MANUAL — \$49.95
Specify 3.0 or 4.0 Basic (Works with or without toolkit)

EASTERN HOUSE SOFTWARE (919) 924-2889
3239 Linda Drive (919) 748-8446
Winston-Salem, N. C. 27106

Powerful & Efficient Apple Software SDS Guarantees It.

You depend on good software to save you time and to have your computer help you do a job more efficiently. Our software is designed to do just that. We are one of the oldest companies supplying software for the Apple II™, and one of the very few that offers an unconditional guarantee of satisfaction or your money back! Here are a few that you'll want to add to your library:

Super Terminal Software

ASCII EXPRESS II, by Bill Blue: The most complete communications package available for the Apple II. Designed for the most efficient transfer of data to or from practically any online computer. Fully supports upper/lower case, including characters normally unavailable: underscore, rubout, break, and most others. Keyboard macros allow you to define dual keystrokes as entire strings for fast sign-ons, sign-offs, and system commands. A 20K data buffer allows for large files, and a convenient line editor means easy editing before and after transfer. Buffer can be output to printer, disk, or viewed at any time. Supports Micromodem II™ and most other communication devices.

Price: \$64.95 on Disk.

And for the Z80 Apple...

Z-TERM, by Bill Blue: A flexible communications package for the Apple II equipped with Z80 Softcard™ and the CP/M™ environment. Allows file transfers to or from all types of dial-in systems. Fully supports Micromodem II and most other communication devices, as well as 80 column display boards and external terminals! Utilizes standard CP/M sequential text files, with up to a 40K internal buffer (using additional RAM or Language Card.) Supports multiple modes of data transfer and includes keyboard macros, autodial (with Micromodem II), and upper/lower case.

Price \$99.95 on 16 sector diskette.

Also available...

APPLE-DOC, by Roger Wagner: A set of several utilities to speed up software development and customization. **Vardoc** makes a list of all the variables in a program and every line on which they occur. Also allows you to create a list of descriptors of what each one does. **Linedoc** makes a similar list for each line/subroutine called by a GOTO, GOSUB, etc. **Condoc** is similar but documents all numeric constants — great for scientific & business uses! **Replace** is a powerful replacement editor which makes changing any occurrence of a variable or group of statements a breeze!

Price \$34.95, Disk.

THE CORRESPONDENT, by Roger Wagner: An extremely versatile program! Designed primarily for writing letters and other documents in a very visual way. The Apple screen acts as a "window" onto a 40-80 column page. 4-directional scrolling lets you see any part of the page just as it will be printed. Editor functions include full upper/lower case & control chars., block move/copy, split screen option, even math functions! Additional utilities & uses include printing form letters, a free-form database, putting bi-directional scrolling in your own programs, single-disk copy program, DOS remove for greater storage on diskettes, and more!

Price: \$44.95 on Disk.

*Apple II is a registered trademark of Apple Computer Co.
*Micromodem II is a registered trademark of Hayes Microcomputer Products, Inc.
*Z80 Softcard is a registered trademark of Microsoft Consumer Products, Inc.
*CP/M is a registered trademark of Digital Research, Inc.

All programs require 48K and Applesoft in ROM or language card. Specify DOS 3.2 or 3.3. California residents add 6% to all prices.

See these and other S.D.S. products at your local dealer, or for more information, write or call:

SDS

southwestern data systems
P.O. Box 582-C2 • Santee, CA 92071 • (714) 562-3670

THE PERFORMANCE SLICE

SBCS

GENERAL LEDGER

This package features 31 character account names, 6 digit account numbers, and 10 levels of subtotals for more detailed income statements and balance sheets. Up to 2000 entries can be processed per session.

ACCOUNTS RECEIVABLE

This package allows entry of invoices at any time, credit and debit memos, full or partial invoice payment, invoice aging, and printing of statements. Amounts billed this year and year previous, total payments, and progress billing information are maintained.

ACCOUNTS PAYABLE COMING SOON!

★ Complete your accounting system with the soon to be released A/P package, featuring automatic application of credit and debit memos, open or closed item listing, full invoice aging, and multiple reports that provide a complete transaction review.

★ Your bookkeeping doesn't have to be a bulky, complicated process. The SBCS Accounting System is designed for flexibility and high performance with a cost effectiveness sure to benefit your business!

YOU NEED EXPERIENCE WORKING FOR YOU

★ Packages available at your local Apple dealer.

SMALL BUSINESS COMPUTER SYSTEMS

4140 Greenwood, Lincoln, NE 68504 (402) 467-1878

DISK 1

A fantastic new game disk for OSI-C4P's running with DOS 3.2. Most of the games on this disk are single player such as a multiple level Othello game, Cavern Chase, and more. Other games are designed for single or multiple players such as Tank, and Crystalize. Over seven games and variations in all. [Note that many of these games are in machine code for enhanced play-ability and speed.]

To order send \$26 to:

Simulations
Programming

Rt. 2 Box 98
Burton, WA
98013

Singing the file transfer blues? Then...

Get B.I.T.S.!

Use your Micromodem II,¹ A IO² Card, or Apple Comm Card³ to:

Send data files, BASIC programs, even machine code

to most computers over phone lines.

Copy anything you see

into a 31K buffer then save it on disk and/or print it under your complete control.

Many more features!

See it at your favorite computer store today.

Trademarks held by:

1 - Hayes Microcomputer Products Inc.

2 - SSM

3 - Apple Computer Inc.

B.I.T.S. is a trademark of:

Microsoft Systems

7927 Jones Branch Dr. Suite 400

McLean, Virginia 22102

(703) 385-2944



For your **AIM 65**

MI-JI User applications connector

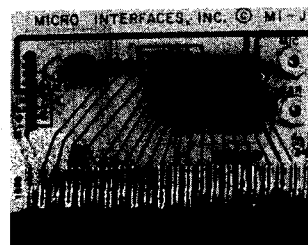
Convenient features

MIC and EAR jacks for cassette recorders

RS232C or 20mAmp serial ports

16 I/O lines + 4 control lines -- VIA signals

Solder pads for remote lines



MICRO INTERFACES, Inc.

P.O. Box 14520

Minneapolis, MN 55414

(612-426-4603)

Assembled, tested 29.95

(90-day warranty)

Bare PC board 13.95

Ask about our Microcomputer DC Control System (MCS) with ROM-based software for real-time control with BASIC

Cursor Control for the C1P

This ½K utility provides the C1P with some new abilities such as editing, user-selectable windows, one-key screen clear, and a cassette "view" mode.

Kerry V. Lourash
1220 North Dennis
Decatur, Illinois 62522

Lack of an editing capability is perhaps the most serious shortcoming of the C1P and Superboard. OSI and Microsoft have provided a video routine ideally suited for a teletype, but lousy for a TV screen. I felt this situation was unbearable and designed my own version of what a video routine should be.

The Cursor Control program replaces OSI's cursor with a dynamic super-cursor that can be moved anywhere on the screen. The view through the TV screen is dramatically improved with the addition of two user-selectable windows and a 'view' mode that lets you look at programs on tape without loading them into memory. If you don't like what you see, a one-key screen clear whisks it away. There's even an edit command for redecorating any line on the screen, and the space-gobbling 'OK' is banished forever.

Cursor Movement Commands

CTRL < Move cursor back one space.
CTRL > Move cursor forward one space.
CTRL U Move cursor up one line.
CTRL D Move cursor down one line.
ESC Move cursor from one window to the other.

Edit Commands

CTRL E Edit. Store character in memory.

SHIFT O Erase last character, move cursor back one space.

Other Commands

RUBOUT Clear window cursor is in.
CTRL V Display contents of tape without loading into memory.

Using the Cursor Control Program

You'll notice two changes to the OSI format immediately. First, the two-line 'OK' message has been replaced by a one-character white block [graphics character 161]. At times, this white block will appear at the end of an error message or other line. If you wish to save screen space, you may start typing without hitting 'RETURN'. The second change is in the cursor. It's now a half-tone block [graphics character 187].

Check the cursor movement commands by pressing the keys for each of the first four commands. Holding the keys down will move the cursor at a constant rate. If you should accidentally move the cursor past the top or bottom of the screen, simply move the cursor in the opposite direction until it reappears or hit the 'ESC' or 'RUBOUT' key.

Now for the edit commands. 'CTRL E' moves the cursor forward like 'CTRL >', but it also enters characters into memory as the cursor passes over them. It's just as if you had typed the character in yourself. To edit a line of BASIC, first list the line. Use the cursor movement commands to put the cursor at the start of the line number. Run the cursor over the line with 'CTRL E' until you reach the part you want to change. You now have four options: to change, delete, or insert characters, or to combine two lines.

To change the line, simply type over the characters you wish to change, 'CTRL E' to the end of the line, and hit 'RETURN'. To delete characters, move the cursor over them with the 'CTRL >' instead of 'CTRL E'. To insert, 'CTRL E' to the point where you want the insertion and use 'CTRL <' to move backward the number of spaces your insertion will occupy. Type your insertion and 'CTRL E' to the end of the line. [I usually 'CTRL <' a little further than I think I have to go, type the insertion, and then 'CTRL >' to the point where I want to use the 'CTRL E'. This saves counting spaces.] Don't worry about the characters you type over when doing an insertion; they're already stored in memory and you're just changing the video display. If it's necessary to combine two lines, use 'CTRL E' to input part or all of the first line, then use the cursor movement keys to move to the second line. 'CTRL E' over what you want in the second line.

A word of caution — you can change a line number by typing a different number before editing the rest of the line. The original line will still be in memory, however, and must be deleted. I usually 'CTRL E' over the original line number and hit 'RETURN'. This deletes the line. Then I go back to the line number, change it, and 'CTRL E' over the rest of the line.

Notice that when you edit lines and hit 'RETURN', the cursor moves to the start of the next line and there is no scroll. To get back 'home', hit the 'ESC' key twice, or the 'RUBOUT' key once. 'ESC' switches windows and homes the cursor (puts it at the start of the bottom line of the window). 'RUBOUT' clears the current window and homes the cursor.

The 'SHIFT O' command erases the letter to the left of the cursor from the screen and from memory, and moves the cursor back one space. Another caution here — if you haven't entered the

character from the keyboard or 'CTRL E', don't try to erase it with 'SHIFT O'.

Finally, there is the 'CTRL V' command. 'CTRL V' lets you see what is on a tape without actually loading it into memory. A tank character [255] is printed to the left of each line to indicate the view mode. You may want to change an address in the view routine [\$D384 in line 89] if the character isn't visible.

Ed. Note: to move the tank one space to the right, change location \$1E9E from 84 to 85 and \$1FAE from 1A to 19. Changing \$1FAE moves the cursor home column to prevent the tank from being printed over the input line.

Exit the view mode by typing a space, just as you would when in the LOAD mode.

Using Windows

Windows are reserved areas of the screen that act like separate, self-contained displays. The Cursor Control program has two scrolling windows, and a third, non-scrolling window for graphics can be created by setting the scrolling windows to occupy less than the whole screen. The screen can be divided horizontally in 1-line increments.

You can set the bottom window to be 4 lines high and do all your immediate mode commands such as PEEK, LIST LOAD, or calculations and then use the large top window to edit BASIC lines. You can have two windows of equal size and run two programs alternately. Directions for a program can be displayed in one window while the program is run in the other, or graphics can be done in the non-scrolling window, and scores or input displayed in the scrolling windows. Unfortunately, I wasn't able to come up with an easy way to set the windows. I was determined to keep the Cursor Control down to 1/2K of memory and it was like trying to close an overstuffed suitcase — some things had to be left out.

Selecting Windows

Ten zero-page locations are used by the Cursor Control to store the current cursor location and the start and end addresses of two scrolling windows (see figure 1). To change the size of the windows, the values stored in these locations must be changed. Look at figure 2. The video display lines are numbered 1 to 32, with hex addresses

FIGURE 1 - ZERO PAGE USE

LOCATION (DECIMAL)	DESCRIPTION	CONTENTS (DEC) (HEX)	
224	CURSOR POSITION	133	\$85
225		211	\$D3
226	START, TOP WINDOW	128	\$80
227		208	\$D0
228	END, TOP WINDOW	128	\$80
229		211	\$D3
230	START, BOT WINDOW	128	\$80
231		208	\$D0
232	END, BOT WINDOW	128	\$80
233		211	\$D3

FIGURE 2 - WINDOW SETTINGS

POKES	LINE	(HEX)	
0, 208	1	\$D000	
32, 208	2	\$D020	
64, 208	3	\$D040	
96, 208	4	\$D060	
128, 208	5	\$D080	(TOP LINE)
160, 208	6	\$D0A0	
192, 208	7	\$D0C0	
224, 208	8	\$D0E0	
0, 209	9	\$D100	
32, 209	10	\$D120	
64, 209	11	\$D140	
96, 209	12	\$D160	
128, 209	13	\$D180	
160, 209	14	\$D1A0	
192, 209	15	\$D1C0	
224, 209	16	\$D1E0	
0, 210	17	\$D200	
32, 210	18	\$D220	
64, 210	19	\$D240	
96, 210	20	\$D260	
128, 210	21	\$D280	
160, 210	22	\$D2A0	
192, 210	23	\$D2C0	
224, 210	24	\$D2E0	
0, 211	25	\$D300	
32, 211	26	\$D320	
64, 211	27	\$D340	
96, 211	28	\$D360	
128, 211	29	\$D380	(BOTTOM)
160, 211	30	\$D3A0	
192, 211	31	\$D3C0	
224, 211	32	\$D3E0	

```

0800      ;CURSOR CONTROL FOR C1P
0800      ;BY LOURASH
0800      ;MICRO #36
1E00      ORG $1E00
1E00      OBJ $0800
1E00      CURSOR EPZ $E0
1E00      START EPZ $E2
1E00      END EPZ $E4
1E00      ;
1E00 2C0302 INPUT BIT $203 ;CHECK LOAD FLAG
1E03 1003 BPL IN
1E05 4CBFFF JMP $FFBF
1E08 8A IN TXA
1E09 48 PHA
1E0A 98 TYA
1E0B 48 PHA
1E0C 2000FD JSR $FD00 ;GET CHARACTER
1E0F      ;
1E0F 4C121E PATCH JMP *+3
1E12      ;
1E12 C9EC BACK CMP #$EC ;CTRL < ?
1E14 D009 BNE UP
1E16 201F1F JSR PRINT
1E19 20B01F JSR REVRSE ;CURSOR - 1
1E1C 4C4D1E JMP F0
1E1F      ;
1E1F C915 UP CMP #$15 ;CTRL U ?
1E21 D010 BNE DOWN
1E23 201F1F JSR PRINT
1E26 A5E0 LDA CURSOR ;CURSOR - 20
1E28 38 SEC
1E29 E920 SBC #$20
1E2B 85E0 STA CURSOR
1E2D B01E BCS F0
1E2F C6E1 DEC CURSOR+1
1E31 D01A BNE F0
1E33      ;
1E33 C904 DOWN CMP #$04 ;CTRL D ?
1E35 D009 BNE FORWD
1E37 201F1F JSR PRINT
1E3A 207D1F JSR FEED ;CURSOR + 20
1E3D 4C4D1E JMP F0
1E40      ;
1E40 C9EE FORWD CMP #$EE ;CTRL > ?
1E42 D011 BNE EDIT
1E44 201F1F JSR PRINT
1E47 E6E0 INC CURSOR ;CURSOR + 1
1E49 D002 BNE F0
1E4B E6E1 INC CURSOR+1
1E4D 20141F F0 JSR PCURSR
1E50 A901 LDA #$01 ;NON-PRINT CHAR
1E52 4CB7FD F1 JMP $FDB7 ;EXIT
1E55      ;
1E55 C905 EDIT CMP #$05 ;CTRL E ?
1E57 D003 BNE ESCAPE
1E59 AD0102 LDA $201 ;CHAR INTO 201
1E5C      ;
1E5C C91B ESCAPE CMP #$1B ;ESC ?
1E5E D020 BNE RUBOUT
1E60 A203 LDX #$03 ;SWITCH WINDOW
1E62 B5E2 ES LDA START,X ;LOCATIONS
1E64 48 PHA
1E65 B5E6 LDA START+4,X
1E67 95E2 STA START,X
1E69 68 PLA
1E6A 95E6 STA START+4,X
1E6C CA DEX
1E6D 10F3 BPL ES
1E6F 201F1F JSR PRINT
1E72 A5E5 HOME LDA END+1 ;HOME CURSOR
1E74 85E1 STA CURSOR+1
1E76 A5E4 LDA END
1E78 20AA1F JSR RETURN+2
1E7B 85E0 STA CURSOR
1E7D 4C4D1E JMP F0
1E80      ;
1E80 C97F RUBOUT CMP #$7F ;RUBOUT ?
1E82 D009 BNE VIEW
1E84 20891F JSR CLEAR ;CLEAR WINDOW
1E87 20731F JSR LINE ;CLEAR HOME LINE
1E8A 4C721E JMP HOME
1E8D      ;
1E8D C916 VIEW CMP #$16 ;CTRL V ?
1E8F D0C1 BNE F1
1E91 208BFF JSR $FF8B ;TURN ON LOAD

```

on the right. Not all of the lines are displayed on the screen because the vertical retrace of the TV blanks out some lines. In the Cursor Control, both windows are initially set to cover the screen from line 5 (\$D080) to line 29 (\$D380).

Let's change the windows so that the bottom window is 4 lines high and the top window covers the rest of the screen. Counting up from the bottom line (29), we find that the boundary between the windows is between line 25 and 26. We set the end of the top window (locations 228, 229) at line 25 and the start of the bottom window (locations 230, 231) at line 26. The two numbers to POKE are listed at the left in figure 2. We type:

```
POKE 228,0:POKE 229,211:
POKE 230,32:POKE 231,211
```

Maybe we would like two lines at the bottom of the screen in order to display scores and have the rest of the screen free for graphics. In this case, the start and end of each window would be the same. The cursor line should be below the bottom of the screen so that we won't waste a line at the bottom. Lines 30-28 for both windows:

```
POKE 226,96:POKE 227,211:
POKE 228,160:POKE 229,211
POKE 230,96:POKE 231,211:
POKE 232,160:POKE 233,211
```

To gain extra lines at the bottom (or top) of the screen, you can change the TABLE in the last line of the program. Also, you can change line length or position of lines on the screen to customize the Cursor Control to your particular TV. The SBC #\$1A instruction in the RETURN subroutine determines the starting point of video lines. The SBC #2 instruction in the LETTER routine controls the end point of the lines. If you change the line length you'll also have to change the SBC instruction in the REVRSE subroutine. If you increase the line length, decrease the SBC instruction by the same amount, and vice versa.

People who have video monitors without retrace blanking can eliminate retrace smear by not setting windows on lines smeared by the retrace. Users with 600 baud conversions might not have to add NULLs when SAVEing if they use a small window (4 lines?) when LOADING. (This speeds up the scroll routine.)

Subroutines

HOME	Changes cursor location to home position and prints cursor.
PCURSR	Saves character at cursor address in location \$201 and prints cursor 'over' the character.
PRINT	Prints contents of \$201 (character 'underneath' the cursor) at cursor location.
LOAD	Initializes RAM locations \$207-\$20D for use in scroll, clear screen routines.
SCROLL	Goes through every byte in window and puts the contents in original location + \$20 (one line above).
LINE	Clears home line.
FEED	Moves cursor location down one line.
CLEAR	Clears window.
RETURN	Puts cursor at start of line.
SETUP	Sets Cursor Control patches, HIMEM, initializes stack.
REVRSE	Moves cursor back one space.

How Cursor Control Works

First, the Cursor Control looks at the LOAD flag and jumps to the LOAD routine if the flag is set. Otherwise, it checks input from the keyboard for commands. The cursor movement commands change the location of the cursor (224, 225 or hex \$E0, E1) and load a non-printing character in the A register. This causes BASIC to ignore the character and loop back to the start of the input routine.

The 'CTRL E' routine puts the character 'underneath' the cursor into the A register, so that it's treated as if it were a character typed from the keyboard.

The 'ESCAPE' routine switches the contents of the window registers and homes the cursor in the window thus selected.

The 'RUBOUT' routine clears the current window and homes the cursor. By the way, if you put the address of the CLEAR subroutine in locations 11, 12 you have a USR(X) screen clear.

The 'VIEW' routine bypasses the routines that store data in memory and prints data from tape on the screen only.

1E94	20BAFF	V1	JSR \$FFBA	; INPUT CHAR
1E97	20A51E		JSR OUTPUT	; PRINT CHAR
1E9A	AD0302		LDA \$203	; 'LOAD' FLAG
1E9D	8D84D3		STA \$D384	; PRINT IT
1EA0	D0F2		BNE V1	; FLAG ON?
1EA2	4C521E		JMP F1	; NO, EXIT.
1EA5				
1EA5			; OUTPUT ROUTINE	
1EA5				
1EA5	8D0202	OUTPUT	STA \$202	; TEMP SAVE CHAR
1EA8	48		PHA	
1EA9	8A		TXA	
1EAA	48		PHA	
1EAB	98		TYA	
1EAC	48		PHA	
1EAD	AD0202		LDA \$202	; LOAD CHAR
1EB0	F056		BEQ EXIT	; IF NULL, EXIT
1EB2				
1EB2	4CB51E		PATCH2 JMP *+3	
1EB5				
1EB5	C90A	LF	CMP #\$0A	; LINE FEED ?
1EB7	F04F		BEQ EXIT	
1EB9				
1EB9	C90D	CR	CMP #\$0D	; 'RETURN' ?
1EBB	D008		BNE ERASE	
1EBD	A920		LDA #\$20	
1EBF	20221F		JSR PRINT+3	
1EC2	4CEF1E		JMP LO	
1EC5				
1EC5	C95F	ERASE	CMP #\$5F	; SHIFT O ?
1EC7	D013		BNE LETTER	
1EC9	C60E		DEC \$0E	; CHARACTER COUNTER
1ECB	A920		LDA #\$20	; ERASE CHARACTER
1ECD	8D0102		STA \$201	; UNDER CURSOR
1ED0	20221F		JSR PRINT+3	; ERASE CURSOR
1ED3	20B01F		JSR REVRSE	
1ED6	201B1F		JSR P1	; PRINT CURSOR
1ED9	4C081F		JMP EXIT	
1EDC				
1EDC	8D0102	LETTER	STA \$201	
1EDF	20221F		JSR PRINT+3	
1EE2	E6E0		INC CURSOR	
1EE4	A5E0		LDA CURSOR	; CURSOR AT
1EE6	091F		ORA #\$1F	; END OF LINE?
1EE8	38		SEC	
1EE9	E902		SBC #\$02	
1EEB	C5E0		CMP CURSOR	
1EED	D016		BNE LE+3	; NO, BRANCH
1EEF	20A81F	LO	JSR RETURN	
1EF2	85E0		STA CURSOR	
1EF4	C5E4		CMP END	; IS CURSOR
1EF6	A5E1		LDA CURSOR+1	; ON HOME LINE?
1EF8	E5E5		SBC END+1	
1EFA	B006		BCS LE	; YES, SCROLL
1EFC	207D1F		JSR FEED	; NO, DOWN ONE LINE
1EFF	4C051F		JMP LE+3	
1F02	20441F	LE	JSR SCROLL	
1F05	20141F		JSR PCURSR	
1F08				
1F08	68	EXIT	PLA	
1F09	A8		TAY	
1F0A	68		PLA	
1F0B	AA		TAX	
1F0C	68		PLA	
1F0D	4C6CFE		JMP \$FF6C	; TO NORMAL OUTPUT
1F10				
1F10			; SUBROUTINES	
1F10				
1F10	A9A1	OK	LDA #\$A1	
1F12	D00E		BNE PRINT+3	
1F14				
1F14	A000	PCURSR	LDY #\$00	
1F16	B1E0		LDA (CURSOR), Y	; SAVE CHAR
1F18	8D0102		STA \$201	; AT CURSOR LOC
1F1B	A9BB	P1	LDA #\$BB	
1F1D	D003		BNE PRINT+3	; PRINT CURSOR
1F1F				
1F1F	AD0102	PRINT	LDA \$201	; GET CHAR
1F22	A000		LDY #\$00	
1F24	91E0		STA (CURSOR), Y	; PRINT IT
1F26	60		RTS	
1F27				
1F27	A9AD	LOAD	LDA #\$AD	; LDA OP CODE
1F29	8D0702		STA \$207	
1F2C	A98D		LDA #\$8D	; STA OP CODE
1F2E	8D0A02		STA \$20A	

```

1F31 A960          LDA #\$60          ;RTS OP CODE
1F33 8D0D02       STA \$20D
1F36 A5E3         LDA START+1
1F38 8D0902       STA \$209
1F3B 8D0C02       STA \$20C
1F3E A5E2         LDA START
1F40 8D0B02       STA \$20B
1F43 60           RTS
1F44
1F44 20271F       ; SCROLL JSR LOAD
1F47 18           CLC
1F48 6920         ADC #\$20         ;START + 20
1F4A 9003         BCC S0
1F4C EE0902       INC \$209
1F4F 8D0802       S0 STA \$208
1F52 A6E4         LDX END
1F54 A4E5         LDY END+1
1F56 200702       S1 JSR \$207       ;SCROLL ONE BYTE
1F59 EE0802       INC \$208
1F5C D003         BNE S2
1F5E EE0902       INC \$209
1F61 EE0B02       S2 INC \$20B
1F64 D003         BNE S3
1F66 EE0C02       INC \$20C
1F69 EC0B02       S3 CPX \$20B       ;LOW BYTE DONE?
1F6C D0E8         BNE S1           ;HIGH BYTE DONE?
1F6E CC0C02       CPY \$20C
1F71 D0E3         BNE S1           ;ERASE HOME LINE
1F73 A020         LINE LDY #\$20       ;BLANK
1F75 A920         LDA #\$20
1F77 91E4         L1 STA (END),Y
1F79 88           DEY
1F7A D0FB        BNE L1
1F7C 60           RTS
1F7D
1F7D A5E0         ; FEED LDA CURSOR   ;CURSOR DOWN
1F7F 18           CLC              ;ONE LINE
1F80 6920         ADC #\$20         ;CURSOR +20
1F82 85E0         STA CURSOR
1F84 9002         BCC FE
1F86 E6E1         INC CURSOR+1
1F88 60           FE RTS
1F89
1F89 20271F       ; CLEAR JSR LOAD     ;CLEAR WINDOW
1F8C A4E4         LDY END
1F8E A6E5         LDX END+1
1F90 A920         LDA #\$20
1F92 200A02       CL JSR \$20A     ;BLANK
1F95 EE0B02       INC \$20B        ;CLEAR ONE BYTE
1F98 D003         BNE C1
1F9A EE0C02       INC \$20C
1F9D CC0B02       C1 CPY \$20B     ;LOW BYTE DONE?
1FA0 D0F0         BNE CL         ;HIGH BYTE DONE?
1FA2 EC0C02       CPX \$20C
1FA5 D0EB        BNE CL
1FA7 60           RTS
1FA8
1FA8 A5E0         ; RETURN LDA CURSOR
1FAA 091F         ORA #\$1F       ;CURSOR TO START OF LINE
1FAC 38           SEC
1FAD E91A         SBC #\$1A
1FAF 60           RTS
1FB0
1FB0 20A81F       ; REVERSE JSR RETURN ;MOVE BACK
1FB3 C5E0         CMP CURSOR     ;ONE SPACE
1FB5 D00B        BNE RE
1FB7 A5E0         LDA CURSOR
1FB9 38           SEC
1FBA E908         SBC #\$08
1FBC 85E0         STA CURSOR
1FBE B002         BCS RE
1FC0 C6E1         DEC CURSOR+1
1FC2 C6E0        RE DEC CURSOR
1FC4 60           RTS
1FC5
1FC5 A209         ; SETUP LDX #\$09 ;INITIALIZE
1FC7 BDF51F       LDA TABLE,X
1FCA 95E0         STA CURSOR,X
1FCC CA          DEX
1FCD 10F8        BPL SETUP+2
1FCF A900         LDA #INPUT
1FD1 8D1802       STA \$218       ;INPUT VECTOR
1FD4 8585         STA \$85        ;HIMEM SET
1FD6 A91E         LDA /INPUT
1FD8 8D1902       STA \$219
1FDB 8586         STA \$86

```

The 'LETTER' routine prints the character that has been input (from keyboard or tape) and increments the cursor location. If the cursor is not at the end of the line the routine prints the cursor at the new location and exits. If the cursor is at the end of the line or 'RETURN' is hit, the cursor is reset to the start of the line. Then, if the cursor is on the home line, a scroll is done and the cursor is printed at home. If the cursor is not on the home line, the cursor is moved down one line and printed.

Loading the Cursor Control

All directions are for an 8K memory. First, enter the Cursor Control into memory using the monitor (if you use the OSI Assembler/Editor you'll have to assemble the input and output routines separately from the subroutines, unless you have more than 8K of memory). After the program is entered, double check to make sure you've done it right.

Now hit 'BREAK C' and set memory size to 7600. This initializes the BASIC and temporarily protects the Cursor Control from being written over. Complete cold start then hit 'BREAK M'. Change location 1 to \$C5 and location 2 to \$1F. This points warm start to the SETUP routine. Hit 'BREAK W'. This initializes the Cursor Control and a white square — the new 'OK' symbol — should appear in the bottom left corner of the screen. Check all commands to make certain everything works. The 'ESC' key will not appear to do anything at this time because the windows are both set to cover the same area.

Assuming all commands work, you are now faced with the problem of taping the program. Die-hard BASIC hackers will want to convert the Cursor Control to DATA statements. This can be done with one of the programs designed for that purpose. After your BASIC program has POKED the Cursor Control into memory, have it POKE 1,197:POKE 2,31. A 'BREAK W' will bring the Cursor Control to life. I prefer the machine language load method because the Cursor Control can be loaded even if a BASIC program is already in memory. Use a routine such as Hoyt's hex dump (*Best of MICRO*, vol. 2, p. 184) to save the Cursor Control in OSI format. After loading the Cursor Control [if you've used Hoyt's program] change location 0 to \$4C, location 1 to \$C5, location 2 to \$1F. Now hit 'BREAK W' to initialize, and you're in business.

The Cursor Control could be put in an EPROM. OSI's 2K monitor ROM uses only three pages, \$FD00-FFFF, plus a short routine at \$FCB1 to support the C1P. The remainder contains a floppy bootstrap and routines for other models of computers. With a 2716 EPROM, you would have over 1K for your own routines.

There is provision in both input and output routines (PATCH , PATCH 2) for JMP XXXX instructions. You can add extra features to the Cursor Control by JMPing to your code, executing it, and then JMPing back into the Cursor Control. Stack initialization in the SETUP routine solves a small but annoying problem of warm start. Now you can do a PEEK or POKE without getting an error message the first time. The time delay for the video routine controlled by location \$206 is not included in the Cursor Control. If you have room in RAM, a short BASIC program could be written to allow easier manipulation of the windows.

```

1FDD A9A5          LDA #OUTPUT
1FDF 8D1A02        STA $21A
1FE2 A91E          LDA /OUTPUT          ;OUTPUT VECTOR
1FE4 8D1B02        STA $21B
1FE7 A910          LDA #OK                ;'OK' MESSAGE
1FE9 8504          STA $04                ;VECTOR
1FEB A91F          LDA /OK
1FED 8505          STA $05
1FEF A2FE          LDX #$FE          ;SET STACK POINTER
1FF1 9A            TXS
1FF2 4C74A2        JMP $A274              ;WARM START
1FF5 85D3          TABLE ADR $D385
1FF7 80D0          ADR $D080
1FF9 80D3          ADR $D380
1FFB 80D0          ADR $D080
1FFD 80D3          ADR $D380

```

I don't have access to a C2P, so I can't be specific, but with a few changes the Cursor Control could run on a C2P. A disk system could use the Cursor Control if zero page location \$E0-E9 were changed to addresses not used by disk BASIC. Also, location \$0E in the ERASE routine may be used differently in disk BASIC. Finally, the Cursor Control is by no means completed. I welcome constructive

criticism or improvements. Please send to me at the address given at the beginning this article.

Kerry Lourash has owned a Superboard II for a year. He is interested in both hardware and software. Deciphering BASIC-in-ROM and designing utilities are his current obsessions. He is a board member of the Macon County Computer Club.

MICRO

(Classified — continued from page 54)

OHIO SCIENTIFIC

S-FORTH — a full implementation of Fig-FORTH including editor, virtual disk sub-system, and compatibility with OS65D-3 on 5¼" or 8" disk. \$34.95.

Source listing \$24.95.
Both for \$49.95.

TOUCH TYPING MADE EASY — 15 lesson set teaches you to "touch type". Now also available for the C1P. 8K. \$19.95.

TITANIC QUEST — a real time search where you risk your remaining supplies to find the Titanic. 8K. \$6.95.

TEXT EDITOR — the best screen text editor available for OSI C4P, C8P disk systems. \$19.95.

Send for our FREE 14 page software and hardware catalog. Includes photos and complete descriptions of all game, utility, and business software.

Aurora Software Associates
P.O. Box 99553
Cleveland, Ohio 44199
(216) 221-6981

TURNKEY MEDICAL BILLING SYSTEM

Interactive data entry. Automated file management. Outputs: Patient statements, Universal Claim Forms, financial reports. Customized by user-developed text files. Requires Apple, Applesoft, printer. One disk drive manages 150 accounts; 2 drives — 400 accounts. \$350 for programs and 25 pp documentation.

Jerome B. Blumenthal, M.D.
7500 E. Hellman
Rosemead, CA 91770

OSI Accounting System

Small business double entry accounting system based on MDMS structure prints Income Statement, Trial Balance, Charts of Accounts, and others. System is configurable by user and requires only 65D and MDMS capability plus minimum of 8K workspace. Program and data disks plus user guide \$100.00. Listings sold separately for \$20.00 each module.

VIDEO VENTURES SOFTWARE
1708 Beechwood
Fullerton, CA 92635

Hockey for the Apple II

A two-player game using Integer BASIC and two machine language routines for smooth graphics. On cassette, \$9.99.

Arlan Henderson
Rt. 2, Box 46
Saltville, Virginia 24370

STORMGATE: C1P Adventure Series in 8K

THAX—Wield mans' science, weapons and mental forces in an ongoing space quest through a million locations of THAX. MINERVA - Your spacecraft is disabled on a barren world. Use your wits and a robot to survive. Cassette, \$6.50 each.

STORMGATE SOFTWARE
SYSTEMS
P.O. Box 801
Kirksville, Missouri 63501

These low cost ads are actually subsidized in part by MICRO allowing you the perfect place to announce special club events, unusual promotions, or even want-ads.

Each classified ad costs only \$10.00 per insertion, pre-paid with type-written copy. Please limit these entries to less than 40 words. (Title line, name and address not considered in count.) Ads should be received before the 20th of the month preceding the month of publication, i.e. May 20th for the July issue.

If you have any further questions, please call (617)256-5512.

Protecting Memory from DOS

A technique is described to create a "Funny DOS" which automatically protects an area of RAM above DOS. Examples are given of the many uses for this protected RAM.

Glenn R. Sogge
Fantasy Research & Development
P.O. Box 203
Evanston, IL 60204

As most users of the Apple DOS are probably aware, versions 3.2 and 3.2.1 come in two different flavors—memory-size independent (a "master") and memory-size dependent (a "slave"). A slave disk is created with the "INIT" command and produces a disk that will always load the DOS into a specific region of memory. This region is defined by where DOS is sitting when the "INIT" command is given. Generally, this region is at the top of available RAM in the machine.

When a master disk is booted on the machine, the DOS gets loaded underneath address \$4000 (16K) and some relocation code gets loaded under the DOS. This relocation routine finds the top of RAM and moves the DOS to sit right underneath it. On a 16K machine DOS stays where it is, on a 32K machine it moves to under \$8000, and on a 48K machine it moves to under \$C000. The DOS then loads and runs the HELLO program.

As you can see, a master disk clobbers a lot of memory, and once it is booted, all memory below the DOS is available for use. The user can protect memory from BASIC by setting

HIMEM and LOMEM, but that can be inconvenient and easy to forget. It would be much nicer if DOS could do it for us automatically. Well, there is a way to do it quite simply.

If you take a look at the relocation code that a master disk loads in at \$1B00, you will see that the routine starting at \$1B03 is a memory sizer. It starts in the highest possible page of RAM (\$BF) and works its way down until RAM is found. This page is the highest page that the DOS can then occupy. After finding this high page of RAM, the code relocates the DOS to sit from there down. As far as DOS is concerned, then, there is no usable RAM memory above itself.

If we trick the relocater into starting its search for room somewhere else, we can have some free memory that we (and our programs) know about but DOS doesn't. We will have essentially protected some memory from encroachment. Fortunately, it only requires changing one byte to accomplish our task. (Thank heavens for simply structured code.) The byte to change is the "\$BF" at \$1B04. If we change it to "\$BE", we'll protect one page of RAM. If we change it to "\$9F", we'll have free use of the RAM from \$A000 to \$BFFF—8K of space. (All the examples assume that you have a 48K machine; the principles are the same for any memory size.)

The way to accomplish this is to change the byte on the master disk. The byte is at track 0, sector A, byte \$04. With the disk modified, whenever it boots it will start its search for RAM with whatever page number you have given it. You now have a relocating DOS that you have some control over. But since it is a master, it still crashes

large portions of lower memory. This presents no problem. With your 'funny DOS' running, just initialize a new disk. Now you have a 'funny DOS' that boots right into the memory range you picked for it, and only crashes pages \$3,\$8, and \$9 (the boot code and 'nibble buffers'). Anything above the DOS location is still there.

You may have a problem if some of your code depends upon DOS being at particular locations rather than utilizing the jump vectors in page 3 or the HI- and LOMEM pointers in page 0. Also, it would probably be a good idea to have your HELLO program print out something like "40K DOS MASTER" or "44K DOS SLAVE" to remind yourself of what is happening.

Why Would You Want to Do This?

This section of 'protected memory' is an ideal place to put printer and peripheral drivers, machine language sorting routines, utility programs, debug packages and the like. It can also be used as a scratchpad memory area that won't get in the way of other conflicting uses. While running in BASIC, for example, it's not always very easy to determine where the program and variable spaces are, but with a section of memory that BASIC doesn't know about, you can be sure that your data won't get clobbered.

The memory area becomes a 'systems memory area' with you, the programmer, as the systems manager—not DOS, not BASIC. Your HELLO program could load a batch of utilities so they would always be there and always be in the same place for use by many different programs—kind of a

'writable ROM' like the language card system, but with only software, not hardware protection.

What Does 'DOESN'T KNOW ABOUT' Mean?

BASIC and DOS don't know about your hidden memory because it is outside of their HI- and LOMEM limits. Therefore, any functions which use those limits, like loading programs, allocating buffers, creating variables in a program, etc., will not even look to your locations. You can, of course, tell them to if you like. Commands like "BLOAD DEBUG,\$BF00" will still work like they always have because you are supplying DOS with the parameters.

Data and programs can be loaded into, and saved from your protected area without any problem. When you give a DOS command with an address, it blindly does what you tell it to, since it assumes that you know best. Thus, you can BLOAD a program to ROM if you want. This kind of relinquishment of parameter checking is what allows us to create and use the hidden memory. Since the BSAVE, BLOAD, and BRUN routines all use the address information stored when the file is saved, DOS just assumes that it is right and goes ahead with the command, without checking it against what it (incorrectly) knows about the machine's memory. In short, just use the memory area like you would any other area, DOS doesn't care.

Updating 'Funny DOS' Disks

Your modified DOS disks will work just fine with the Update 3.2 and 3.2.1 programs. The Update program has to be run with a master disk in the drive because the first thing it does is load a copy of the DOS image with the relocation code into memory. If the DOS image it loads is a normal, 48K relocating DOS, the disk will be updated to a normal master disk. If, however, the Update program is loaded and you then insert a 'funny DOS' master and then run the Update program, it will create a 'funny DOS' master. Only the DOS image on the disk is modified, so all your files will still be the same.

If you have updated a disk to a 'funny DOS' master, it will once again clobber lots of memory when booting, but it will not crash the area above the memory limit you set.

An Application Suggestion

Both BASICs, and usually any large assembly language program, eat up a huge portion of page zero memory. Particularly with Applesoft, you might want to use a good portion of that same area for some machine language utilities to sort strings or to utilize the Sweet-16 interpreter. (Relocating Sweet-16 to run from RAM in your protected memory area would make it easily usable from both BASICs or assembly language.) Rather than try to figure out which locations have to be saved and which can be used freely, why not use a brute force approach?

Upon entering your utility, the first thing that is done is to save all of page zero in some nice, safe place—like above DOS in a hidden memory area. Now your routine can use any or all of page zero as it likes. When the routine is done, it restores the old page zero and returns to the calling routine.

```
ZTOBF PHP
      PHA
      TXA
      PHA ; SAVE THE REGISTERS
      TYA
      PHA
      LDX #$00
ZTB2  LDA $00,X
      STA $BF00,X ; SAVE PAGE ZERO
      INX          ; AT $BF00-BFFF
      BNE ZTB2
      ...
      ...
      (YOUR PROCESSING HAPPENS
      HERE)
      ...
      ...
      BFTOZ LDX #$00
      BFZ2  LDA $BF00,X
      STA $00,X ; RESTORE PAGE
              ZERO
      INX
      BNE BFZ2
      PLA
      TAY
      PLA
```

TAX ; RESTORE THE
REGISTERS

PLA
PLP
RTS

Although this type of routine is not the most efficient (all of page zero might not need to be saved), it does have the programming advantage of simplicity. If you don't have to remember which locations to be careful with, you can get on with the process of writing the code to get the job done. As the length and complexity of your processing routine grows, so, probably, will your need for page zero locations. The above routines solve the problem in one fell swoop. In addition, the time spent in the transfer loops as a percentage of the processing time will decrease as the routine grows in complexity. The programming ease will greatly outweigh the almost negligible time spent saving and restoring the page. This is particularly true if the routine is related to operator I/O.

By saving the registers, in addition to the page zero locations, the processing routine becomes completely transparent to the calling routine. BASIC (or a machine language mainline) can call an extremely complex and powerful routine and not have to know a thing about what it does or how it does it. All the 'variables' (locations) your routine uses are completely local to your routine with no global side effects. (Your routine could alter locations in the saved page to pass back values, though.)

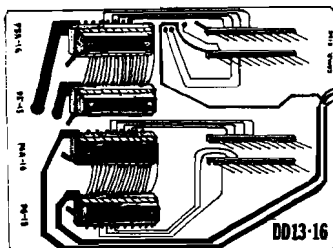
An extension of the procedure would be to swap copies of page zero rather than just saving the 'main' copy. This allows your routines to have their own page zero that is not altered by the calling program. Passing values back and forth gets a little bit tricky then; you might want to dedicate the upper half of page zero to 'private use' and pass values in the lower locations. [This means only saving or swapping half of the page which will also speed up the routines.] Another extension, of course, of this concept is to save the stack page, too. With an interrupt-driven scheduler and multiple-save areas, time sharing and multi-tasking are just around the proverbial corner.

MICRO

NEW

DOUBLE DOS Plus

FOR
**APPLE
COMPUTERS**



\$39

MICRO-WARE

DISTRIBUTING INC.
P. O. Box 112
Pompton Plains, N.J. 07444

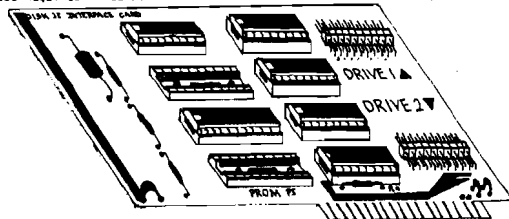
201-839-3478

DEALER INQUIRIES INVITED !!

WHY IS DOUBLE DOS PLUS BETTER

NOTE: APPLE is a registered trademark of APPLE COMPUTER INC., CUPERTINO, California.
*DOUBLE DOS PLUS REQUIRES APPLE DOS ROMS

- 1) Nothing needs to be soldered, just plug and go.
- 2) Since all four ROMs are used, all software will work even early 3.1 DOS.
- 3) Because the ROMS fit on the back of the board, it has the thinnest configuration allowing full use of slot #7.
- 4) One set of ROMS is powered up at time thereby saving power.
- 5) Full 90 day warranty



DOUBLE DOS PLUS - A piggyback board that plugs into the disk controller card so that you can switch select between DOS 3.2 and DOS 3.3. Works with the language system eliminating the need in many cases to boot the Basics disk. Also eliminates the chore of converting all of your 3.2 disks to 3.3.

FROM



FORTH
DISK PROGRAMMING SYSTEM FOR
THE APPLE II COMPUTER...



Fast, inherently structured, programming system ready for your APPLE II or II+ (24K).

Extensive, professional, 100 page bound documentation. Cleanly interfaced to DOS 3.2 or 3.3. Files are completely compatible with DOS or BASIC.

- Control C break and continue for reasonable debugging.
- Built-in, convenient editor.
- FORTH structured assembler.
- The best blend of FORTH and the APPLE's capabilities.
- Supports games, music, I/O, graphics, disk, tape.
- Supplied on APPLE diskette.
- Excellent for applications or systems programming.
- After two years, still **\$49.95**
Calif. residents add \$3.00 sales tax

From your dealer or direct from:
SOFTAPE, Dept. f.
10432 Burbank Blvd.
North Hollywood, CA 91601
or Call: 1-213-985-5763

FOG-279 \$49.95

Master Charge/Visa Accepted.

Apple is a registered trademark of APPLE COMPUTER, INC.

NEW!

FROM **Brøderbund Software**

STRATEGY GAMES!

FAST ACTION GAMES!



**THE SAGA CONTINUES . . .
IV TAWALA'S LAST REDOUBT**

The cruel Emperor Tawala has been forced from his throne on the world of Galactica and has fled for his life to the planet of Farside, where he and a small bank of adherents prepare to make their last stand. Extreme solar conditions have isolated Farside from the rest of the galaxy, and so it remains to Benthil, leader of the local insurrectionists, to press the final assault on Tawala and his minions.

TAWALA'S LAST REDOUBT puts you in the position of rebel leader. You must intercept and decipher Tawala's secret messages to his supporters, form alliances with local chiefs, detect Tawala's spies in your midst, separate hard intelligence from enemy disinformation, avoid Tawala's military forays against you and, finally, lead the assault against the Prince's stronghold.

Minimum Configuration:

- TRS-80 Cassette, 16K, Level II, \$19.95
- TRS-80 Disk, 32K, \$24.95
- APPLE Disk, 48K with APPLESOFT, \$29.95

Apple, Apple II Plus and Applesoft are trademarks of Apple Computer Co. TRS-80 is a trademark of Radio Shack.

APPLE GALAXIAN

Apple Galaxian - In brilliantly colored array, the Galaxians swoop down from all sides in dazzlingly swift attacks to do battle upon the lone defender. This faithful rendition of that most popular of all bar games may drive you around the bend, but think of all the quarters you'll be saving! Apple II Integer or Plus, 48K disk, \$24.95.

How to order: Ask your dealer or send check or money order for the exact retail price to:



Brøderbund Software

Box 3266, Eugene, Oregon 97403

Call (503) 343-9024 to order. NO CHARGE FOR SHIPPING AND HANDLING!
Visa and Mastercard accepted.

We've got more! Send for our free catalog!

User Defined Routines in UCSD Pascal

The UCSD Pascal system has several features which allow a user to create a collection of procedures and functions to be used as subroutines by other host Pascal programs. Such routines can be written using Pascal itself or using the 6502 assembler provided with the UCSD system. These capabilities will be illustrated in two parts. Pascal subroutines are discussed in part one, which constitutes the remainder of this note. Assembler routines are discussed in part two, which will appear next month. It is assumed that the reader is familiar with the UCSD Pascal system, especially the use of the Editor, to create and store program files.

Part One — Pascal Subroutines

A. DRAWCHAR — A Simple Example of a Pascal Routine

The C4P and C8P series of Ohio Scientific computers use a memory mapped video system which supports a character set of 256 graphics characters. The C4P and C8P users' manuals include tables listing the numeric equivalent for each of the graphics characters. The following Pascal procedure displays the graphics character corresponding to a given CHARNUM at the screen location with coordinates (XCOOR,YCOOR) relative to the upper left-hand corner of the screen. (Note: Although the display screen on these systems is nominally 32 rows x 64 columns, the Pascal system allows the user to adjust the borders to accommodate minor variations between individual monitors.)

```
PROCEDURE DRAWCHAR
  (CHARNUM,XCOOR,YCOOR: INTEGER);
BEGIN
  GOTO XY(XCOOR,YCOOR);
  WRITE (CHR(CHARNUM))
END;
```

This procedure uses the built-in UCSD Pascal routine GOTOXY to move the cursor to the desired location and then uses the Pascal WRITE routine to display the desired character on the screen.

The preceding version of the procedure DRAWCHAR does not yield the desired results for CHARNUM values less than 32, since many of the corresponding characters are assigned special meanings by the UCSD Pascal input/output system. Part two will include an alternate version of DRAWCHAR, which uses a POKE procedure written in assembler to store the value CHARNUM in the memory location corresponding to the screen position (XCOOR,YCOOR). This alternate version of DRAWCHAR works for all values of CHARNUM and is considerably faster.

Since the DRAWCHAR routine is reasonably short, it would be relatively easy to type the above declaration in as part of any host program in which it is needed. The purpose of this note is to present several more sophisticated ways provided by the UCSD Pascal system of accomplishing the same thing.

Before proceeding, use the Editor to enter the above procedure as a new workfile and write it out into a disk file named DRAWCHAR.TEXT. Although it is not necessary to store the procedure in a file of the same name, doing so makes it easy to remember where it is stored. In the following section, two methods of including this procedure as part of a host program are illustrated.

B. Including DRAWCHAR in a Host Pascal Program

The following Pascal program displays a subset of the 256 graphics character set on the screen, using the procedure DRAWCHAR. Use the Editor to enter this program exactly as it is shown.

```
PROGRAM CHARSET;
  VAR XCOOR,YCOOR,CHARNUM: INTEGER;
  BEGIN
    XCOOR:= 5;
    YCOOR:= 3;
    CHARNUM:= 32;
    REPEAT
      REPEAT
        DRAWCHAR(CHARNUM,XCOOR,YCOOR);
        XCOOR:= XCOOR + 2;
        CHARNUM:= CHARNUM + 1
      UNTIL XCOOR = 53;
      XCOOR:= 5;
      YCOOR:= YCOOR + 2
    UNTIL YCOOR = 21
  END.
```

This program will not compile correctly in its current form since there is no declaration for the procedure DRAWCHAR. The following two subsections illustrate two methods of correcting this problem.

1. The C(copy Option of the Editor

The C(copy option of the Editor can be used to physically copy the contents of a file into the workfile at the current location of the cursor. The following series of steps will copy the procedure DRAWCHAR into the above program.

- a) Move the cursor to the beginning of the row immediately above the BEGIN statement in the preceding program.
- b) Depress "C" to select the C(copy option of the Editor.

CALL 1-800-321-6850 TOLL FREE

SMALL SYSTEMS JOURNAL

- c) Answer the "Copy: Buffer From file < esc >" prompt by depressing "F".
- d) Answer the "Copy: From what file [marker,marker]?" by entering the file name DRAWCHAR.TEXT.

These steps physically copy the declaration for the procedure DRAWCHAR into the program CHARSET. With this addition the program CHARSET can be compiled and run.

2. The \$INCLUDE Compiler Directive

The compiler directive (*\$! DRAWCHAR.TEXT*) can be placed on the line above the BEGIN in the program CHARSET instead of physically inserting the text of the procedure DRAWCHAR. This directive instructs the compiler to include the contents of DRAWCHAR.TEXT when the program is compiled.

3. General Comments

Regardless of which of the two approaches given above is used, the results are essentially the same. The contents of the file DRAWCHAR.TEXT are compiled as part of the CODE file for the program CHARSET. The \$INCLUDE compiler directive usually requires more memory at compile time than if the text is usually copied into the workfile. Consequently, on systems with 48K bytes of memory the \$INCLUDE directive may not be appropriate for larger programs.

Each of the above approaches requires DRAWCHAR to be compiled each time it is used. The following section shows how to place a compiled version of the routine in the SYSTEM.LIBRARY.

C. Adding DRAWCHAR to the SYSTEM.LIBRARY

The UCSD Pascal system allows the user to group a collection of related functions and procedures together as a unit. Units are discussed in chapter 9 of the *Beginner's Guide to the UCSD Pascal System* and in section 3.3 of the *UCSD Pascal User's Manual*. The major difference between using a unit and using a \$INCLUDE compiler directive is that a unit can be separately compiled and placed in the system library. The compiled routines in the unit are then automatically linked whenever a host program which uses them is run.

1. MYPLOT1 — A UNIT containing two procedures DRAWCHAR and ERASCHAR

The following is a very simple example of a unit containing two procedures. The first is the procedure DRAWCHAR introduced in section A. The other is a related procedure which erases the

character stored at any location on the screen. As illustrated by this example there are two sections in a unit. The first is an INTERFACE section which, in this case, declares the two procedures defined in this unit by giving their names and describing their parameters. These declarations are automatically provided to any host program which uses this unit. This allows the compiler to perform type checking for each invocation of the routines DRAWCHAR and ERASCHAR by the host program. The second section is the IMPLEMENTATION section which includes the actual programs defining DRAWCHAR and ERASCHAR.

```
UNIT MYPLOT1;
INTERFACE
  PROCEDURE DRAWCHAR(CHARNUM,XCOORD,YCOORD:
    INTEGER);
  PROCEDURE ERASCHAR(XCOORD,YCOORD: INTEGER);
IMPLEMENTATION
  PROCEDURE DRAWCHAR; (*PARAMETERS
    DECLARED ABOVE*)
  BEGIN
    GOTOXY(XCOORD,YCOORD);
    WRITE(CHR(CHARNUM))
  END;
  PROCEDURE ERASCHAR; (*PARAMETERS
    DECLARED ABOVE*)
  BEGIN
    DRAWCHAR(32,XCOORD,YCOORD) (* CHR(32) =
    BLANK*)
  END;
END. (*END OF UNIT*)
```

Before proceeding, use the Editor to enter this unit as a new workfile and write it into a file named MYPLOT1.TEXT. The next section describes the steps necessary to place this unit in the system library.

2. Adding MYPLOT1 to the SYSTEM.LIBRARY

The MYPLOT1 unit must be compiled before it can be added to the system library. Units are compiled in the same manner as standard Pascal programs. Leave the Editor and enter the C(ompile command. Answer each of the prompts ("Compile what text?" and "To what codefile?") by entering the file name MYPLOT1. The compiler will place the object version of the MYPLOT1 unit in the file named MYPLOT1.CODE.

The LIBRARY.CODE utility program supplied with the UCSD Pascal system is used to modify the system library. The use of this utility is described in detail in section 4.1 of (2). The following steps create a file named NEW.LIBRARY which includes all of the old SYSTEM.LIBRARY together with the MYPLOT1 unit. Before proceeding, use the FILER to verify that both the files SYSTEM.LIBRARY and MYPLOT1.CODE are present on the disk in disk drive #4 (the top disk drive) and return to the system prompt line.

- a) Execute the program LIBRARY.CODE by depressing "X" and then typing "LIBRARY" or "#5:LIBRARY" If the file LIBRARY.CODE is located on the disk in disk drive #5 (the lower disk drive) in response to the prompt "Execute what file?".
- b) Enter the name NEW.LIBRARY as the name of the output codefile.
- c) When the response "Link Code File ->" is displayed, enter SYSTEM.LIBRARY. The following table of all the segments currently in SYSTEM.LIBRARY will be displayed.

```

0 - TRANSCEN 1154 4 - 0 8 - 0 0
1 - DECOPS   1750 5 - 0 9 - 0 0
2 - PASCALIO 1838 6 - 0 10 - 0 0
3 -          0 7 - 0 11 - 0 0

```

and the prompt line

Segment # to link and < SPACE > N(ew file, Q(uit, A(bort will be presented.

The following sequence of responses links each of the segments currently in the SYSTEM.LIBRARY into NEW.LIBRARY.

```

0 < SPACE >
Seg to link into? 0 < SPACE >
1 < SPACE >
Seg to link into? 1 < SPACE >
2 < SPACE >
Seg to link into? 2 < SPACE >

```

As each segment is linked, its name appears in a similar table for NEW.LIBRARY. Once the old SYSTEM.LIBRARY has been copied into the NEW.LIBRARY type "N" for N(ew file and then enter the file name MYPLOT1.CODE in response to the prompt "Link Code File ->". The previous segment map for SYSTEM.LIBRARY is replaced by the following display:

```

0 - 0 4 -          0 8 - 0 0
1 - 0 5 -          0 9 - 0 0
2 - 0 6 -          0 10 - 0 0
3 - 0 7 - MYPLOT1 52 11 - 0 0

```

The final step in creating NEW.LIBRARY is to link the unit MYPLOT1 from segment 7 into segment 3 of NEW.LIBRARY by entering

```

7 < SPACE >
Seg to link into? 3 < SPACE >

```

Once MYPLOT1 has been linked into NEW.LIBRARY, the segment map table will appear as follows:

```

0 - TRANSCEN 1154 4 - 0 8 - 0 0
1 - DECOPS   1750 5 - 0 9 - 0 0
2 - PASCALIO 1838 6 - 0 10 - 0 0
3 - MYPLOT1  52 7 - 0 11 - 0 0

```

At this point, enter "Q" to terminate the execution of the LIBRARY utility. When the "Notice?" prompt is displayed depress RETURN and the file NEW.LIBRARY will automatically be stored on disk. The next section shows how to use the MYPLOT 1 unit in a Pascal program.

3. Using the MYPLOT1 Unit as a Pascal Program

The following Pascal program is a modification of the program CHARSET presented in section B. Use the Editor to enter this program and then store it in the system file SYSTEM.WRK.TEXT by selecting the U(pdate option when you leave the Editor.

```

PROGRAM CHARSET;
USES MYPLOT1;
VAR XCOOR,YCOOR,CHARNUM: INTEGER;
BEGIN
  (*DISPLAY CHARACTERS*)
  XCOOR:=5;
  YCOOR:=3;
  CHARNUM:=32;
  REPEAT
    REPEAT
      DRAWCHAR(CHARNUM,XCOOR,YCOOR);
      XCOOR:=XCOOR+2;
      CHARNUM:=CHARNUM+1
    UNTIL XCOOR=53;
    XCOOR:=5;
    YCOOR:=YCOOR+2
  UNTIL YCOOR=21;
  (*ERASE CHARACTERS*)
  XCOOR:=5;
  YCOOR:=3;
  REPEAT
    REPEAT
      ERASCHAR(XCOOR,YCOOR);
      YCOOR:=YCOOR+2
    UNTIL YCOOR=21;
    YCOOR:=3;
    XCOOR:=XCOOR+2
  UNTIL XCOOR=53
END.

```

The second line in this program notifies the compiler that this program uses the unit MYPLOT1, which has been placed in the system library. This program uses DRAWCHAR to display several lines of graphics characters and then uses ERASCHAR to erase them one at a time.

CALL 1-800-321-6850 TOLL FREE

SMALL SYSTEMS JOURNAL

Before this program can be compiled and run it is necessary to designate the file NEW.LIBRARY which contains the unit MYPLOT1 as the SYSTEM.LIBRARY. Enter the FILER and use the C(hange option to first change the name of SYSTEM.LIBRARY to OLD.LIBRARY, and then change the name of NEW.LIBRARY to SYSTEM.LIBRARY.

To run this program depress "R". The following sequence of events is automatically initiated.

- The contents of SYSTEM.WRK.TEXT are compiled and placed in SYSTEM.WRK.CODE. During the compile, the INTER-FACE section of MYPLOT1 is accessed to verify the references to DRAWCHAR and ERASCHAR.
- The LINKER is invoked and the object code of procedures referenced in the system library (including DRAWCHAR and ERASCHAR) is linked into SYSTEM.WRK.CODE.
- Once the library routines are linked into the codefile, the file SYSTEM.WRK.CODE is executed.

Subsequent runs of the program simply execute the resultant SYSTEM.WRK.CODE file skipping steps a and b. The automatic compile, link and execute process can only be used for programs stored in the system workfile. If the program CHARSET is stored in a named file (CHARSET.TEXT) which is not in the system workfile, then separate commands must be given for the compilation, the linking and the execution.

Bibliography

- Bowles, Kenneth L., *Beginner's Guide to the UCSD Pascal System*, Peterborough: Byte Books, 1980.
- UCSD Pascal User's Manual*, San Diego: Softech Microsystems, 1978.

Universal Modem Program

Universal Modem Program

This is a BASIC program which will set up a machine code modem routine designed for use with a standard modem (with RS-232). The routine will operate with the modem ports on the Ohio Scientific C1P, C4P, and C8P computers. The 630 and UTI board modem ports are exceptions to this and are not supported by this routine.

This is basically a dumb terminal routine with only two local commands:

CONTROL-D Toggles the output back and forth between full and half duplex mode. (Sometimes echoed as a comma.)

CONTROL-B Returns to BASIC if the routine is operating on a cassette system, or runs "BEXEC*" if it is operating on a disk system, effectively terminating the call.*

Shift-0 is still used to output a delete character code. Since ROM BASIC doesn't process a backspace, the previous character will be omitted from the text, but not on the video screen. The delete code will be displayed as a graphic backspace, a forward space, and another graphic backspace on the ROM BASIC computers.

Note: If this program is to run on a disk system, create two buffers using the change utility before entering the program.

*You must physically hang up the phone to complete call termination.

```

10 REM MODEM PROGRAM
20 FOR I=1 TO 30:PRINT:NEXT I:PRINT"MODEM ROUTINE LOADING"
30 V=PEEK(2):Z=PEEK(164774)
40 IF Z=32 THEN GOSUB 3000:GOTO 60
50 GOSUB 4000
60 FOR I=1 TO 32:PRINT:NEXT I:PRINT"MODEM READY"
70 X=USR(X)
80 RESTORE:GOSUB 500:IF Y=4 THEN RUN"BEXEC*"
90 END
500 PS=1:IF PEEK(9800)=32 THEN PS=2
510 IF Y<>4 OR Z<>32 THEN FOR I=1 TO 48:READ P:PRINT:RETURN
520 READ P,C(1),C(2):IF P THEN POKE P,C(PS):GOTO 520
530 RETURN
540 DATA 9730,8,16
550 DATA 9743,7,15
560 DATA 9723,31,63
570 DATA 9736,31,63
580 DATA 9725,4,10
590 DATA 9738,29,59
610 DATA 9800,32,64
620 DATA 9636,101,75
630 DATA 9766,101,75
640 DATA 9770,101,75
650 DATA 9815,101,75
670 DATA 9670,125,123
680 DATA 9783,125,123
690 DATA 9682,95,164
990 DATA 5296,0,1,0,0,0
1500 FOR I=0 TO 255:IF I THEN READ X
1510 IF X=-1 THEN X=INT(1/256)
1520 POKE I,X:NEXT I
1530 RETURN
2000 DATA 32,13,37,173,0,240,74,144,6,173,1,240,32,67,35
2010 DATA 32,93,-1,240,239,201,2,240,22,201,4,240,21,72,32
2020 DATA 67,35,173,0,240,74,74,144,249,104,141,1,240,76,37
2030 DATA -1,76,13,37,173,63,-1,73,12,141,63,-1,208,225,138
2035 DATA 72,152,72
2040 DATA 169,1,32,190,252,32,198,252,208,5,10,208,245,240,83
2050 DATA 74,144,9,42,224,33,208,243,169,27,208,33,32,200,233
2060 DATA 152,141,19,2,10,10,10,54,237,19,2,141,19,2,168,138
2070 DATA 74,240,49,134,200,74,144,252,208,42,234,185,207,253,205
2080 DATA 21,2,208,38,206,20,2,240,43,160,5,162,200,202,208,253
2090 DATA 136,208,248,240,67,201,1,240,53,160,0,201,2,240,54,160
2100 DATA 192,201,32,240,48,169,0,141,22,2,141,21,2,169,2,141
2110 DATA 20,2,208,36,162,150,205,22,2,208,2,162,14,142,20,2
2120 DATA 141,22,2,169,1,32,190,252,32,207,252,74,144,3,76
2130 DATA 143,253,208,194,160,32,76,167,253,169,0,76,183,253
3000 GOSUB 500
3005 IF Y=4 THEN POKE 574,34:POKE 575,66:IF=16930:GOTO 1500
3008 F=546:GOSUB 1500
3010 POKE 546,44:POKE 592,96
3020 POKE 559,251:POKE 560,2:POKE 576,251:POKE 577,2
3030 POKE 763,41:POKE 764,127:POKE 765,76:POKE 766,45:POKE 767,191
3040 POKE 11,34:POKE 12,2:RETURN
4000 GOSUB 3000
4010 POKE F+65,141:POKE F+66,0:POKE F+67,223
4020 POKE F+68,174:POKE F+69,0:POKE F+70,223
4030 POKE F+193,141:POKE F+194,0:POKE F+195,223
4040 POKE F+196,173:POKE F+197,0:POKE F+198,223
4050 POKE F+1,68:POKE F+2,36
4060 POKE F+7,68:POKE F+8,38
4070 POKE F+5,252:POKE F+11,252:POKE F+34,252:POKE F+42,252
4080 IF Y=4 THEN POKE 6325,52:POKE 6451,2
4090 RETURN

```

Software Catalog: XXXII

Name: **AGS-1 Natal Horoscope**
System: Apple II or TRS-80
Memory: 48K RAM
Language: For Apple II, Applesoft in ROM with DOS 3.2, for TRS-80, Disk Basic 2.3

Hardware: For Apple II, 1 disk drive and line printer, for TRS-80, 2 disk drives and line printer

Description: A very complete calculation program for astrologers. Erects a horoscope for any date and time from A.D. 1800 to 2000, accurate to one minute of arc or better. Printout is in spoked wheel form with many extras: detailed aspectarian, geocentric and heliocentric longitude and latitude, right ascension and declination, retrogrades, 24-hour distance traveled, dignities, and more. Each program has two zodiacs and seven house systems to choose from, and planet and sign glyphs are available for some printers. NATAL HOROSCOPE feeds into 16 other programs for further astrological calculations.

Copies: Must be special-ordered. We tailor to your system.

Price: \$125.00

Author: **Robert S. Hand**

Available: **AGS Software**
Box 28
Orleans, MA 02653

Name: **The Arrow**
System: CBM with new ROM's 2.0 or 4.0
Memory: 8 - 32K
Language: Machine Language
Hardware: Contained in 2716 EPROM

Description: Save/Load at 3600 baud with your C2N cassette deck. BASIC programs, machine code blocks and data files plus Verify, Append and F. Fwd tape positioning supported. Also full 80 x 50 graphics and hex calculator. 10 new commands.

Copies: Just released

Price: \$45

Author: **Milton Bathurst**

Available: **DataCap**
73, rue du Village
B4545 Feueur
Belgium

Name: **The Demo Disk**
System: Apple II
Memory: 48K
Language: Applesoft, Machine
Hardware: Apple II, Disk II
Description: Contains a program exemplifying usage of "Superfront" letters and utilities from *Super Draw and Write* disk. Also includes "Instant Graphics (Sound Option)" from same disk. "Conditioning" from our *Conditioning Life Dynamics* disk, and "Rationality?" from our *Aliveness Life Dynamic* disk, are available, as well as the incomparable "Jungle Safari" from our *Environment Life Dynamic* disk. You get the best of Avant-Garde Creations' programs at an unbelievable price.

Copies: Many

Price: \$9.95 includes disk, game card/drawing card

Author: **Avant-Garde Creations**

Available: **Avant-Garde Creations**
P.O. Box 30161 MCC
Eugene, Oregon 97403

Name: **Chaos Version 2.1**
System: OSI Superboard II or Challenger 1-P
Memory: 4K RAM or more
Language: 6502 Machine
Hardware: Real-time Clock (optional)

Description: CHAOS saves and loads BASIC programs up to 2 times faster than BASIC, consuming up to 50% less tape! Each program may be given a unique file name of any length. The program is not listed as it is saved or loaded. Do you have a real-time clock? CHAOS will save the date and time along with your program! Now for the best part: CHAOS *does not use any BASIC programming memory!* Stop waiting for OSI BASIC — order CHAOS today (or send an SASE for further information).

Copies: On demand

Price: \$12.95 includes CHAOS cassette and complete operating manual.

Author: **Paul Morey**

Available: **PROCOM Software**
8 Hampton South
Southampton, MA 01073

Name: **The Super Bar and Wine Guide**

System: Apple II
Memory: 48K
Language: Applesoft
Hardware: Disk 3.2, 3.3

Description: The new Super Bar and Wine Guide is an education in the art of selecting and enjoying fine wines. This program places at your fingertips the combined knowledge of wine experts, distributors and months of research, the most recent wine prices (1981), as well as fifty-four of the most popular and well known red, white and specialty wines. Included are a complete description of each wine, a *Serving Suggestions* category that offers over two-hundred combinations of food and wine, a *Glossary of Terms* section of the most commonly used words, a complete *Pronunciation Guide*, a section called *Wine Tips* that gives information about usage of wine; and the newest addition is the *Computer Wine Steward*, a program within itself! It lets the computer do the selecting from over two-hundred 'brand name wines' and their most recent prices, from a Meal Selection menu of your favorite dishes (25 Dinner Selections). And finally, a *Bartender's Guide* for forty of the more popular mixed drinks.

Price: \$24.95 includes yearly updates \$5.00

Author: **Donald E. Martin**

Available: **CINE-AERO**
1821 N. Frederic St.
Burbank, California
91505

Name: **Small Business Accounting (SBA)**

System: OSI C4P MF
Language: BASIC under OS65D
Hardware: Printer, 2 Disks (second optional)

Description: Provides double-entry journal system for cash flow analysis and reports. Automatic checking of distribution account totals at time of entry. User-defined fields in data base files; up to 99 expense and income accounts, 999 vendor/customer accounts, with names up to 72 characters. Six digit (XXXX.xx)

capability in base module is expandable. Prints Income Statement, Trial Balance, Charts of Accounts and Vendor/Customer lists. Summary financial information totalable by month, quarter, and YTD. Sorting is available on user specified fields. All records are MDMS-compatible and code allows user system configuration.

Price: \$100.00 (3rd class mail free, 1st class add \$2.00).

Includes: (1) program disk and (1) data disk with sample file. User Manual and Accounting System Guide and sample source documents provided. Program listings only are available for \$20.00 each.

Author: **J.O. Rector**

Available: **Video Ventures**
1708 Beechwood Avenue
Fullerton, California
92635

Name: **Stand-Alone fig-FORTH**

System: OSI, C1, C2, and C4 minifloppy

Memory: 24K

Hardware: No extra hardware required

Description: Complete FORTH high-level language system—no operating system needed. Disk files are OS-65D compatible. Strictly adheres to FIG standards. Includes disk, display and keyboard drivers for OSI. A structured 6502 macro-assembler and disk utilities are also included, plus the FIG portable line editor. These can all be in memory at once with plenty of room for applications. Complete technical documentation and a fig-FORTH glossary are included.

Copies: Just released

Price: \$49.95 check or money order, volume discounts for dealers.

Author: **Michael Butts and Forth Interest Group**

Available: **FORTH Tools**
Box 12054
Seattle, WA 98102

Name: **GRAFFAK APPLE**

System: Apple II

Memory: 32K minimum

Language: BASIC or 6502 machine language

Hardware: Disk and graphic printer

Description: GRAFFAK is a family of programs for reproducing the Hi-Res pages — using grab-the-wire printer graphics. 1x and 2x scaling are standard, and 3x and 5x are available with some printers. Normal and inverse inking is selectable, and variable indent is provided. Features vary with make and model of printer. Packages available for IDS-440, 445 and 460, Anadex DP-9xxx

family, and Epson MX-70 and MX-80 with graphic PROMS.

Price: \$24.95 (+\$1.65 in Ohio) includes diskette and user's guide (specify DOS release and printer model).

Author: **SmartWare**

Available: **SmartWare**
2281 Cobble Stone Court
Dayton, Ohio 45431

Name: **Journey to Mt. Doom**

System: SYM with BAS-1 or KIM 8K BASIC at 2000 H.

Memory: 16K

Language: BASIC

Hardware: Terminal using standard serial I/O ports on SYM or KIM

Description: An adventure game in which you wander through a network of caverns in search of the Necromancer's gold ring. Once you find the ring you must then discover the secret way to Mt. Doom where the ring is to be destroyed. You'll encounter goblins and other creatures along the way and also find treasure. You communicate with the computer with one and two word commands.

Copies: Just released

Price: \$10.00 on cassette tape, ppd. in U.S. only

Author: **Lee Chapel**

Available: **Lee Associates**
2349 Wiggins Ave.
Springfield, IL 62704

Name: **DISASM (2.0)**

System: Apple II or Apple II Plus

Language: Machine

Hardware: Disk

Description: DISASM serves as an invaluable aid for understanding and modifying machine language programs. It is a symbolic disassembler which generates source code, with labels, directly compatible with DOS Toolkit, Lisa and S-C assemblers. Default labels are categorized as page zero, external or internal. Optional user-defined label name table permits substitution of more meaningful label assignments. Monitor ROM label name table included with over 100 standard subroutine and memory address names. Equate definitions generated in ascending order. No restriction on disassembled block length. Correctly disassembles displaced object code, auto source segmentation for easier reading, and more!

Copies: Over 40

Price: \$30.00 (Program diskette and user documentation)

Author: **Bob Kovacs**

Available: **RAK-WARE**
41 Ralph Road
West Orange, NJ 07052

Name: **DOS/65**

System: All 65xx

Memory: minimum of 16K to 24K

Language: machine

Hardware: 8" single density, soft sect disk

Description: DOS/65 is a flexible disk operating system for the 6502 which allows the user to configure the system for his environment similar to what CP/M allows for the 8080/Z-80. Included are a two pass assembler, an editor, a debugger, a sysgen routine and other utilities. Standard system is configured only for Tarbell controller but full interface instructions are included.

Copies: New Release

Price: \$100-\$150 (more for custom)

Author: **Richard A. Leary**

Available: **Richard A. Leary**
1363 Nathan Hale Drive
Phoenixville, PA 19460

Name: **Poker**

System: Apple II Plus

Memory: 48K w/ROM Applesoft

Language: Applesoft

Hardware: Disk II

Description: Tired of playing "Poker" games that amount to nothing more than Blackjack? This game pits four computer opponents against you and allows for up to three rounds of betting. You can exchange cards, pass, bluff, call at anytime, and bet little, big or fold—and so can they. A detailed model of real poker.

Copies: Just Released

Price: \$15

Author: **Jeff Brower**

Available: **Galaxy Sales, Inc.**
30815 28th Avenue South
Federal Way, WA 98003

Name: **Disk Directory**

System: Pet 16K/32K + 3040
Disk Drive

Memory: Minimum 16K

Language: BASIC

Description: Indexes on master diskette (drive 0), the directory of diskette in drive 1. Enables all directories of all diskettes to be kept on one master diskette. Options available: format diskette; update or create index; display single directory or all directories indexed. Search option: finds and displays which diskette(s) holds a particular programme; with auto load facility. Summary of all indexes: disk ID; name and bytes free; delete entry.

Price: \$25

Author: **D. Milnes**

Available: **13, Delmont Close**
Whitelee Road
Batley
West Yorkshire WF178AQ.
England



The Newest In

Apple Fun

We've taken five of our most popular programs and combined them into one tremendous package full of fun and excitement. This disk-based package now offers you these great games:

Mimic—How good is your memory? Here's a chance to find out! Your Apple will display a sequence of figures on a 3x3 grid. You must respond with the exact same sequence, within the time limit.

There are five different, increasingly difficult versions of the game, including one that will keep going indefinitely. Mimic is exciting, fast paced and challenging—fun for all!

Air Flight Simulation—Your mission: Take off and land your aircraft without crashing. You're flying blind—on instruments only.

A full tank of fuel gives you a maximum range of about 50 miles. The computer will constantly display updates of your air speed, compass heading and altitude. Your most important instrument is the Angle of Ascent/Bank Indicator. It tells if the plane is climbing or descending, whether banking into a right or left turn.

After you've acquired a few hours of flying time, you can try flying a course against a map or doing aerobic maneuvers. Get a little more flight time under your belt, the sky's the limit.

Colormaster—Test your powers of deduction as you try to guess the secret color code in this Mastermind-type game. There are two levels of difficulty, and three options of play to vary your games. Not only can you guess the computer's color code, but it will guess yours! It can also serve as referee in a game between two human opponents. Can you make and break the color code...?

Star Ship Attack—Your mission is to protect our orbiting food station satellites from destruction by an enemy star ship. You must capture, destroy or drive off the attacking ship. If you fail, our planet is doomed...

Trilogy—This contest has its origins in the simple game of tic-tac-toe. The object of the game is to place three of your colors, in a row, into the delta-like, multi-level display. The rows may be horizontal, vertical, diagonal and wrapped around, through the "third dimension". Your Apple will be trying to do the same. You can even have your Apple play against itself!

Minimum system requirements are an Apple II or Apple II Plus computer with 32K of memory and one minidisk drive. Mimic requires Applesoft in ROM, all others run in RAM or ROM Applesoft.

Order No. 0161AD \$19.95

Paddle Fun

This new Apple disk package requires a steady eye and a quick hand at the game paddles! It includes:

Invaders—You must destroy an invading fleet of 55 flying saucers while dodging the carpet of bombs they drop. Your bomb shelters will help you—for a while. Our version of a well known arcade game! Requires Applesoft in ROM.

Howitzer—This is a one or two person game in which you must fire upon another howitzer position. This program is written in HIGH-RESOLUTION graphics using different terrain and wind conditions each round to make this a demanding game. The difficulty level can be altered to suit the ability of the players. Requires Applesoft in ROM.

Space Wars—This program has three parts: (1) Two flying saucers meet in laser combat—for two players, (2) two saucers compete to see which can shoot out the most stars—for two players, and (3) one saucer shoots the stars in order to get a higher rank—for one player only. Requires Applesoft.

Golf—Whether you win or lose, you're bound to have fun on our 18 hole Apple golf course. Choose your club and your direction and hope to avoid the sandtraps. Losing too many strokes in the water hazards? You can always increase your handicap. Get off the tee and onto the green with Apple Golf. Requires Applesoft.

The minimum system requirement for this package is an Apple II or Apple II Plus computer with 32K of memory and one minidisk drive.

Order No. 0163AD \$19.95

Solar Energy For The Home

With the price of fossil fuels rising astronomically, solar space-heating systems are starting to become very attractive. But is solar heat cost-effective for you? This program can answer that question.

Just input this data for your home: location, size, interior details and amount of window space. It will then calculate your current heat loss and the amount of gain from any south facing windows. Then, enter the data for the contemplated solar heating installation. The program will compute the NET heating gain, the cost of conventional fuels vs. solar heat, and the calculated payback period—showing if the investment will save you money.

Solar Energy for the Home: It's a natural for architects, designers, contractors, homeowners... anyone who wants to tap the limitless energy of our sun.

Minimum system requirements are an Apple II or Apple II Plus with one disk drive and 28K of RAM. Includes AppleDOS 3.2.

Order No. 0235AD (disk-based version) \$34.95

Math Fun

The Math Fun package uses the techniques of immediate feedback and positive reinforcement so that students can improve their math skills while playing these games:

Hanging—A little man is walking up the steps to the hangman's noose. But YOU can save him by answering the decimal math problems posed by the computer. Correct answers will move the man down the steps and cheat the hangman.

Spellbinder—You are a magician battling a computerized wizard. In order to cast death clouds, fireballs and other magic spells on him, you must correctly answer problems involving fractions.

Whole Space—Pilot your space craft to attack the enemy planet. Each time you give a correct answer to the whole number problems, you can move your ship or fire. But for every wrong answer, the enemy gets a chance to fire at you.

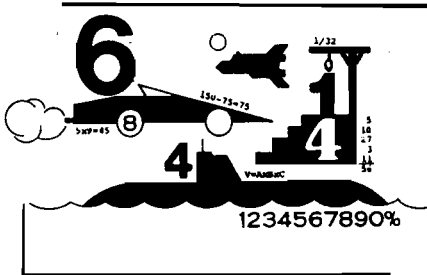
Car Jump—Make your stunt car jump the ramps. Each correct answer will increase the number of buses your car must jump over. These problems involve calculating the areas of different geometric figures.

Robot Duel—Fire your laser at the computer's robot. If you give the correct answer to problems on calculating volumes, your robot can shoot at his opponent. If you give the wrong answer, your shield power will be depleted and the computer's robot can shoot at yours.

Sub Attack—Practice using percentages as you maneuver your sub into the harbor. A correct answer lets you move your sub and fire at the enemy fleet.

All of these programs run in Applesoft BASIC, except Whole Space, which requires Integer BASIC.

Order No. 0160AD \$19.95



Skybombers

Two nations, separated by The Big Green Mountain, are in mortal combat! Because of the terrain, their's is an aerial war—a war of SKYBOMBERS!

In this two-player game, you and your opponent command opposing fleets of fighter-bombers armed with bombs and missiles. Your orders? Fly over the mountain and bomb the enemy blockhouse into dust!

Flying a bombing mission over that innocent looking mountain is no milk run. The opposition's aircraft can fire missiles at you or you may even be destroyed by the bombs as they drop. Desperate pilots may even ram your plane or plunge into your blockhouse, suicidally.

Flight personnel are sometimes forced to parachute from badly damaged aircraft. As they float helplessly to earth, they become targets for enemy missiles.

The greater the damage you deal to your enemy, the higher your score, which is constantly updated at the bottom of the display screen.

The sounds of battle, from exploding bombs to the pathetic screams from wounded parachutists, remind each micro-commander of his bounden duty. Press On, SKYBOMBERS—Press On!

Minimum system requirements: An Apple II or Apple II Plus, with 32K RAM, one disk drive and game paddles.

Order No. 0271AD (disk-based version) \$19.95



*A trademark of Apple Computer Inc.

PETERBOROUGH, N.H. 03458
603-924-7296

Instant Software™

6502 Bibliography: Part XXXII

926. Call Apple 3, No. 8 (October, 1980)

Reynolds, Lee, "Hexadecimal and Binary Number Systems," pg. 7-10.

A tutorial on HEX/DEC and the Apple monitor.

Weston, David, "Comparing Ten Sort Algorithms," pg. 13-19.

A good demo of various sort methods, with listings for the Apple.

Lee, Scott and Rose, Steve, "Demuffin!," pg. 21.

Use this DOS 3.3 program to transfer DOS 3.3 programs to DOS 3.2 disks.

Robinson, Alan H., "Apple FORTRAN: First Impressions," pg. 23-24.

A review of FORTRAN for the Apple.

Manly, Kenneth, "Why Don't You Watch Where You're Going?," pg. 25-28.

A tutorial on the Apple Hi-Res Screen Function, with demo listings.

Anon., "Use FID with DOS 3.2!," pg. 34.

How to use the handy file handler with DOS 3.2 disks.

Murdoch, David M., "&CATALOG," pg. 34.

A short POKE routine to enable the use of the Ampersand to produce the Catalog command on the Apple.

Capes, Nelson R., "Data Communications with the Electronic Systems Card," pg. 37-41.

An inexpensive way to interface the Apple with a modem and download from other systems, with listing to implement the system.

Huelsdonk, Bob, "Making BASIC Behave: Part VI," pg. 43-44.

Some handy Apple utilities and some input hints.

Kluepfel, Charles, "Applesoft Program Splitter Mods," pg. 45-48.

Some improvements on a previously published Apple program.

Lewellen, Tom K., "A Patch for 80 Column Video Boards and Apple Pascal," pg. 51-52.

Some hints for the Apple Pascal users.

Reynolds, Lee, "Decimal Packing and Unpacking," pg. 55-56.

A memory saving technique for the Apple.

Eckert, Paul and Bronstein, Neil, "GOTO A," pg. 56.

How to use the forbidden variable in a GOTO on the Apple.

Lustig, Henry G., "Line Number Cross Reference for Applesoft," pg. 58-59.

A utility listing for the Apple.

927. The Cider Press (July, 1980)

Poindexter, Ed, "Machine Language Mysteries Revealed! A BASIC Approach," pg. 6-7.

A tutorial on the Apple machine language.

Crossman, Craig, "Fun With Assembly Language," pg. 8-9.

An interesting article discussing some simple assembly language operations on the Apple.

Uhley, John, "Credit Plus," pg. 10-12.

A short machine language program to assist programmers in adding credit statements to their listings on the Apple.

Uhley, John, "DOS with Trace," pg. 12.

A discussion of how this combination is achieved on the Apple.

Nareff, Max J., "Program Decimal ROM," pg. 13-14.

A Pascal program to convert decimal numbers to Roman Numerals.

Nareff, Max J., "Addfractns," pg. 14.

A program in Apple Pascal.

Rowe, Pete, "The Mysterious Orange Vertical Line," pg. 15.

A discussion of a quirk of Apple Hi-Res.

Rowe, Pete, "Apple ASCII/BASIC Token and Hi Res Address Reference," pg. 16-17.

Reference information for Apple programmers.

928. Apple Cookbook 1, No. 4 (November, 1980)

Weber, Stan, "Getting Fancy with Formats," pg. 1-2.

A routine providing formatting for Apple programs.

Anon., "Informer Update," pg. 5-7.

An update routine for a previously published Apple utility.

Busdiecker, Roy, "The Number Game: An Introduction to Computer Arithmetic."

A tutorial with a listing to convert decimal numbers to binary equivalents.

929. The Seed 2, No. 11 (November, 1980)

Anon., "Apple Pi Conventions," pg. 2.

A listing to assist programmers in writing credit lines for programs submitted to newsletters.

Dulk, G.A., "Use of Apple As A Word Processor" pg. 4-8.

The Pascal system has many of the desirable features of a word processor.

White, Harry, "DISK, Shape Up!," pg. 9.

A listing to permit a quick and dirty examination of Apple Hi-Res shape tables.

Eliason, Andrew H., "The Apple II Hardware," pg. 11-13.

A tutorial on the Apple II keyboard.

Duplissey, Claude, "Applesoft Strings," pg. 14.

A tutorial on Apple strings.

930. The Cider Press (August/September, 1980)

Silverman, Ken, "Configuring Your Apple—Don't Overload Your Apple II," pg. 6-7.

A chart of the voltage and current requirements of various Apple boards and peripherals.

Weiglin, Peter C., "Build a Better Error Trap," pg. 8.

Help the Apple in its quest for valid data.

Rowe, Pete, "Int and FP Machine Language Interface," pg. 12-19.

A tutorial for the Apple with several listings in machine language.

Wilson, Gene, "Je M'Apple' Pascal," pg. 20-23.

A tutorial on Apple Pascal with a listing for a disk utility.

Norris, Paul, "Why Pascal? Why Not?," pg. 24.

The pro's and con's of Apple Pascal are discussed.

931. The Target (September/October, 1980)

- Hollibaugh, Larry, "Touch-Tone Dialer," pg. 2-5.
Generate touch-tones with the aid of AIM-65 and an AY-3-8910 programmable sound generator using this article and accompanying hardware and program listing.
- Bresson, Steve, "Offset Load," pg. 6.
Program for the AIM 65 to load an object program from tape at an offset from the save address.
- Buchen, D., "EPROM Programmer," pg. 8-14.
A programmer for the AIM 65 to program EPROM's 2708, 2716, 2516 and 2532.

932. Microcomputer Index 1, No. 2 (April-June, 1980)

A subject index covering over 850 microcomputer magazine articles, many on 6502-related subjects.

933. Microcomputer Index 1, No. 3 (July-September, 1980)

A subject index covering over 1000 microcomputer magazine articles, many on 6502-related subjects.

934. Softalk 1 (November, 1980)

- Stinson, Craig, "The All-American Apple Music Machine," pg. 14-21.
A discussion of the Apple and various systems for generating music on the Apple.
- Wagner, Roger, "Assembly Lines," pg. 34-35.
Everyone's guide to machine language on the Apple, part 2.

935. Stems From Apple 3, Issue 11 (November, 1980)

- Dial, Wm. R., "Mystery Program," pg. 3.
A short whimsy for the Apple.
- Ward, Dennis, "Dennis Does It Again," pg. 7-9.
Several short programs for the Apple.
- Hoggatt, Ken, "The Twelve Days of Christmas," pg. 9-10.
Listings for Apple Pascal and for Applesoft BASIC.
- Shelton, Janice and Hoggatt, Ken, "Christmas Gift Exchange," pg. 11-12.
Apple Pascal and Applesoft listings for Apple.
- Anon., "IAC Application Note: Program Transfer," pg. 14-20.
An Apple Pascal routine for sending and receiving files or whole volumes over serial lines.

936. From The Core (November, 1980)

- Budge, Joe, "Natterings from the Nabob," pg. 2.
Among other tips a fix for a bus in early issues of DOS 3.3 for the Apple.
- Andrews, Wilbur C., "PTEXT," pg. 5.
PTEXT is a text formatting program written in Apple Pascal.
- Graham, Johnny, "16 to 13 Sector Hardware Mod," pg. 8.
A mod that allows switching the disk controller card in the Apple Disk system from 13 to 16 sector and vice versa.

937. Sym-Physis Issue 5/6 (September-December, 1980)

- Cole, Stephen E., "Power-On Routine," pg. ?.
A power-on routine for the SYM-1.
- Campbell, Hugh, "Apple Tape Loader Program," pg. 21-22.
A loader for transferring Apple tapes to the SYM.

Kwok, Kin-Ping, "A BASIC Word Processing System," pg. 41-43.

Two BASIC programs providing a word processing capability on the SYM-1.

Staff, "How to 'Rewire' the VIA at \$A800."

Two methods applicable to the SYM-1.

Anon., "Mystery Program," pg. 46-47.

SYM program for file handling.

938. Personal Computing 4, No. 12 (December, 1980)

- Schlarb, Keith N., "Required Reading," pg. 68-71.
Apple program for storing information in the classroom.
- Staff, "London's World Micro Chess Champions," pg. 79-80.
Chess programs based on the 6502 take most of the honors.

939. Compute! 2, Issue 7, No. 6 (November/December, 1980)

- McNeil, Arthur L., "Small Computers and Small Libraries," pg. 24-29.
A PET program to print out catalogue cards for the library.
- Richter, Mike, "Efficiency with Subroutines," pg. 30-32.
A tutorial for the PET system.
- Flynn, Brian J., "Computing Correlation Coefficients," pg. 36-41.
Listing and explanation of a listing for 6502 micros.
- Baker, Al, "Al Baker's Programming Hints: Apple," pg. 42-43.
Exploring the Apple paddle and the joystick.
- Kelly, Derek A., "The Anatomy of a Word-Research Processing Program for the Apple," pg. 44-49.
A model for structured programming.
- Castevens, Philip, "Hard Disks for the Apple." Discussion and directory of disk hardware.
- Harris, Neil, "Times Square on your Atari," pg. 56-58.
A scrolling program for the Atari.
- Lindsay, Len, "Error Reporting System for the Atari," pg. 58-59.
Gives Atari error messages in plain language instead of just error numbers.
- White, Jerry, "Monthly Bar Graph Program," pg. 61.
An Atari BASIC tutorial with bar routine listing.
- Seivert, William D., "Card Games in Graphics Modes 1 and 2," pg. 62-63.
Hints for Atari game programmers.
- Bruun, James L., "Using TAB in Atari BASIC," pg. 64.
Create a TAB function for your Atari.
- Brannon, Charles, "'Pokin' Around," pg. 66.
A tutorial on the Atari POKE function.
- Stewart, Charles, "Coded Data for OS11P," pg. 70-71.
A program for OSI computers which hides data statements in ASCII code.
- Garland, W. Blaine, "OSI Graphics Character Set," pg. 71.
A demo program to show the characters and memory location.
- Stanford, Charles L., "Atari Joysticks on the OSI C1P," pg. 72-77.
Interface the Atari joystick to the C1P, hardware and software. With a listing for a typical game with joystick.

Butterfield, Jim, "BASIC CBM 8010 Modem Routines," pg. 78.

All about using the PET with a Modem, with listings of required software.

Busdeicker, Roy, "Programmer's Notes for the CBM 8032," pg. 80-82.

Discussion and hints for using the CBM 8032 micro.

Brannon, Charles, "Keyprint," pg. 84-86.

A routine to enable the PET screen to be dumped to a printer at any time.

Butterfield, Jim, "PET 4.0 ROM Routines," pg. 88-90.

Addresses of PET ROM routines.

Butterfield, Jim, "BASIC 4.0 Memory Map," pg. 92-93.

Useful information for PET users.

Deal, Elizabeth, "Algebraic Expression Input for the PET, Version 2," pg. 94-96.

Discussion of inputting on the PET with utility routine.

Winter, M.J., "Defining a Function Whilst Running a Program," pg. 96.

A routine for the PET.

Butterfield, Jim, "Machine Language Addressing Modes," pg. 98-100.

A discussion of 6502 addressing modes, oriented to the PET.

Covitz, Frank, "Visible Memory Printer Dump," pg. 104-109.

Print Dump for the PET/MTU visible memory/CBM 2022 printer combination.

Baker, Robert W., "Disk Lister," pg. 110-114.

A disk cataloguing program for the PET and 2040 disk.

Zumchak, Gene, "Nuts and Volts," pg. 116-121.

Discussion of the 6502/6522 combination for I/O functions.

DeJong, Marvin L., "Interfacing the Am9511 Arithmetic Processing Unit," pg. 122-127.

Use of the Am9511/6502 combination, with driver routines.

Butterfield, Jim, "Interfacing KIM/SYM/AIM/OSI with BASIC," pg. 128-131.

Discussion of single board monitor systems.

Herman, Harvey B., "KIM-1 Tidbits," pg. 134-136.

A program for KIM which makes data statements from a machine language program.

Flynn, Christopher J., "AIM 65 Tape Copy Utility," pg. 137-139.

A short routine to make direct tape copying easy.

Wells, George, "Combining BASIC and Machine-Language Programs on Tape," pg. 140-142.

A procedure for SYM-1 users to combine BASIC and machine language programs in a single cassette tape file.

Bean, Fred D., "Base Converter," pg. 144.

A PET program for converting decimal numbers to numbers with other bases.

940. The Apple-Dillo (November, 1980)

Huffman, David, "PLE Notes," pg. 3-4.

Some new functions possible with special macros entered into the Program Line Editor utility.

Teas, George, "Pascal Primer," pg. 5.

Discussion of WAIT routine for Pascal users.

Bartley, David, "Getting There Faster in Applesoft: Part II," pg. 5-7.

Two machine-language enhancements for the Applesoft GOTO interpreter.

941. KB Microcomputing No. 47 (December, 1980)

Baker, Robert W., "PET-Pourri," pg. 7-8.

Discussion of PET ROM changes, character generator ROMs, and programming hints.

Bendix, Peter, "Music Transcriber," pg. 43-63.

Write sheet music instantly on your TV screen using the KIM and a piano-like keyboard.

Kupke, D., "Super Sound with your Superboard II," pg. 130-131.

A simple and inexpensive modification to unlock the OSI Superboard II's secrets of sound generation.

Urschel, Robert, "The GI Programmable Sound Generator," pg. 134-140.

Use a music/sound effects chip with the Apple, with music generation program listing.

Kelly, Derek A., "Computerized Project Management," pg. 142-148.

A program for the Apple to help plan and schedule complex projects.

Mendelsohn, Stephen, "Hard Copy for the OSI Challengers," pg. 165-166.

A simple modification accommodating both hard copy and cassette I/O on the Challenger IIP.

Davison, John W., "Apple II Plus Plus," pg. 214.

Upgrade your Apple to Apple II Plus and more.

Piper, Neil, "Give Character to your PET Printer," pg. 218-220.

Creating user-defined characters on the Commodore 2022 and 2023 printers.

942. BYTE 5, No. 12 (December, 1980)

Martellaro, John, "Sargon II," pg. 114-118.

An improved Chess-Playing program for the Apple II.

943. L.A.U.G.H.S. 2, No. 7 (December, 1980)

Finn, Mike, "The RWTS Subroutine: Part I."

A tutorial for the Apple disk system. Includes a diagnostic program.

944. Creative Computing 6, No. 12 (December, 1980)

Berggren, Stephen R., "Christmas Tree," pg. 124-125.

Decorate the Christmas Tree with this program for the Apple.

Berggren, Stephen R., "Apple Nuclear Power Plant," pg. 128-137.

Try your skill in running a reactor.

Raymer, Paul, "Weather Station," pg. 142.

Bring your Apple in touch with the real world with this weather program.

Blank, George, "Outpost: Atari," pg. 200-201.

Programming hints for the Atari microcomputer.

Carpenter, Chuck, "Apple-Cart," pg. 202-207.

Discussion of 6502 books, the Galfo/Massimo CW-RTTY communications programs, software by phone, new Apple boards, etc.

945. Nibble No. 7 (December, 1980)

Weinstock, Michael D., "Apple A.I.M.," pg. 9-17.

Automatic Intelligent Mailing list and label program for the Apple.

Floeter, Alan D., "Apple Concordance," pg. 21-26.

An Apple utility to locate variables in a listing.

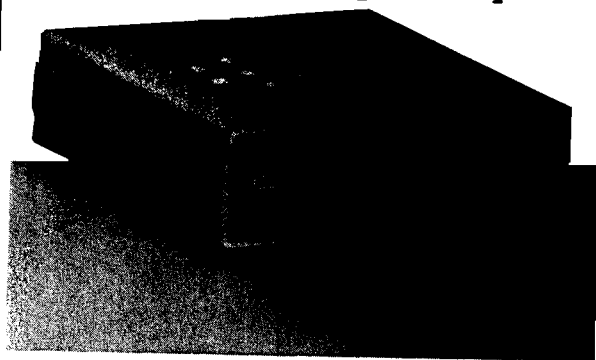
Reynolds, William, III, "Tough Plus!," pg. 29-31, 43, 53.

New enhancements and a "Find and Replace."

Guy, Rudy A., "Low Score II," pg. 33-37.

A graphics game for the Apple.

**INTRODUCING
COGNIVOX Series VIO-1000
A Revolutionary New
Voice Input and Output Peripheral**



**High Fidelity Voice Response
Industrial Quality Recognition**

PET — AIM-65 — APPLE II

COGNIVOX series VIO-1000 is a top-of-the-line voice I/O peripheral for business and educational applications and the demanding hobbyist.

It can be trained to recognize words or short phrases drawn from a vocabulary of 32 entries chosen by the user. It will talk back with up to 32 words or short phrases. In disk based systems, response vocabularies can be stored on the disk and brought to memory as needed, giving an effectively unlimited number of vocabulary entries. The quality of voice response is excellent, and it is far superior to that of speech synthesizers.

COGNIVOX series 1000 comes complete and ready to plug into your computer (the computer must have at least 16K of RAM). It connects to the parallel I/O port of the PET, to the game paddle connector on the Apple and to the J1 port on the AIM-65. Connectors are included as required. Also included are a microphone, cassette with software and extensive user manual. A built-in speaker/amplifier is provided as well as a jack for connecting an external speaker or amplifier.

Software supplied with COGNIVOX includes two voice operated, talking video games, VOTH and VOICETRAP. These games are absolutely captivating to play, and the only voice operated talking games that are commercially available.

Adding voice I/O to your own programs is very simple. A single statement in BASIC is all that is required to say or to recognize a word. Complete instructions on how to do it are provided in the manual.

In keeping with the VOICETEK tradition of high performance at affordable price, we have priced COGNIVOX series 1000 at the unbelievably low, introductory price of **\$249** (plus \$5 shipping in the US, CA add 6% tax. Foreign orders welcome, add 10% for handling and shipping via AIR MAIL). When ordering, please give the make and model of your computer, the amount of RAM and whether you have disks or not.

In addition to COGNIVOX series VIO-1000, VOICETEK manufactures a complete line of voice I/O peripherals for most of the popular personal computers. Speech recognition-only peripherals are available for the 8K PET and the 4K AIM.

For more information call us at 805-685-1854 or write at the address below.

Dealer Inquiries invited.

VOICETEK

Dept E, P.O. Box 388
Goleta, CA 93116

ADVERTISERS' INDEX

MAY 1981

Advertiser's Name	Page
Aardvark Technical Services	21
Abacus Software	57
Aurora Software Associates	80
Avant-Garde Creations	57
Beta Computer Devices	37
The Book	54
Broderbund Software	83
Computer Applications Tomorrow	72
The Computerist, Inc.	IFC
Computer Mail Order	61
Connecticut Information Systems	44
Consumer Computers	43
Continental Software	51
Creative Computing	22
Decision Systems	57
Digibyte Systems Corp.	58
Dr. Dobb's Journal	12
Eastern House Software	70, 73
Human Engineered Software	70
Instant Software	90-91
Jini Micro Systems, Inc.	64
Lazer Systems	8
LJK Enterprises	42
MICRO Ink, Inc.	15, IBC
MICRO Classifieds	54, 80
Micro Interfaces, Inc.	74
Microsoft Consumer Products	1
MicroSoftware Systems	74
Micro Technology Unlimited	2, 30
Micro-Ware Distributing	83
Mittendorf Engineering	41
Nibble	18
Nikrom Technical Products	70
Ohio Scientific	BC
Ohio Scientific "Small Systems Journal"	84-87
Perry Peripherals	70
Progressive Computing	27
Rainbow Computing	7
Rosen Grandon Associates	57
Serendipity Systems, Inc.	73
Simulations Programming	74
Small Business Computer Systems	74
Softape	83
Southeastern Software	4
Southwestern Data Systems	73
Strategic Simulations, Inc.	22
TSE-Hardside	28-29
Versa Computing	27
Voicetek	95
Western Micro Data Enterprises	44

Why Advertise in MICRO?

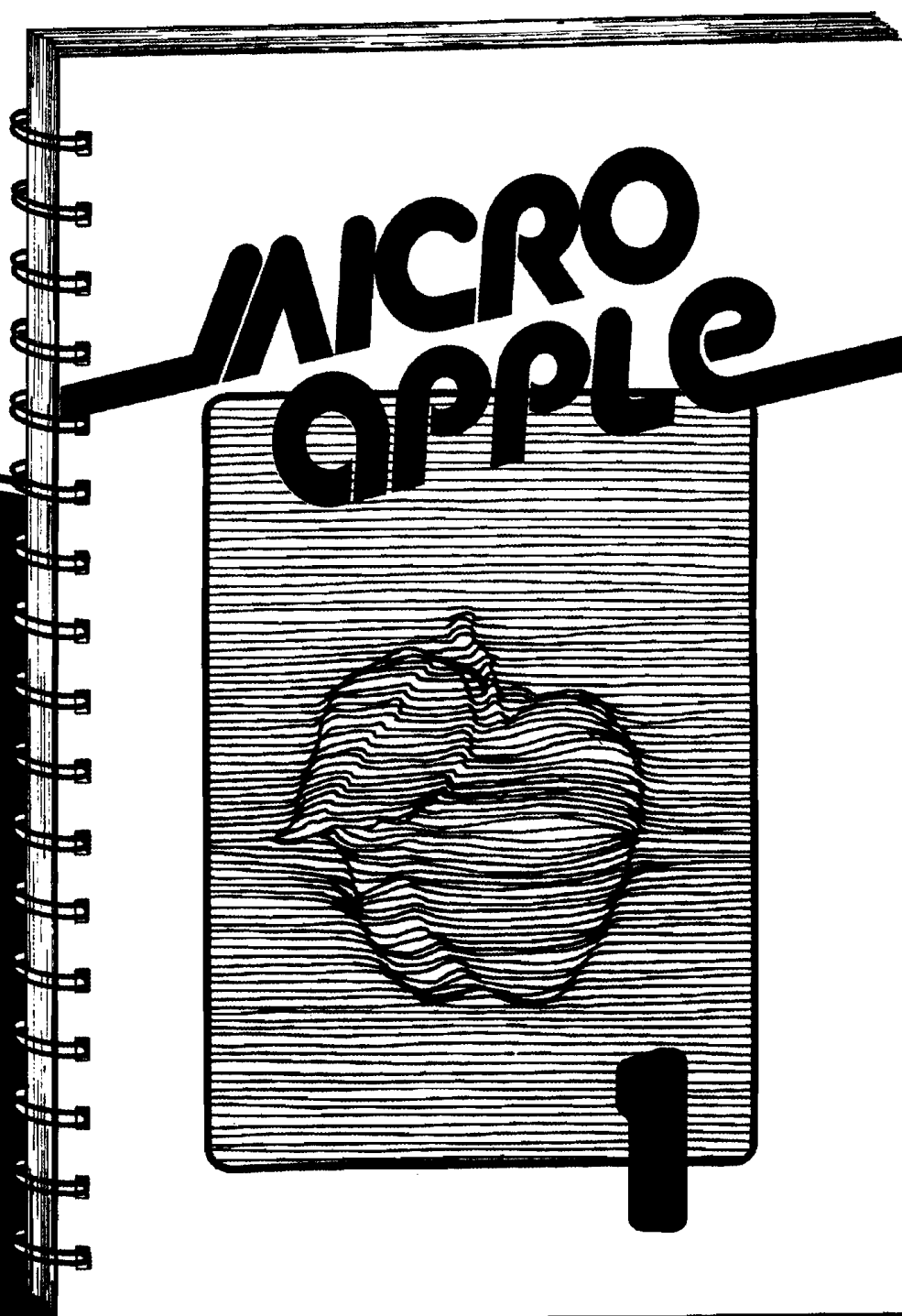
Find Out!

Call (617) 256-5515

Ask for Cathi Bland

- Crossman, Craig, "Apple Tricks," pg. 39.
A program routine to prevent an inadvertent 'reset' and a routine to clear the screen with the ampersand.
- Laird, Alexander, "Fun with the Apple Monitor," pg. 47.
Discussion of the Apple assembler.
- Reynolds, William, III, "Tracing the Apple DOS 3.2 as it Boots," pg. 50.
Step-by-step description of what happens in booting DOS.
- Reynolds, William, III, "Calling the RWTS from BASIC," pg. 50.
A short discussion of RWTS on the Apple.
- Harvey, Mike, "Watch Out for Graphics Overflow," pg. 61.
Tips for avoiding space problems in graphics programs.
- Laird, Alexander, "Get Controls, CHR\$, and Things," pg. 61.
How to implement several useful commands on the Apple.
- 946. The Harvest 2, No. 4 (December, 1980)**
- Schumacher, Kurt G., "Applesoft Variable GOSUB," pg. 1-3.
Hints on implementing the GOSUB command; with 3 listings for the Apple.
- Russ, John, "Universal Input Function for Fortran," pg. 7-8.
An input program for Apple Fortran.
- Anon., "Ask Mr. Apple," pg. 9
Some hints on speeding up Applesoft commands.
- Hartley, Tim, "Changing Volume Numbers," pg. 9.
Change the number on your diskettes with this short routine.
- Dial, Wm. R., "Backwards Apple," pg. 10.
A short program demonstrating a seldom-used possibility of the Apple TAB function.
- Anon., "Not Another Hello Program!," pg. 12.
A Hello program for Apple disks.
- 947. Interactive Issue 2 (Summer, 1980)**
- Anon., "AIM 65 Graphics," pg. 4-5.
Two plotting programs, AIMPLOT and AIMGRAPH, with listings.
- Butterfield, Jim, "Inside BASIC," pg. 6-8.
BASIC Token List, Zero Page Usage, BASIC Entry Points, for the AIM 65.
- Anon., "AIM 65 Sound," pg. 8.
Add a speaker to your AIM 65.
- Anon., "Disassembler Utility," pg. 11.
A utility for the AIM 65 to slow down the display of instructions.
- Reo, Frank, "Offset Loader for AIM 65," pg. 13.
A routine to load object code to a different location.
- Brinkmann, G., "BASIC Banner Program," pg. 15.
Print out banners with this short routine for the AIM 65.
- Reardon, Mark, "Parity Bit Generator Program," pg. 15.
A short machine language program to generate odd or even parity bits for the AIM ASCII characters.
- 948. The G.R.A.P.E. Vine (November, 1980)**
- Anon., "Free Disk Space," pg. 4.
Two short programs for the Apple.
- Anon., "Hello Program," pg. 9.
An appealing Hello program for the Apple disk.
- 949. Interactive Issue 3 (Winter, 1980)**
- Sellers, George, "Solving Simultaneous Equations Using BASIC," pg. 4-5.
Use the AIM 65 to solve up to 20 equations and 20 unknowns.
- Evans, Mel, "Learn to Touch Type," pg. 6-7.
An AIM 65 program to assist the typing learner.
- Smith, Gordon, "BASIC Time Saver," pg. 8-10.
An AIM utility combining automatic line numbering and common BASIC command automatic typist.
- DeJong, Marvin, "Interrupt Driven Keyboard," pg. 12.
A listing of a routine that reads the AIM 65 keyboard on an interrupt basis. One possible use is in sending Morse code.
- Anon., "Super Simple Auto-Start," pg. 15.
A short utility for the AIM 65.
- Anon., "Temperature Conversion Program," pg. 18.
A program for the AIM which prints out Fahrenheit/Centigrade conversion tables.
- April, Georges-Emile, "BASIC USR Helper," pg. 18-19.
Routines to ease the use of USR(X) on the AIM 65.
- Berges, Antonio, "BASIC Recovery Procedure," pg. 20.
How to recover from an error in hitting the wrong AIM key.
- 950. Peek(65) 1, No. 11 (November, 1980)**
- Loos, James, "Modifying the OSI Video Display," pg. 2-8..
How to change the OSI C1P from a 24 x 24 display to 29 x 48.
- Jones, David A., "Cassette Corner," pg. 12-14.
Hints for using cassettes with OSI systems.
- 951. The Cider Press (November, 1980)**
- Anon., "DOM, Disk of the Month, November," pg. 4.
Several useful utilities, for the Apple. Also a 16 sector (DOS 3.3) utility disk.
- Weiglin, Peter C., "Formatting: Part Two," pg. 6-7.
A good tutorial on formatting on the Apple.
- Anon., "Wow! Try These Patches on DOS 3.2," pg. 10.
New features for the Apple DOS.
- Anon., "Plug in a 6809E," pg. 11.
The 6809E can execute programs faster than the 6502 but can co-exist with the 6502 in the new mod.
- Nareff, Max J., "Which DOS is Dat in Dere?," pg. 11.
A short command to print whether the DOS is in effect on your Apple is 3.0, 3.1, 3.2, 3.2.1 or 3.3.
- Pfeifer, Frank J., "Swatting Program Bugs," pg. 12.
Fixes for bugs in some interesting Apple programs.
- Wilson, Gene, "Je M'apple, Pascal," pg. 13-21.
A group of Pascal programs: beginner's notes, using your printer, lower case for Pascal, ROM test, master catalog, etc.
- 952. Apple/Sass 2, No. 9 (December, 1980)**
- Espinosa, Chris and Wyman, Paul, "Peeks, Pokes and Calls," pg. 6-7.
A good reference for Apple programmers.
- Burger, Mike and Lynch, Ron, "Scrolling Lo-Res," pg. 9.
A routine for the Apple graphics.

\$24.95
With
diskette!



MICRO/Apple 1

This first volume in our new series contains 30 articles selected from MICRO, 1977-1980, updated by the authors or our staff. The MICRO staff has added introductory material and re-entered, listed, and tested the programs and put them on diskette (13-sector DOS 3.2 format — convertible to DOS 3.3).

Every user will want this highly practical work next to his Apple, with its chapters on BASIC Aids, Graphics, Education, Games, I/O Enhancements, Runtime Utilities, and References.

Get MICRO/Apple 1 at your local computer store.
224-page book and diskette

More Than 30 Programs on Diskette
— For Less Than \$1.00 Apiece!

No Need to Type in
Hundreds of Lines of Code!



MICRO

P.O. Box 6502
Chelmsford, MA

A cost effective 10 megabyte system from the leader in Winchester based microcomputers.

Ohio Scientific has put a low-cost, high-performance 8" non-removable hard disk together with its popular desk-top micro-computer system. This yields approximately 10 megabytes of fast hard disk capability at a tremendous cost/performance benefit over floppy based microcomputer systems.

C3-D

The 10 megabyte system is also available with the added advantage of triple microprocessors — the 6502A, 68000 and Z-80A. This allows you to make maximum use of Ohio Scientific's extensive software library as well as programs offered by independent suppliers and publishers. \$7,600.



C2-D Standard Features

- 52K RAM
- 8" floppy disk drive for program transport and backup.
- OS-65U small business operating system.
- 9-digit precision BASIC by Microsoft.
- Available in OEM quantities at attractive discounts.

For literature and the name of your local dealer,
CALL 1-800-321-6850 TOLL FREE.

OHIO SCIENTIFIC
a **MACOM** Company

1333 SOUTH CHILlicoTHE ROAD, ALIPIORA, OH 44202 • (216) 831-5600