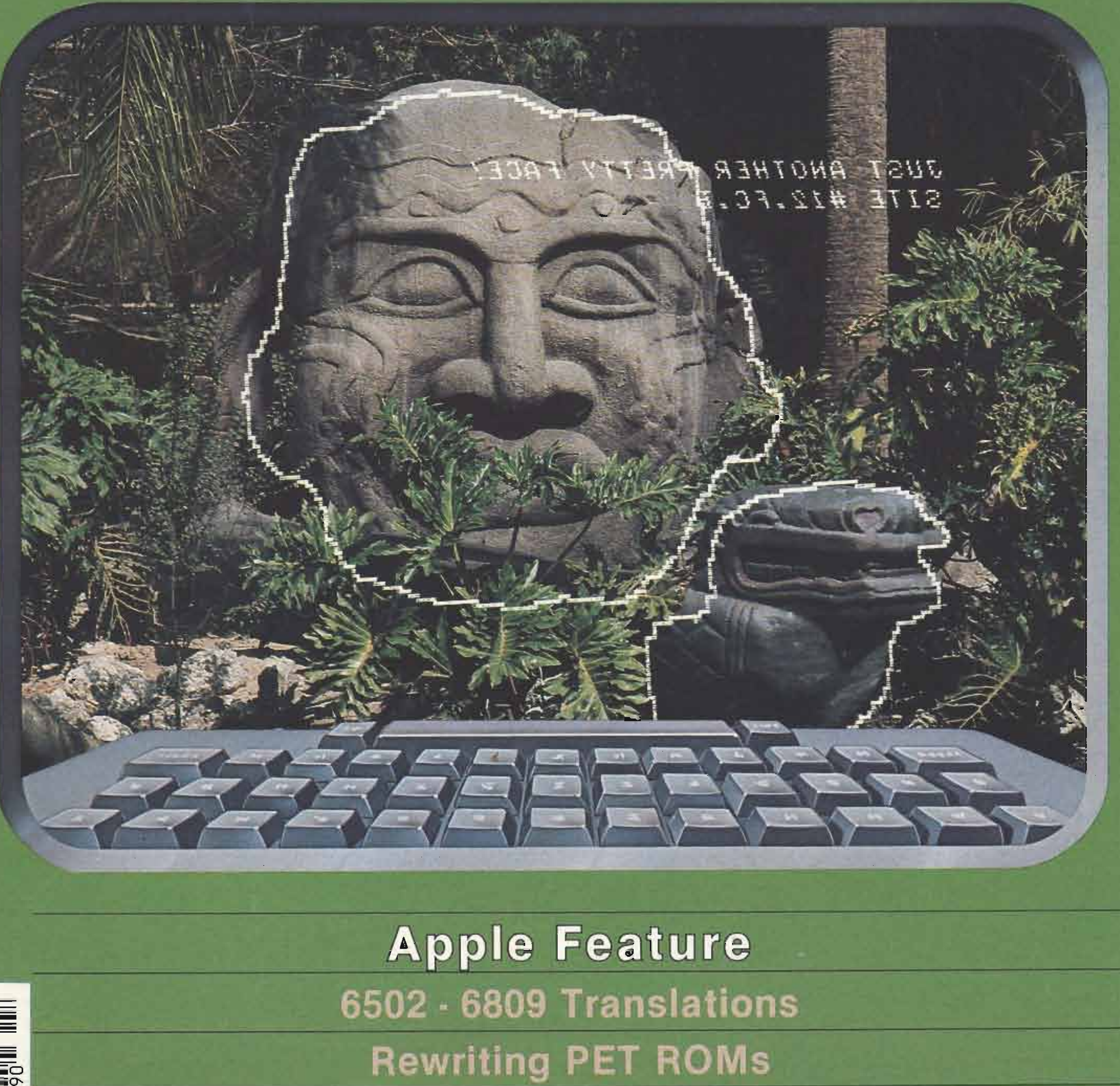
NO. 50 JULY 1982

U.S./Canada Edition: $2.50
International Edition: $2.95
United Kingdom Edition: £1.80

MICRO™

THE 6502/6809 JOURNAL



Apple Feature

6502 - 6809 Translations

Rewriting PET ROMs

AIM Tape Copy Utility

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

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

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

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

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

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

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

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

MICROSOFT
CONSUMER PRODUCTS

MICROSOFT Inc.
10700 Northup Way • Bellevue, WA 98004

2MHZ 6809 SYSTEMS

GIMIX offers you a variety to choose from!

38 MB WINCHESTER SYSTEM $17,498.99

HARDWARE FEATURES:

- ★ 2MHz 6809 CPU
- ★ 512KB Static RAM
- ★ 8 RS232C Serial Ports
- ★ 2 Parallel Ports
- ★ DMA Double Density Floppy Disk Controller
- ★ Dual 8" DSDD Floppy Disk System
- ★ Dual Winchester Subsystem with Two19 MB 5¼" Winchester Drives

SOFTWARE FEATURES:

- ★ OS-9 LEVEL TWO Multi-User Operating System
- ★ OS-9 Debugger
- ★ OS-9 Text Editor
- ★ OS-9 Assembler

19 MB WINCHESTER SYSTEM $8998.09

HARDWARE FEATURES:

- ★ 128K Static Ram
- ★ 2MHz 6809 CPU
- ★ 19 MB 5¼" Winchester DMA Subsystem
- ★ 4 RS232C Serial Ports
- ★ 1 MB 5¼" Floppy Disk Drive
- ★ DMA Double Density Floppy Disk Controller

SOFTWARE FEATURES:

- ★ OS-9 LEVEL TWO Multi-User Operating System
- ★ OS-9 Text Editor
- ★ OS-9 Debugger
- ★ OS-9 Assembler

128KB MULTI-USER SYSTEM $6997.39

HARDWARE FEATURES:

- ★ 2MHz 6809 CPU
- ★ DMA Double Density Floppy Disk Controller
- ★ 128KB Static Ram
- ★ 2 RS232C Serial Ports
- ★ Dual 8" DSDD Floppy Disk System

SOFTWARE FEATURES: Your choice of either UniFLEX or OS-9 LEVEL TWO. Both are Unix-like Multi-User/Multi-Tasking Operating Systems.

56KB FLEX/OS-9 "SWITCHING" SYSTEM $4148.49

HARDWARE FEATURES:

- ★ 2MHz 6809 CPU
- ★ 56K Static Ram
- ★ 2 RS232C Serial Ports
- ★ DMA Double Density Floppy Disk Controller
- ★ 2 Built-in 5¼" 40tr DSDD Disk Drives (80 Track DSDD Drive Option . . add $400.00)

SOFTWARE FEATURES:

- ★ GMXBUG monitor — FLEX Disk Operating System
- ★ OS-9 LEVEL ONE Multi-tasking operating system for up to 56K of memory

WINCHESTER SUBSYSTEMS

Winchester packages are available for upgrading current **GIMIX** 6809 systems equipped with DMA controllers, at least one floppy disk drive, and running FLEX, OS-9 LEVEL ONE or OS-9 LEVEL TWO. The packages include one or two 19MB (unformatted) Winchester drives, DMA Hard Disk Interface, and the appropriate software drivers. The Interface can handle two 5¼" Winchester Drives, providing Automatic Data Error Detection and Correction: up to 22 bit burst error detection and 11 bit burst error correction.

Dual drives can be used together to provide over 30 MBytes of on line storage -- or use one for back-up of the other. (More convenient and reliable than tape backup systems.

#90 includes one 19MB Drive, Interface, and Software	**$4288.90**
#91 includes two 19MB Drives, Interface, and Software	**$6688.91**

Contact GIMIX for systems customized to your needs or for more information.

50 HZ Export Versions Available

GIMIX Inc. reserves the right to change pricing and product specifications at any time without further notice.

1337 WEST 37th PLACE
CHICAGO, ILLINOIS 60609
(312) 927-5510
TWX 910-221-4055

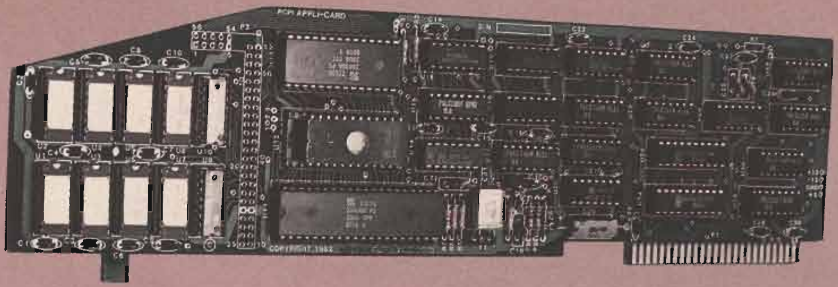
GIMIX Inc.

GIMIX® and GHOST® are registered trademarks of GIMIX Inc.
FLEX and UniFLEX are trademarks of Technical Systems Consultants Inc.
OS-9 is a trademark of Microware Inc.

NEW. . . For Apple II & III*
from **PERSONAL COMPUTER PRODUCTS, INC.**

APPLI-CARD™

4 or 6 mhz Z-80 and 64K ON-CARD MEMORY



THE ONE CARD SOLUTION TO EXECUTE CP/M* APPLICATION PROGRAMS

- Development Languages Available
- Applications Available
- Compare Our Features And Get The Best Value And Performance For Your Money.

FEATURES	Z-CARD*	SoftCard*	APPLI-CARD
6 mhz Z-80 available	No	No	Yes
64K on-card memory	No	No	Yes
CP/M available	Yes	Yes	Yes
SB-80* with card	No	No	Yes
40 col. to 255 col. horizontal scroll	No	No	Yes
Choice of application	No	No	Yes
2K PROM on the card	No	No	Yes
Real time clock available on the card	No	No	Yes
Expansion interface on the card	No	No	Yes
70 col. upper & lower case	No	No	Yes
A self-contained Z-80A or Z-80B with memory	No	No	Yes
One-card Wordstar* execution	No	No	Yes
63K available for program development or execution	No	No	Yes
Menu driven set up	No	No	Yes

Call today for pricing and product information. Dealer inquiries invited.

PERSONAL COMPUTER PRODUCTS, INC.
16776 Bernardo Center Drive,
San Diego, California 92128
(714) 485-8411 or call your local dealer

*Registered Trade Marks: Apple II & III Apple Computer, Inc. (APPLI-CARD for Apple III will be available fall 1982), CP/M® Digital Research, Inc., Z-CARD Advanced Logic Systems, SoftCard Microsoft Consumer Products, SB-80 Lifeboat Associates, Wordstar MicroPro, Inc.

NO. 50 JULY 1982

MICRO™

THE 6502/6809 JOURNAL

Answers to the June crossword puzzle can be found on page 104.

STAFF

President/Editor-in-Chief
ROBERT M. TRIPP

Publisher
MARY GRACE SMITH

Senior Editor
LAURENCE KEPPLE

Editors
MARY ANN CURTIS
MARJORIE MORSE
TIM OSBORN
LOREN WRIGHT

Production Manager
PAULA M. KRAMER

Typesetting
EMMALYN H. BENTLEY

Advertising Manager
CATHI BLAND

Promotion
MAUREEN DUBE

Circulation Manager
CAROL A. STARK

Dealer Orders
LINDA HENSDILL

Comptroller
DONNA M. TRIPP

Bookkeeper
KAY COLLINS

Advertising
Sales Representative
KEVIN B. RUSHALKO
603/547-2970

Subscription/Dealer Inquiries
(617) 256-5515

DEPARTMENTS

THE CHIEFTAIN™ 5¼-INCH WINCHESTER HARD DISK COMPUTER

SO ADVANCED IN SO MANY WAYS . . .
AND SO COST-EFFECTIVE . . .
IT OBSOLETES MOST OTHER SYSTEMS
AVAILABLE TODAY AT ANY PRICE.

● HARD DISK SYSTEM CAPACITY

The Chieftain series includes 5¼- and 8-inch Winchesters that range from 4- to 60-megabyte capacity, and higher as technology advances. All hard disk Chieftains include 64-k memory with two serial ports and DOS69D disk operating system.

● LIGHTNING ACCESS TIME

Average access time for 5¼-inch Winchesters is 70-msec, comparable to far more costly hard disk systems. That means data transfer ***ten-times faster*** than floppy disk systems.

The Chieftain Computer Systems:

Here are the Chieftain 6809-based hard disk computers that are destined to change data processing . . .

□ **CHIEFTAIN 95W4**
4-megabyte, 5¼-inch Winchester with a 360-k floppy disk drive (pictured).

□ **CHIEFTAIN 95XW4**
4-megabyte, 5¼-inch Winchester with a 750-k octo-density floppy disk drive.

□ **CHIEFTAIN 98W15**
15-megabyte, 5¼-inch Winchester with a 1-megabyte 8-inch floppy disk drive.

□ **CHIEFTAIN 9W15T20**
15-megabyte, 5¼-inch Winchester with a 20-megabyte tape streamer.

● 2-MHZ OPERATION

All Chieftains operate at 2-MHz, regardless of disk storage type or operating system used. Compare this to other hard disk systems, no matter ***how*** much they cost!

● DMA DATA TRANSFER

DMA data transfer to-and-from tape and disk is provided for optimum speed. A special design technique eliminates the necessity of halting the processor to wait for data which normally transfers at a slower speed, determined by the rotational velocity of the disk.

● RUNS UNDER DOS OR OS-9

No matter which Chieftain you select . . . 5¼- or 8-inch floppy, or 5¼- or 8-inch Winchester with tape or floppy back-up . . . they ***all*** run under DOS or OS-9 with ***no need*** to modify hardware or software.

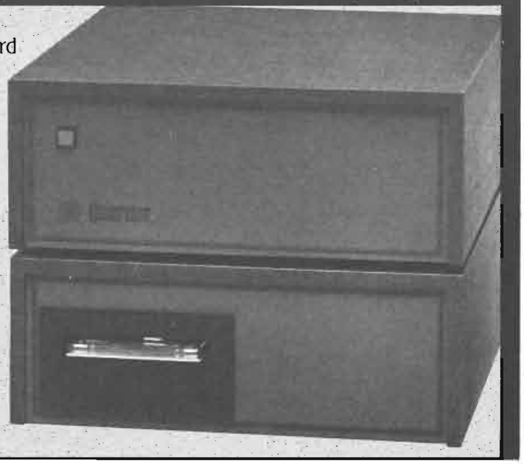
● UNBOUNDED FLEXIBILITY

You'll probably never use it, but any Chieftain hard disk system can drive up to 20 other Winchesters, and four tape drives, with a single DMA interface board!

● SMOKE SIGNAL'S HERITAGE OF EXCELLENCE

This new-generation computer is accompanied by the same ***Endurance-Certified*** quality Dealers and end-users all over the world have come to expect from Smoke Signal. And support, software selection and extremely competitive pricing are very much a part of that enviable reputation.

20-Megabyte Tape Streamer Back-Up Option



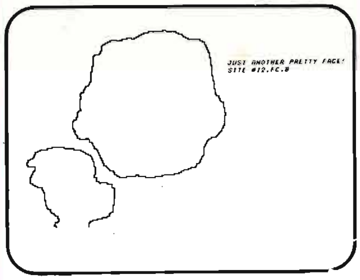
Available with all Chieftain hard disk configurations. This cartridge tape capability provides full 20-megabyte disk back-up in less than five minutes with just one command, or copy command for individual file transfers. Transfers data tape-to-disk or disk-to-tape. Floppy back-up is also available in a variety of configurations.

Write or call today for details (including the low prices) on the Chieftain Series. . . and on dealership opportunities

SMOKE SIGNAL BROADCASTING®
31336 VIA COLINAS
WESTLAKE VILLAGE, CA 91362
TEL (213) 889-9340

Name ______________________
Company ______________________
Address ______________________
City ______________ State ________ Zip ________
Telephone ( ) ______________________

About the Cover



Our cover this month sets you into a replica of an archaeological site. Today's archaeologists are able to use computers on site in their fieldwork and artifact analysis. Data collected from digs is sorted by various algorithms and organized into meaningful material. Computers are also useful for archaeological report preparation.

Cover Photo: Davey Gomes
Lowell, Massachusetts

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

Send subscriptions, change of address, USPS Form 3579, requests for back issues and all other fulfillment questions to

MICRO
34 Chelmsford Street
P.O. Box 6502
Chelmsford, MA 01824

or call
617/256-5515
Telex: 955329 TLX SRVC
800-227-1617

Subscription Rates	*Per Year*
U.S.	$24.00
	2 yr. / $42.00
Foreign surface mail	$27.00
Air mail:	
Europe	$42.00
Mexico, Central America, Middle East, North Africa, Central Africa	$48.00
South America, South Africa, Far East, Australasia, New Zealand	$72.00

MICRO™ Editorial

MICRO at the Faire

The Seventh West Coast Computer Faire at the San Francisco Civic Center gave me a chance to meet many MICRO readers. Our readers reflect the quality and range of the magazine's coverage. They are involved in intermediate to advanced projects on all of the major 6502 and 6809 machines. At this Faire, it was the Apple contingent that stole the show. Not the Apple itself; it has been around too long for headlines. Not even the nearby Apple corporation attracted exceptional attention; those folks are playing it cool in the face of demands for new product announcements that they are not yet ready to make. It was the Apple users themselves who made the show an event. Thousands of them packed the aisles, pursuing with enthusiasm their ongoing relationship with the Apple microcomputer.

The Apple II succeeded so well because its design encouraged the largest possible amount of outside support. Software developers were able to benefit from good system documentation. And hardware developers found that the Apple created a large market for add-on boards. Instead of discouraging users from looking inside the computer, the Apple's removable cover made it easy for people to see how "the computer" was in fact a modular system. Since the original Apple II did have some shortcomings (40-column video display, for example), Apple users became a receptive market for boards (such as the 80-column board) that improve performance and are easily installed. This upgradability has made it possible for the Apple to reach markets (such as small business) that might never have been interested in the original machine.

The Apple's open design has also made it less susceptible to obsolescence. Although the Apple's CPU, the 6502, has been a powerful and flexible success, the fact remains that it was designed more as a dedicated than as a general-purpose processor. Now that more powerful processors are available, Apple owners can still take advantage of them by plugging in new boards. For example, Stellation II's The Mill and ESD Labs' Excel-9 give Apple owners access to the 6809 and its powerful operating systems FLEX and OS-9. Such products enable the Apple to handle applications that could not have been imagined as possible for the system at the time of its introduction.

Sixteen-bit processors (8088, 68000) are also becoming available as add-ons to the Apple. Apple users can gain significant educational advantages by running these new processors on their existing machines. However, the basic Apple architecture can only stretch so far. The full power of a processor like the 68000 can only be realized in an environment that has been created especially for it. At the Faire, two such 68000-based systems vied for attention: the Fortune 32:16 and the SAGE II. The Fortune system is directed at the end-user market. It runs the UNIX operating system and offers a solid range of basic applications packages. The SAGE II system is being marketed to OEMs as one of the fastest 68000 software development systems currently available. According to company president Rod Coleman, it runs at 8 megahertz and can compile 1800 lines of code per minute.

Machines like the Fortune 32:16 and the SAGE II show how much progress has been made since the Apple was introduced. The minicomputer market is being seriously challenged at the microcomputer level. But only a few of us can participate — now — in the development of the latest 68000 systems. Many thousands of people are participating in the use and development of the Apple, and that is the real significance of the Apple computer. No longer impressive from the hardware point of view, the Apple has achieved an outstanding level of support. The Seventh West Coast Computer Faire was not, by and large, notable for its new product announcements. The most remarkable exhibit was the Apple community's collective display of individual effort.

Laurence Kepple

MICRO™ Letterbox

Dear Editor:

I read with interest the editorial in the March 1982 issue of MICRO entitled "Hello, OSI?". In response, I offer the following statement:

"M/A COM Office Systems, Inc., formerly known as Ohio Scientific, fully intends to continue its presence in the personal computer market and to support our customer base. Our new management also supports the company's traditional leadership position in microcomputer technology and makes a firm commitment to maintain that position in the growing market for personal and small business computers."

Forthcoming product announcements will exemplify this commitment.

Philip M. Johnson
Corporate Communications Manager
M/A COM Office Systems, Inc.
7 Oak Park
Bedford, MA 01730

Dear Editor:

I have noted several significant changes in MICRO in the last several months. Some of these are good, but some are quite disappointing.

Please don't neglect your long-time readership — the people who started reading MICRO to better learn programming in assembler and the hardware of the computers they own.

I realize that many of your readers are Apple owners. I am also, but there are several other fine publications dedicated to the Applesoft programmer. Stay with what historically you have done so well.

I am also disappointed with the reduction in the size of the 6502 Bibliography. This alone was worth the price of the magazine.

Hardware construction articles are applicable to several different machines including the single boards. I built the clock described several months ago in MICRO (37:99) and would like to see similar articles.

Alan P. Wilson
415 N. Indiana St.
Salem, IL 62881

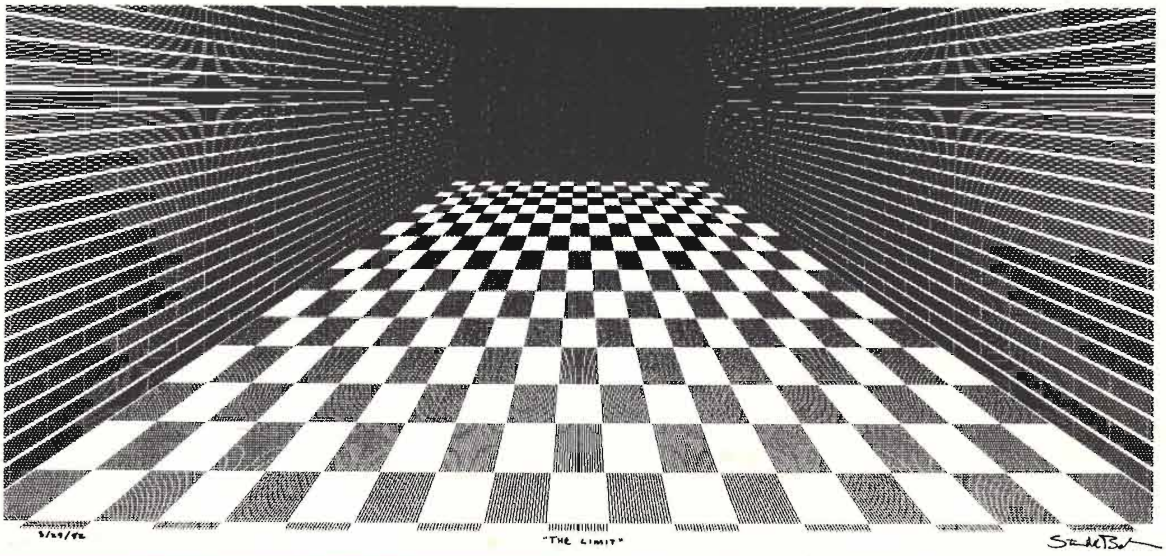
Editor's note: The 6502 Bibliography now includes only the most significant and useful articles selected from a variety of publications in the month covered.

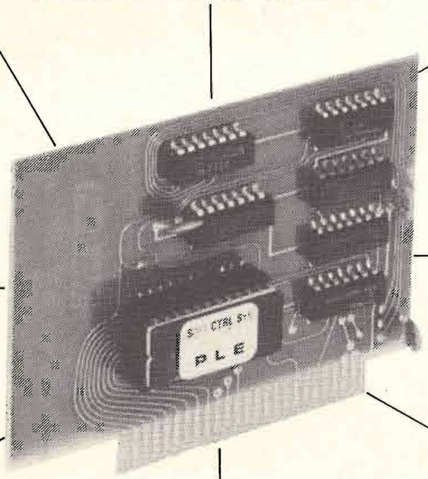
Dear Editor:

I am preparing a book on the use of Microcomputer Programs in Medicine. I would appreciate hearing from your readers interested in having their programs included in this book. The programs will include file organization of medical records, data extraction, file statistics and general statistics used in medical research, graphic plotting of research data, patient history taking and history summarization, patient scheduling, and billing routines. I plan to publish the programs for the Apple II, Commodore PET, and the TRS-80.

Derek Enlander, M.D.
Dept. of Nuclear Medicine
New York University Medical Center
560 First Avenue
New York, NY 10016

Stephen M. Boker of Data Transforms, Inc. sent MICRO this sample of a graphic technique he calls "ultra-res." Approaching the limit!





SOFT CTRL SYS
PLE

Skyles Electric Works

Epson-PET/CBM Graphic ROM Pack

For PET/CBM Owners Who Want:

Complete Program Listing Printouts
Complete Screen Graphic Printouts
Graphic Printouts From Programs

on your Epson Printer

Order the Skyles Electric Works EPSON-PET GRAPHIC ROM Package. The ROM when installed in an EPSON MX80, MX80FT, or MX100 with Graftrax Plus printer will reproduce most of the PET/CBM graphics characters. Most importantly when using the accompanying high speed machine language program, the Epson-PET Graphic ROM pack gives a complete program listing with all screen controls shown (cursor, home, clear, etc.). This high speed machine language program for program listing, screen image printout (screen dumps) and BASIC program controlled printing (i.e. PRINT) automatically translates the PET-ASCII characters to the EPSON-GRAPHIC ROM characters. A BASIC sample program and PRINT subroutine, that may be incorporated into any existing BASIC program, completes this "complete solution" package.

EPSON MODEL	ROM MODEL
MX70	Not Available
MX80 (serial no. to 359999)	EPG80 (EPG82, 3 ROM Version)
MX80 (serial no. after 360000)	EPG81 (EPG83, 3 ROM Version)
MX80FT	EPG8F
MX80 Graftrax	Not Available
MX80 Graftrax Plus	EPG8G+
MX80FT Graftrax Plus	EPG8G+
MX100	Not Available
MX100 Graftrax Plus	EPG10G+

The Epson-PET Graphics ROM Pack has been designed to furnish you with PET/CBM graphics printing in the easiest way possible. This is done by furnishing a high speed machine language program that is "hidden" at the top of your PET/CBM memory.

The machine language program serves 3 major functions.

1: Translates PET-ASCII code to ASCII code for program listing.

2: Translates screen code to ASCII code for screen image printouts.

3: Translates PET-ASCII code strings to ASCII strings for normal program printout. This feature may also be used for making ASCII files for your disk or tape recorder.

INSTALLATION: **Installs into your Epson printer**

PRICE: **Epson-PET Graphics ROM Pack EPG80, EPG81, EPG8F.....$75.00**

Please specify your Epson printer model type and and serial number when ordering.

For all PET/CBM's BASIC 2.0/Revision 3, or BASIC 4.0

AVAILABILITY: Immediately from your LOCAL DEALER

or

VISA, MASTERCHARGE ORDERS CALL (800) 227-9998 (except California residents)
CALIFORNIA ORDERS PLEASE CALL (415) 965-1735

Skyles Electric Works

231E South Whisman Road
Mountain View, CA 94041
(415) 965-1735

Rewriting PET ROMs

by James Yost

By replacing one or more of your PET's ROMs with a reprogrammed EPROM, you can correct minor bugs or add whole new capabilities.

Rewriting requires:

PET — any operating system
EPROM programmer
2716 or 2532 EPROM
Possibly some electronic components

You've undoubtedly been annoyed when your PET/CBM gives you an error message after you've opened a file that has already been opened. The operating system not only claims an error, it closes the file. And you must have noticed that when doing a machine-language save from the monitor, a title just over some arbitrary length causes a "?" to be printed. You must shorten the title and remove the "?". Such frustrations provide strong motivation for rewriting the PET's operating system, and this article will tell you how to go about it. The above-mentioned annoyances can be eliminated by changing just three bytes in one ROM; other changes of varying complexity will be suggested. To make these changes you should have a 6502 disassembler, such as in the extended monitors (SuperMon, MicroMon, ExtraMon, etc.), and an interest in learning machine language.

The title in tape headers could be used for more extensive documentation if the 16-character limit were increased to a value determined by the 80-character command limit. It turns out that there is no reason for limiting titles to 16 characters, so let's see what we can do. The tape buffer will hold 187 characters of title, but the 80-character command lines limit the savable title length to 77 characters in BASIC and 64 characters in the monitor. When printing "FOUND...", however, we probably won't need to take up a third line for the overflow from the first two lines, so the value shown in table 1 is chosen to print 73 characters and keep the title on two lines. The ML save limitation value is simply changed in the error-checking code. When a file to be opened is determined to be already open, a branch is made to code that closes all files and then prints the "FILE OPEN ERROR" message. The value shown in table 1 changes the branch instruction to start *after* the subroutine that closes all files. For information on making these changes in your machine, see accompanying boxes.

It is possible to change the reset code so that it does not wipe out memory. You could change the code at $E10C-$E134(1.0)/$E158-$E173(3.0)/$D3F0-$D416(4.0) to simply store your top-of-memory value instead of overwriting memory to test it. The ROM memory test routine doesn't detect the most common types of failure, anyway. This change is slightly more involved than the last ones, and is a good exercise if you're learning machine language.

A repeat-key routine that is built into the normal interrupt routine would be very handy, as it wouldn't interfere with LOAD and SAVE operations the way add-on utilities do. Finding a place to put the code is a little problem in the 1.0 and 3.0 ROMs, but if you are cutting out the memory check on power-up, curtailing the power-up message will give you sufficient space. Power-up reset code starts at $E0D2(1.0)/$D3B6(4.0), and the power-up message is at $E174-$E19A(1.0/$E1B7-$E1DD (3.0)/$DEA4-$DEC1(4.0). The 4.0 ROMs have plenty of available space — look for runs of aa's. Over 1700 bytes are available in ROM address space if you make your own EPROMs with your own code added. This compensates for the loss of the $B000 socket for 4.0. This project is considerably more complex than the others, and is recommended only for those who have experience in machine language.

Table 1: **Simple Changes to PET ROMs**

1.0	3.0	4.0	Was	Now	Change
—	FF42	D6A8	$10	$49	M.L. save title length
F5D8	F5D0	F60F	$15	$4E	"FOUND " title length
—	F52E	F56D	$41	$44	OPENing open files doesn't close them

Table 2: **Locations to Change for Automatic Linking of Auxiliary Chips**

	1.0	3.0	4.0
CHRGET	E0B5-E0CC	E0F9-E110	D399-D3B0
Initial IRQ	E1F4-E1FD	E1EC-E1F3	E00E-E013
Also IRQ	FD34,FD35	FD0D,FD0E	FD58,FD59

***Listing 1:* ROM Changes to Dump Screenful with Monitor 'M' Command**

```
                        3.0
FE63   20   CF   FF   C9   0D   D0   03   4C
FE6B   B1   FF   20   A7   E7   90   22   20
FE73   97   E7   20   01   F3   F0   17   A6
FE7B   DE   D0   13   38   4C   ED   FF

FE90   E4


FFB1   A5   FD   69   B7   85   FB   A5   FE
FFB9   69   00   85   FC   4C   72   FE
FFED   A5   FD   E5   FB   A5   FE   E5   FC
FFF5   90   B7   4C   82   FE


                        4.0
D5C7   4C   C2   DE

DEC2   20   CF   FF   C9   0D   F0   03   4C
DECA   CA   D5   A5   FD   69   B7   85   FB
DED2   A5   FE   69   00   85   FC   4C   CF
DEDA   D5
```

***Listing 2:* Modification for LIST Only On Key Down**

```
                        1.0
C611   4C   29   CC

CC28   00   A8   AD   12   E8   C9   FF   F0
CC30   F9   B1   AE   4C   14   C6


                        3.0
C620   4C   13   CC
CC12   00   A8   AD   12   E8   C9   FF   F0
CC1A   F9   B1   5C   4C   23   C6


                        4.0
B69B   4C   0D   BD

BD0C   00   A8   AD   12   E8   C9   FF   F0
BD14   F9   B1   5C   4C   9E   B6
```

To prevent my work from scrolling off the top of the screen while I examine memory, I developed a ready-made modification for the monitor. Listing 1 shows the changes needed to dump a screenful of memory when the M command is used with only one address. If this can be accomplished so easily, imagine how many other significant improvements you might make.

Wouldn't it be nice if a BASIC listing scrolled by only when you held the space bar down? The LIST code is located at $C5A8-$C648 (1.0)/$C5B5-$C657 (3.0)/$B630-$B6DD (4.0) and the changes are shown in listing 2. On 40-column graphics keyboards, holding down the space bar will scroll the listing quickly and the RVS/OFF key will scroll it slowly. Releasing the key will immediately halt the scrolling, and pressing the RUN/STOP key stops the listing, restoring full keyboard operation. On business keyboards, other keys, such as the left arrow and the colon, may be required instead. This comes at the expense of dropping "FROM START" from the "?REDO FROM START" message. The extra bytes check the keyboard VIA and continue to loop until the VIA shows a key depressed. If you want to reverse scroll, you will need quite a bit of extra code. A utility chip might be your best bet.

If you regularly use one of the auxiliary chips that adds commands to BASIC or the monitor, you can rewrite the ROMs to automatically link the chip on power-up. The CHRGET routine that is transferred to low memory

Making Your Own ROMs

You've probably heard about EPROMs (Erasable Programmable Read-Only Memories), but you may not realize how easy it is to use them. They are erasable, so your mistakes won't have drastic consequences, and you can make revisions easily. Erasure is by ultraviolet light, but don't expect to get by cheaply using sunlight — it takes too long and is unreliable. A GE #G8T5 germicidal lamp will do the job in 20 or 30 minutes if the chips are about one inch from the lamp. A high-intensity UV source like this can damage your eyes, so be careful if you build your own eraser. An 8-track plastic tape storage box is a good size for the lamp hardware. You might decide to pay $40 and up to get a ready-made one, but you will get only polish and convenience for spending more than that; erasure isn't any faster or better.

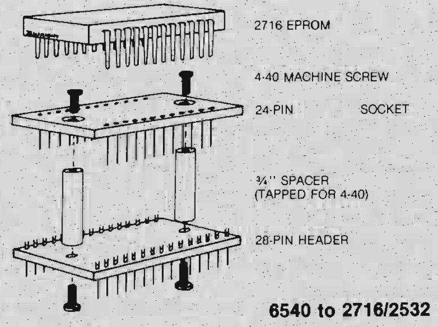
The best value available is to "burn" your own EPROMs with the Branding Iron available from AB Computers (252 Bethlehem Pike, Colmar, PA 18915). It programs single-voltage 2716's (2K) and 2532's (4K) that plug directly into the newer PET/CBM boards with 24-pin sockets. If yours has 28-pin ROM sockets, see accompanying box for owners of old PETs. The software that comes with the programmer is a model of efficiency and convenience.

To program, plug an EPROM in the socket provided and flip the program switch on. If the chip gets hot, check to see if you have inserted it correctly in the socket. Load the machine-language code supplied (occupies $1800-$2000), type SYS6144, and the software hooks into the monitor to provide extra programming commands. If you have code at $2000 to put on a 2716, type .P 2000 27FF. The EPROM will be checked for erasure, and any programmed locations will be printed in reverse field. The programming then proceeds for a couple of minutes showing a running address on the screen. Each byte is checked for success, and failures are printed out. A really stuck bit (happened once) will show up here. When programming is completed, it is checked again, and a new dot prompt appears. The contents may be verified with .V 2000 27FF, which compares memory locations with bytes in the EPROM addressed by the last three hex digits (addressing on 2K (2716)/4K (2532) boundaries). Mismatches are printed out in reverse field. To view the contents of an EPROM, .C 2000 27FF must be used to copy into memory, where the M command can access it. Typing .T 32 enables 2532 operation. Except for programming, these commands are virtually instantaneous. A Zero Insertion Force (ZIF) socket with clamp/release lever will plug in and make EPROM changing much simpler.

on power-up resides in the locations specified in table 2. Simply note the changes in $70 to $87 ($C2-D9 in 1.0) and replace them in the ROM. If your chip has repeat keys, or otherwise changes the IRQ vector in $90, $91 ($0219, $021A in 1.0), that vector needs to be changed in the locations shown in table 2.

These changes are the result of a long process of learning about machine language, primarily by figuring out how other programs work. I started with little programs in the second cassette buffer and worked up to the PET's large operating system. My primary tool was a disassembler program that converts byte values to mnemonics. A SWTPC printer that prints 40 columns on 3½-inch adding machine paper was very valuable. In addition to being economical, the paper is ideally sized for disassembly, and can be accordion-folded so that an entire operating system is available at your fingertips. If you would like to personalize your PET, I encourage you to dig into it and make it work your way.

The author may be contacted at P.O. Box 556, Somerville, MA 02143.



6540 to 2716/2532 Pin Correspondences

For Owners of Old PETs Only

Just try to find a 28-pin EPROM to match the 6540 ROM! A little engineering can bridge the gap. All the signals needed for the 2716 are available at the 28-pin socket in one form or another. First the easy ones: the address and data lines are exactly the same — just on different pins. A given 28-pin 6540 ROM chip is connected to the data lines by a 4K select signal to the Pin 3 Chip Select. That will Output Enable both the 2716 and 2532 on Pin 20. However, the 6540 and 2716 occupy only half of the 4K address space, and address bit 11 (A11) is low for the first half of this space and high for the rest. Therefore, A11 is supplied to lower-half 6540's on CS3 (Pin 4, a Chip Select requiring lows), and to upper-half 6540's on CS1 (Pin 17, a Chip Select pin requiring highs). To replace lower-half 6540's with 2716's, or lower and upper with 2532's, connect 6540 Pin 4 (A11) to 2716/2532 Pin 18. This will enable a 2716 for lower-half addresses, and provide A11 to the proper pin for 2532's. To replace upper-half 6540's with 2716's, A11 is available at 6540 Pin 17 and needs to go through an inverter to provide 2716 Pin 18 with the low for upper-half addresses that we need. This inverter can be a 7404 that derives power from 6540 Pins 1 and 12. The +5 volt Vcc and ground connect to their proper pins, of course. Not too bad, all in all; a series of wires running from various pins of the 28-pin socket to appropriate different pins on a 24-pin socket, and possibly an inverter IC somewhere. The table shows these pin connections.

I use a sturdy 24-pin socket screwed to a 28-pin header using ¾-inch tapped spacers for 28-pin conversion (see figure A). This creates a solid unit that is easy to wire and use. Drill two holes in the solid bottom of a 24-pin socket and matching holes in a 28-pin header. For spacer stock I obtained plastic tubing of a size suitable for 4-40 tapping and sawed off pieces. A 6-foot length costs less than a dollar. Ready-tapped aluminum spacers may also be used.

Wires should be soldered between pins according to the table in figure 1. On 28-pin boards, the ROMs are in C0, D0, E0, F0, C8, D8, F8 order beginning at the right-hand end (H-1). A 2716 would always be used to replace E0, as the upper half of the E 4K space is used by the VIAs. A 2532 in the C0, D0, F0 sockets with Pin 18 (A11) connected to 6540 Pin 4 will eliminate the need for an inverter. For a 2716 in a C8, D8, or F8 socket, 6540 Pin 17 supplies 2716 Pin 18 through an inverter. Otherwise 6540 Pin 4 connects directly to 2716/ 2532 Pin 18.

6540 to 2716/2532 Pin Correspondence

6540 name	6540 pin	2716 pin	2716 name	2532 name
gnd	1	12	gnd	
CS5	2			
CS4	3	20	OE	
CS3	4	(18)	CE	A11
A0	5	8	A0	
A1	6	7	A1	
A2	7	6	A2	
A3	8	5	A3	
A4	9	4	A4	
A5	10	3	A5	
A9	11	22	A9	
V_{cc}	12	21	V_{pp}	
V_{cc}	12	24	V	
A8	13	23	A8	
A7	14	1	A7	
A6	15	2	A6	
2	16			
CS1	17 (inv)	(18)	CE	
A10	18	19	A10	
DB7	19	17	07	
DB6	20	16	06	
DB5	21	15	05	
BD4	22	14	04	
DB3	23	13	03	
DB2	24	11	02	
DB1	25	10	01	
DB0	26	9	00	
CS2	27			
nc	28			

MICRO

ROCKWELL Microcomputers from Excert, Inc.

THE AIM 65/40
Single Board or Smorgasbord

- A full size terminal style keyboard w/8 special function keys
- A smart, 40 character display with its own microprocessor
- A 40 column printer w/text and graphic output
- Up to 64K of on-board RAM and ROM
- On-board interfaces include RS232, dual audio cassette and 2 user I/O R6522 devices
- Firmware includes interactive monitor and text editor w/options of Assembler, BASIC, FORTH and PL/65

THE AIM 65
Take-Out Order

- A full size terminal style keyboard w/3 special function keys
- A 20 character display
- A 20 column printer w/text and graphic output capability
- Up to 4K RAM and 20K ROM on-board
- On-board interfaces include 20MA TTY, dual audio cassette and 1 user I/O R6522 device
- Firmware includes interactive monitor and text editor w/options of Assembler, BASIC, FORTH, PASCAL, & PL/65

And if the above isn't enough,
Try the RM65 — a product line filled with embellishments including:

32K DRAM Board
CRT Controller
Floppy Disk Controller
PROM Programmer

ACIA Board
IEEE-488Board
CPU/SBC Board
4-16 Slot Card Cages

Prototype cards
Adaptor Buffer Modules
General Purpose I/O Board
PROM/ROM Board

NEW LOWER PRICES AND A CASH DISCOUNT* TO BOOT!

A65/40-16 (16K RAM)	$1225
A65/40-32 (32K RAM)	$1295
A65/40-A (Assembler)	$ 85
A65/40-B (BASIC)	$ 65

A65-1 (1K RAM)	$420
A65-4 (4K RAM)	$445
A65-4B (4K RAM w/BASIC)	$495
A65-PS (PASCAL)	$100
A65-F (FORTH)	$ 65
A65-A (Assembler)	$ 35

Mail Order to:

Educational Computer Division
EXCERT INCORPORATED

- SALES
- SERVICE
- INSTALLATION
- CONSULTING

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

Higher quantities quoted upon request, COD's accepted, shipping will be added. *Deduct 5% cash discount on prepaid orders. Minnesota residents add 5% sales tax. Prices subject to change without notice.

Timing and Counting with the 6522

by Marvin L. De Jong

This article describes techniques to use the 6522 Versatile Interface Adapter for practical timing and counting tasks. Included are programming examples and application suggestions.

requires:

Any system with a 6522 Versatile Interface Adapter (VIA)

Programming examples for Apple II with John Bell 6522 board in slot #7.

Once it is understood, the 6522 certainly lives up to its name "Versatile Interface Adapter." The SYM-1 and the AIM 65 both come equipped with at least one 6522. The Apple II is easily equipped with a 6522, once the peculiarities in the Apple II timing design are understood.[1, 2] (Perhaps the easiest way to interface a 6522 to an Apple II is to purchase a 6522 board from John Bell Engineering, P.O. Box 338, Redwood City, CA 94064.) The 6522 is an important device in many real-time control applications and it is very useful in handling data acquisition tasks in the laboratory.

The purpose of this article is to describe some simple 6502 assembly language programs to be used in conjunction with a 6522 VIA (Versatile Interface Adapter) to perform precision timing and/or counting tasks. The techniques described require the simplest possible hardware accessories. In fact, *no additional hardware* is required if your laboratory instrumentation produces TTL-level pulses; a single connection to the PB6 pin on the 6522 will suffice. I also offer some simple examples that illustrate how the programs may be used to make measurements of time, temperature, velocity, etc.

Typically, we think of time measurements in terms of the time between *two events*; the start and end of a race, for example, or the arrival of successive cosmic rays. In this case it is assumed that suitable transducers, such as phototransistors, mark the events with a logic-zero pulse as illustrated in figure 1. Figure 2 shows a simple circuit that produces a logic-zero pulse when a light flashes on either of the two phototransistors. Many other schemes exist for signaling the starting and ending events, but the scheme in figure 2 simply illustrates the time-measurement concept.

The timing program described below will make the measurement of

Figure 1: **Timing diagram for measuring the time, T, between two events.**

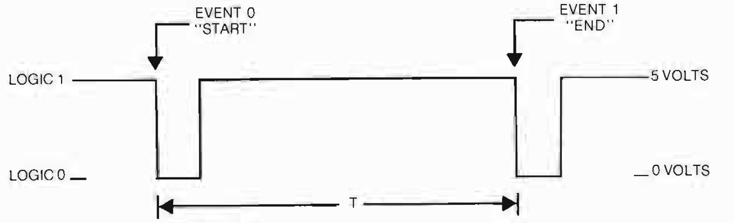


Figure 2: **A simple phototransistor circuit: light striking either phototransistor will produce a logic-zero output voltage. A flash of light will produce a pulse.**

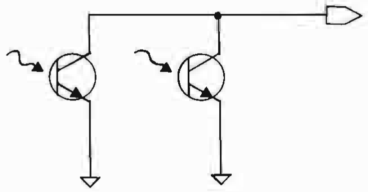
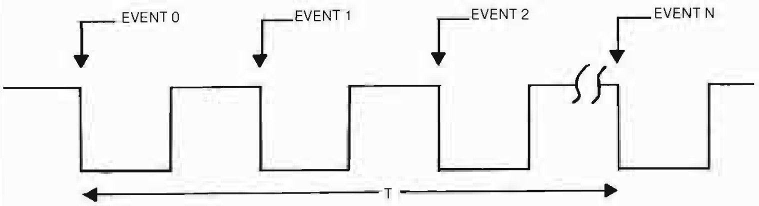


Figure 3: **Timing diagram that illustrates N pulses occurring in time T.**



the time interval, T, between two successive event pulses as illustrated in figure 1. However, it is also worthwhile to be able to measure the time required for *N events* to occur. For example, to measure the *frequency* of a periodic pulse train it is sufficient to measure the time, T, it takes for N pulses to occur. The frequency of the periodic pulse train is given by the formula

$$f = N/T$$

where N is the number of pulses that have occurred in time T. Refer to figure 3 for an illustration of such a pulse train. Note that the waveform need not be symmetrical (50% duty cycle), nor must the waveform be periodic. The time for N randomly spaced pulses can also be measured. Thus, the pulse-rate from a pulse train produced by a radioactive decay experiment can also be measured.

Since the 6522 VIA can perform both timing and counting functions, it is ideally suited to make the measurements just described. The T2 counter/timer is used to count the N pulses, and the T1 timer is used to measure the time interval, T, in which these N pulses occur. I digress for a moment to introduce the registers of the 6522 that will be used to make the measurements.

6522 Control and Flag Registers

For the moment our concern will be with three of the 6522's sixteen registers. These are the Auxiliary Control Register (ACR), the Interrupt Flag Register (IFR), and the Interrupt Enable Register (IER). The ACR controls the behavior of both the T1 timer and the T2 counter/timer. It is a control register. That is, setting or clearing bits in this register determines how the various control pins and timers of the 6522 are going to function. A diagram of this register is shown in figure 4. This diagram indicates the control function of each bit. Note that bits five, six, and seven control the behavior of the T1 timer and the T2 counter/timer.

To be specific about the memory location of this register, we must think in terms of a specific machine. Assume we are using VIA #2 on the SYM-1. Then the ACR has the address $A80B. T2 must be in its pulse-counting mode, so bit five (ACR5) must be one. The T1 timer will be used in its free-running mode (without toggling Pin PB7) so bits six and seven must be loaded with a one and a zero, respectively. That is, ACR6 will be one and ACR7 will be zero. All of this can be accomplished by loading the ACR with $60; that is, with LDA #$60 and STA $A80B instructions.

The IFR (Interrupt Flag Register) is used to "watch" either T1 or T2 to see if either has counted through zero. A diagram of this register is shown in figure 5. Note that bit five of the IFR (IFR5) is set when T2 counts through zero, while IFR6 is set when T1 counts through zero. Of course, the counting rate of T1 is determined by the system clock rate, which is typically one MHz.

If we assume that the 6522 being used is the VIA #2 on the SYM-1, then the address of the IFR register is $A80D. These flags are cleared by reading or writing to their corresponding timer locations, to be described below.

Finally, T1 will be used to create evenly spaced interrupts on the IRQ line of the microcomputer system. Therefore, the 6522 must be functioning so that when T1 times out it will produce an IRQ-type interrupt. This is

***Figure 4:* A Diagram of the Auxiliary Control Register (ACR)**

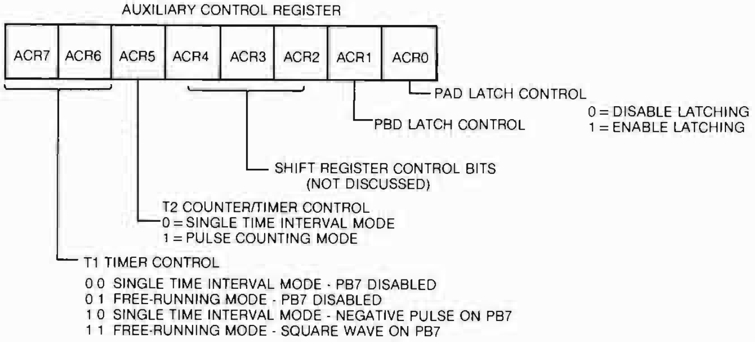

AUXILIARY CONTROL REGISTER
ACR7
ACR6
ACR5
ACR4
ACR3
ACR2
ACR1
ACR0
PAD LATCH CONTROL
0 = DISABLE LATCHING
1 = ENABLE LATCHING
PBD LATCH CONTROL
SHIFT REGISTER CONTROL BITS
(NOT DISCUSSED)
T2 COUNTER/TIMER CONTROL
0 = SINGLE TIME INTERVAL MODE
1 = PULSE COUNTING MODE
T1 TIMER CONTROL
0 0 SINGLE TIME INTERVAL MODE - PB7 DISABLED
0 1 FREE-RUNNING MODE - PB7 DISABLED
1 0 SINGLE TIME INTERVAL MODE - NEGATIVE PULSE ON PB7
1 1 FREE-RUNNING MODE - SQUARE WAVE ON PB7

***Figure 5:* A diagram of the 6522's Interrupt Flag Register (IFR)**

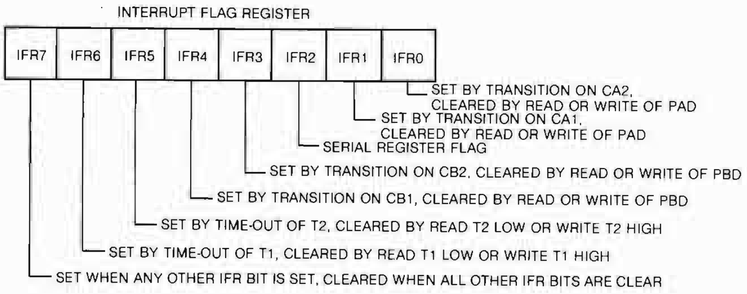
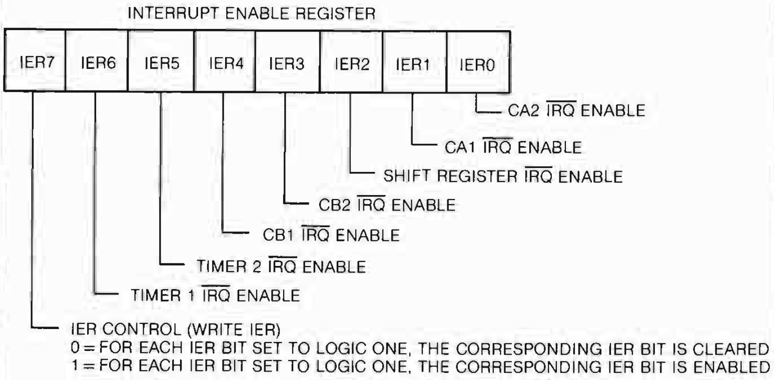

INTERRUPT FLAG REGISTER
IFR7
IFR6
IFR5
IFR4
IFR3
IFR2
IFR1
IFR0
SET BY TRANSITION ON CA2, CLEARED BY READ OR WRITE OF PAD
SET BY TRANSITION ON CA1, CLEARED BY READ OR WRITE OF PAD
SERIAL REGISTER FLAG
SET BY TRANSITION ON CB2, CLEARED BY READ OR WRITE OF PBD
SET BY TRANSITION ON CB1, CLEARED BY READ OR WRITE OF PBD
SET BY TIME-OUT OF T2, CLEARED BY READ T2 LOW OR WRITE T2 HIGH
SET BY TIME-OUT OF T1, CLEARED BY READ T1 LOW OR WRITE T1 HIGH
SET WHEN ANY OTHER IFR BIT IS SET, CLEARED WHEN ALL OTHER IFR BITS ARE CLEAR

***Figure 6:* A diagram of the 6522's Interrupt Enable Register (IER).**


INTERRUPT ENABLE REGISTER
IER7
IER6
IER5
IER4
IER3
IER2
IER1
IER0
CA2 $\overline{\text{IRQ}}$ ENABLE
CA1 $\overline{\text{IRQ}}$ ENABLE
SHIFT REGISTER $\overline{\text{IRQ}}$ ENABLE
CB2 $\overline{\text{IRQ}}$ ENABLE
CB1 $\overline{\text{IRQ}}$ ENABLE
TIMER 2 $\overline{\text{IRQ}}$ ENABLE
TIMER 1 $\overline{\text{IRQ}}$ ENABLE
IER CONTROL (WRITE IER)
0 = FOR EACH IER BIT SET TO LOGIC ONE, THE CORRESPONDING IER BIT IS CLEARED
1 = FOR EACH IER BIT SET TO LOGIC ONE, THE CORRESPONDING IER BIT IS ENABLED

accomplished with the IER (Interrupt Enable Register) diagrammed in figure 6. To enable interrupts from T1, IER6 must be set by loading it with a one. The instructions LDA #$C0 and STA $A80E will enable interrupts from T1 if the SYM-1 is being used. Note that bit seven (ACR7) must be set to logic one when writing to the IER if a particular interrupt is to be enabled.

6522 Timing and Counting Registers

The timing and counting registers are numbered four through nine. They occupy locations $A804 through $A809 on the SYM-1, VIA #2. Both T1 and T2 are 16-bit devices. Typically the low-order byte is stored first and timing or counting commences when the high-order byte is stored in the appropriate register. For example, if T2 must be set up to count 256 ($FF + 1) pulses, then its low-order latch at location $A808 is loaded with 255 ($FF) and its high-order counter is loaded with $00, in that order. Observe that the number loaded into the 16-bit register (two 8-bit registers) is one less than the number of pulses to be counted, sincc the 6522 must count through zero in order to get the T2 flag (IFR5).

In the programs that follow, the T1 timer latches will be loaded with $FFFE to produce evenly spaced interrupts every 65536 clock cycles. Observe that the number of clock cycles between interrupts is two more than the 16-bit number loaded into the two 8-bit latches of the T1 timer. Thus, location $A806 will be loaded with the number $FE and, when timing is to commence, location $A805 will be loaded with $FF. Finally, in order to clear the interrupt from T1, its low-order counter must be read. This is accomplished by reading location $A804, and this instruction will be part of the interrupt routine.

(For additional details regarding the timers, see references three and four at the end of this paper.)

The interrupt routine is used to increment a two-byte interrupt counter consisting of two locations in page zero of memory. Since interrupts occur every 65536 clock cycles, the two-byte interrupt counter starts to function when the time, T, exceeds 65536 clock cycles. The two-byte interrupt counter keeps track of the number of 65536 clock-cycle time intervals in T, while the 16-bit counter register in T1 measures the number of clock cycles in T up to 65535. The longest time interval, T, that can be measured is approximately one hour. This maximum time is easily increased by making the two-byte interrupt counter a three-byte interrupt counter.

What will be the function of the T2 counter? It must first detect the zeroth (or starting) event. To do this, T2 is initially loaded with $0000. The first logic-zero pulse will set IFR5, the bit in the IFR that is set when T2 counts through zero. When that event is detected, the T1 timer is started. Next, T2 is loaded with the number of events to be timed, less one since it must count through zero. For example, if timing is to cease with the next logic-zero pulse, then T2 is loaded with zero corresponding to one event. If 10,000 pulses are to be counted, then T2 is loaded with 9,999. When T2 counts through zero the second time, the T1 timer is then read.

Since T1 counts down from $FFFE, the number in T1 must be subtracted from $FFFE to give the correct number of clock cycles. This two-byte result

and the number in the two-byte interrupt counter form a four-byte number representing the number of clock cycles between the zeroth event and the Nth event. Refer again to figures 1 and 3. Minor complications of these basic ideas will be discussed later. However, the basic concept is that T1 measures the time, T, for the N events counted by T2 to occur. In certain applications, the quantity T will be the one desired. In other applications, the frequency f, where f = N/T, will be the desired quantity. Thus, the program is capable of measuring either time or frequency.

The Timing Program

The program to measure the time, T, for N events is given in listing 1. This listing assumes that a SYM-1 microcomputer is being used; that is, the addresses of the 6522 registers are identical to those of VIA #2 on the SYM-1. If you have an AIM 65, then you will have to drop the "8" in all the 6522 addresses, and the same program will work. If you have an AIM 65, be sure to load the indirect jump vector at $A404-$05 with $0300, the starting address of the interrupt routine. The Apple II interrupt structure is slightly different, so we have provided a separate listing for it. [5] Refer to listing 2 if you are an Apple II user. In listing 2 it is assumed that the John Bell Engineering 6522 board is located in slot seven and 6522-1 (U1) is being used. A 16-lead ribbon connector (DIP Jumper) makes a convenient connection between the 6522 board on the Apple and the outside world.

Note that the only hardware required is a single connection between the source of TTL level pulses and Pin PB6 on the 6522. No gates, no flip-flops, no inverters, and no rat's nest of wires are required.

The previous discussion, when coupled with the comments in the two listings, should make the assembly language routine understandable. It may be worth adding a few points related to the corrections that are made (lines 62-72 in listing 1) to the time after it is measured. Refer to listing 1. Immediately after the last event is detected, the two counter registers of the T1 timer are read (the LDY T1CL and the LDX T1CH instructions on lines 59 and 60). Recall that the counters count down. If the low-order byte of T1 is less than $04, then by the time the high-order byte is read the counter will have modified this byte to be one less than it should be. The INX instruction on line

***Listing 1:* Source File — Timer Program**

```
A804:           2 T1CL    EQU  $A804
A805:           3 T1CH    EQU  $A805
A806:           4 T1LL    EQU  $A806
A808:           5 T2CL    EQU  $A808
A809:           6 T2CH    EQU  $A809
A80B:           7 ACR     EQU  $A80B
A80D:           8 IFR     EQU  $A80D
A80E:           9 IER     EQU  $A80E
0019:          10 NUMB    EQU  $19        ;LOCATIONS $0019 AND
0000:          11 ;$001A CONTAIN THE NUMBER OF EVENTS
0000:          12 ;LESS ONE, TO BE COUNTED BY T2.
0000:          13 ;THE LEAST-SIGNIFICANT BYTE IS IN $0019.

001D:          15 TIME    EQU  $1D        ;LOCATIONS $001B TO
0000:          16 ;$001E CONTAIN THE FOUR-BYTE BINARY
0000:          17 ;MEASUREMENT OF THE TIME, T.
0000:          18 ;THE LEAST-SIGNIFICANT BYTE IS IN $001B.

----- NEXT OBJECT FILE NAME IS TIMER PROGRAM.
0300:          20         ORG  $0300


               22 ******************
0300:          23 ;INTERRUPT ROUTINE
0300:48        24         PHA             ;SAVE THE ACCUMULATOR ON THE STACK.
0301:E6 1D     25         INC  TIME       ;INCREMENT A TWO-BYTE
0303:D0 02     26         BNE  BR1        ;COUNTER FOR EACH
0305:E6 1E     27         INC  TIME+1     ;T1 INTERRUPT.
0307:AD 04 A8  28 BR1     LDA  T1CL       ;CLEAR T1 INTERRUPT FLAG.
030A:68        29         PLA             ;GET A FROM THE STACK.
030B:40        30         RTI


               32 ******************
030C:          33 ;TIMER SUBROUTINE
030C:D8        34         CLD             ;CLEAR THE DECIMAL MODE.
030D:A2 FF     35         LDX  #$FF
030F:A9 60     36         LDA  #$60       ;SET UP T1 TO RUN FREE
0311:8D 0B A8  37         STA  ACR        ;AND T2 TO COUNT PULSES.
0314:A9 FE     38         LDA  #$FE       ;SET UP THE T1 TIMER
0316:8D 06 A8  39         STA  T1LL       ;WITH $FFFE.
0319:A9 C0     40         LDA  #$C0       ;ENABLE IRQ FROM T1.
031B:8D 0E A8  41         STA  IER
031E:A9 00     42         LDA  #00        ;CLEAR TWO-BYTE
0320:85 1D     43         STA  TIME       ;INTERRUPT COUNTER.
0322:85 1E     44         STA  TIME+1
0324:8D 08 A8  45         STA  T2CL       ;START WITH 0 IN T2 TO
0327:8D 09 A8  46         STA  T2CH       ;DETECT THE ZEROTH EVENT.
032A:A9 20     47         LDA  #$20       ;SET UP MASK TO TEST T2
032C:2C 0D A8  48 WAIT    BIT  IFR        ;INTERRUPT FLAG, IFR5.
032F:F0 FB     49         BEQ  WAIT       ;WAIT FOR ZEROTH EVENT.
0331:8E 05 A8  50         STX  T1CH       ;START THE TIMER.
0334:58        51         CLI             ;MAKE SURE IRQ IS NOT MASKED.
0335:A5 19     52         LDA  NUMB       ;RELOAD T2 WITH
0337:8D 08 A8  53         STA  T2CL       ;NUMBER OF EVENTS.
033A:A5 1A     54         LDA  NUMB+1
033C:8D 09 A8  55         STA  T2CH
033F:A9 20     56         LDA  #$20       ;SET UP MASK FOR IFR5,
0341:2C 0D A8  57 LOAF    BIT  IFR        ;THE T2 FLAG.
0344:F0 FB     58         BEQ  LOAF       ;WAIT FOR ALL THE EVENTS.
0346:AC 04 A8  59         LDY  T1CL       ;READ THE LOW BYTE OF T1.
0349:AE 05 A8  60         LDX  T1CH       ;READ THE HIGH BYTE OF T1.
034C:78        61         SEI             ;MASK INTERRUPTS.
034D:C0 04     62         CPY  #04        ;ADJUST FOR READING HIGH BYTE AFTER
034F:B0 10     63         BCS  ARND       ;READING THE LOW BYTE.
0351:E8        64         INX             ;MAKE CORRECTION TO THE HIGH BYTE.
0352:D0 0D     65         BNE  ARND       ;DOES INTERRUPT COUNTER NEED
0354:38        66         SEC             ;CORRECTION? YES, DECREMENT IT
0355:A5 1D     67         LDA  TIME       ;BY SUBTRACTING ONE.
0357:E9 01     68         SBC  #01
0359:85 1D     69         STA  TIME
035B:A5 1E     70         LDA  TIME+1
035D:E9 00     71         SBC  #00
035F:85 1E     72         STA  TIME+1
0361:84 1B     73 ARND    STY  TIME-2     ;STORE LOW BYTE.
0363:86 1C     74         STX  TIME-1     ;STORE HIGH BYTE.
0365:A9 FE     75         LDA  #$FE       ;FIND THE LOW COUNT.
0367:E5 1B     76         SBC  TIME-2
0369:85 1B     77         STA  TIME-2     ;STORE IT.
036B:A9 FF     78         LDA  #$FF       ;FIND THE HIGH COUNT.
036D:E5 1C     79         SBC  TIME-1
036F:85 1C     80         STA  TIME-1     ;STORE IT.
0371:60        81         RTS

0372:          83 ;LOAD $A67E AND $A67F WITH $00 AND
0372:          84 ;$03, RESPECTIVELY, TO PRODUCE THE
0372:          85 ;INTERRUPT VECTOR FOR THE SYM-1.

*** SUCCESSFUL ASSEMBLY: NO ERRORS
```

***Listing 2:* Source File — Apple II Timer**

```
C704:            2 T1CL     EQU  $C704
C705:            3 T1CH     EQU  $C705
C706:            4 T1LL     EQU  $C706
C708:            5 T2CL     EQU  $C708
C709:            6 T2CH     EQU  $C709
C70B:            7 ACR      EQU  $C70B
C70D:            8 IFR      EQU  $C70D
C70E:            9 IER      EQU  $C70E
0019:           10 NUMB     EQU  $19         ;LOCATIONS $0019 AND
0000:           11 ;$001A CONTAIN THE NUMBER OF EVENTS
0000:           12 ;LESS ONE, TO BE COUNTED BY T2.
0000:           13 ;THE LEAST-SIGNIFICANT BYTE IS IN $0019.

001D:           15 TIME     EQU  $1D         ;LOCATIONS $001B TO
0000:           16 ;$001E CONTAIN THE FOUR-BYTE BINARY
0000:           17 ;MEASUREMENT OF THE TIME, T.
0000:           18 ;THE LEAST-SIGNIFICANT BYTE IS IN $001B.

----- NEXT OBJECT FILE NAME IS APPLE II TIMER.
1000:           20          ORG  $1000


                22 ******************
1000:           23 ;INTERRUPT ROUTINE
1000:E6 1D      24          INC  TIME        ;INCREMENT A TWO-BYTE
1002:D0 02      25          BNE  BR1         ;COUNTER FOR EACH
1004:E6 1E      26          INC  TIME+1      ;T1 INTERRUPT.
1006:AD 04 C7   27 BR1      LDA  T1CL        ;CLEAR T1 INTERRUPT FLAG.
1009:A5 45      28          LDA  $45         ;RESTORE THE ACCUMULATOR.
100B:40         29          RTI


                31 *******************
100C:           32 ;TIMER SUBROUTINE
100C:D8         33          CLD              ;CLEAR THE DECIMAL MODE.
100D:A2 FF      34          LDX  #$FF
100F:A9 60      35          LDA  #$60        ;SET UP T1 TO RUN FREE
1011:8D 0B C7   36          STA  ACR         ;AND T2 TO COUNT PULSES.
1014:A9 FE      37          LDA  #$FE        ;SET UP THE T1 TIMER
1016:8D 06 C7   38          STA  T1LL        ;WITH $FFFE.
1019:A9 C0      39          LDA  #$C0        ;ENABLE IRQ FROM T1.
101B:8D 0E C7   40          STA  IER
101E:A9 00      41          LDA  #00         ;CLEAR TWO-BYTE
1020:85 1D      42          STA  TIME        ;INTERRUPT COUNTER.
1022:85 1E      43          STA  TIME+1
1024:8D 08 C7   44          STA  T2CL        ;START WITH 0 IN T2 TO
1027:8D 09 C7   45          STA  T2CH        ;DETECT THE ZEROTH EVENT.
102A:A9 20      46          LDA  #$20        ;SET UP MASK TO TEST T2
102C:2C 0D C7   47 WAIT     BIT  IFR         ;INTERRUPT FLAG, IFR5.
102F:F0 FB      48          BEQ  WAIT        ;WAIT FOR ZEROTH EVENT.
1031:8E 05 C7   49          STX  T1CH        ;START THE TIMER.
1034:58         50          CLI              ;MAKE SURE IRQ IS NOT MASKED.
1035:A5 19      51          LDA  NUMB        ;RELOAD T2 WITH
1037:8D 08 C7   52          STA  T2CL        ;NUMBER OF EVENTS.
103A:A5 1A      53          LDA  NUMB+1
103C:8D 09 C7   54          STA  T2CH
103F:A9 20      55          LDA  #$20        ;SET UP MASK FOR IFR5,
1041:2C 0D C7   56 LOAF     BIT  IFR         ;THE T2 FLAG.
1044:F0 FB      57          BEQ  LOAF        ;WAIT FOR ALL THE EVENTS.
1046:AC 04 C7   58          LDY  T1CL        ;READ THE LOW BYTE OF T1.
1049:AE 05 C7   59          LDX  T1CH        ;READ THE HIGH BYTE OF T1.
104C:78         60          SEI              ;MASK INTERRUPTS.
104D:C0 04      61          CPY  #04         ;ADJUST FOR READING HIGH BYTE AFTER
104F:B0 10      62          BCS  ARND        ;READING THE LOW BYTE.
1051:E8         63          INX              ;MAKE CORRECTION TO THE HIGH BYTE.
1052:D0 0D      64          BNE  ARND        ;DOES INTERRUPT COUNTER NEED
1054:38         65          SEC              ;CORRECTION? YES, DECREMENT IT
1055:A5 1D      66          LDA  TIME        ;BY SUBTRACTING ONE.
1057:E9 01      67          SBC  #01
1059:85 1D      68          STA  TIME
105B:A5 1E      69          LDA  TIME+1
105D:E9 00      70          SBC  #00
105F:85 1E      71          STA  TIME+1
1061:84 1B      72 ARND     STY  TIME-2      ;STORE LOW BYTE.
1063:86 1C      73          STX  TIME-1      ;STORE HIGH BYTE.
1065:A9 FE      74          LDA  #$FE        ;FIND THE LOW COUNT.
1067:E5 1B      75          SBC  TIME-2
1069:85 1B      76          STA  TIME-2      ;STORE IT.
106B:A9 FF      77          LDA  #$FF        ;FIND THE HIGH COUNT.
106D:E5 1C      78          SBC  TIME-1
106F:85 1C      79          STA  TIME-1      ;STORE IT.
1071:60         80          RTS

1072:           82 ;LOAD $03FE AND $03FF WITH $00 AND
1072:           83 ;$10, RESPECTIVELY, TO PRODUCE THE
1072:           84 ;INDIRECT JUMP IN THE IRQ ROUTINE.

*** SUCCESSFUL ASSEMBLY: NO ERRORS
```

64 corrects this mistake should it occur. Furthermore, if the high-byte also decremented through zero in this time interval, then the interrupt counter is also in error (one interrupt too large). The instructions on lines 65-72 make the appropriate correction.

The timer routine must be used with a program to drive it. We chose to use a BASIC program to drive it from the Apple II, and the program we used is given in listing 3. [5] The remarks (REM statements) should make the BASIC program quite easy to understand. The program first requests the number of events to be counted. If you are measuring the time interval between two pulses, then the number you input is *one*. In that case, the most important output line is 95, and you may wish to delete lines 100, 110, 120, and 130.

If you are measuring frequency (i.e., the number of events per unit time), then you will need to enter the number of periods (pulses) to be counted when the program requests "...THE NUMBER OF EVENTS." For example, if you want to measure a frequency that is near 10,000 Hz, you might enter 10,000, giving a readout approximately once every second. If you want to count 50,000 pulses from a photomultiplier, for example, then you would enter 50,000 at this point. Statement 130 lists the number of pulses per second. You may wish to delete the other output statements. Finally, if you are measuring the period of a periodic waveform, then statement 120 will give the most desirable output.

Note that the program in listing 3 requires the *frequency of the microcomputer system clock*. In the Apple II this frequency is approximately 1.022714 MHz. (You should measure the frequency yourself if you want this many significant digits in your answer.) The SYM-1 and the AIM 65 will have a clock frequency that is near 1.00 MHz. The program in listing 3 should be modified accordingly.

Of course, the timer program can also be driven with an assembly language program. Some code to load the number of events to be counted into NUMB and NUMB + 1 will be required. A binary-to-BCD routine and a display routine will also be required. The assembly language code will depend heavily on the particular microcomputer being used, so we leave this problem for the reader. The literature associated with the various machines will probably contain the necessary routines if the machine's monitor does not.

The programs in listings 1, 2, and 3 can be easily modified to measure the *duration* of a *single logic-zero pulse.* Rather than using the T2 timer to detect a pulse, the TTL-level signal is applied to the CB1 pin on the 6522. A negative transition on this pin starts the timing sequence, while a positive transition on this pin is used to terminate the timing. The program in listing 4 shows how the program in listing 1 was modified. Listing 4 may be compared with listing 1 on a line-by-line basis. Note that the Peripheral Control Register is used to control the behavior of the CB1 pin. A diagram of the PCR register is shown in figure 8. Study it in connection with the program in listing 4. For this kind of measurement, the driver program (listing 3, for example) would not measure the period or the frequency; only the duration of the logic-zero pulse is of interest, namely the quantity T. Specific machines will require different driver programs, and the design of such a program is left for the reader.

A Brief Error Analysis

The true time, T, shown in figures 1 and 3, is not measured exactly by the programs described in this paper. The logic transitions that coincide with the events are detected by machine language loops, and the event can occur any time during the loop. However, timing start or end can only take place at the end of the loop. Call Tm the measured time. That is, Tm is the time that is output by the program. If T is less than 65536 clock cycles, then the precision of the measurement is plus or minus seven clock cycles. That is,

$$\mathrm{Tm} - 7\mathrm{Tc} \leq \mathrm{T} \leq \mathrm{Tm} + 7\mathrm{Tc}$$

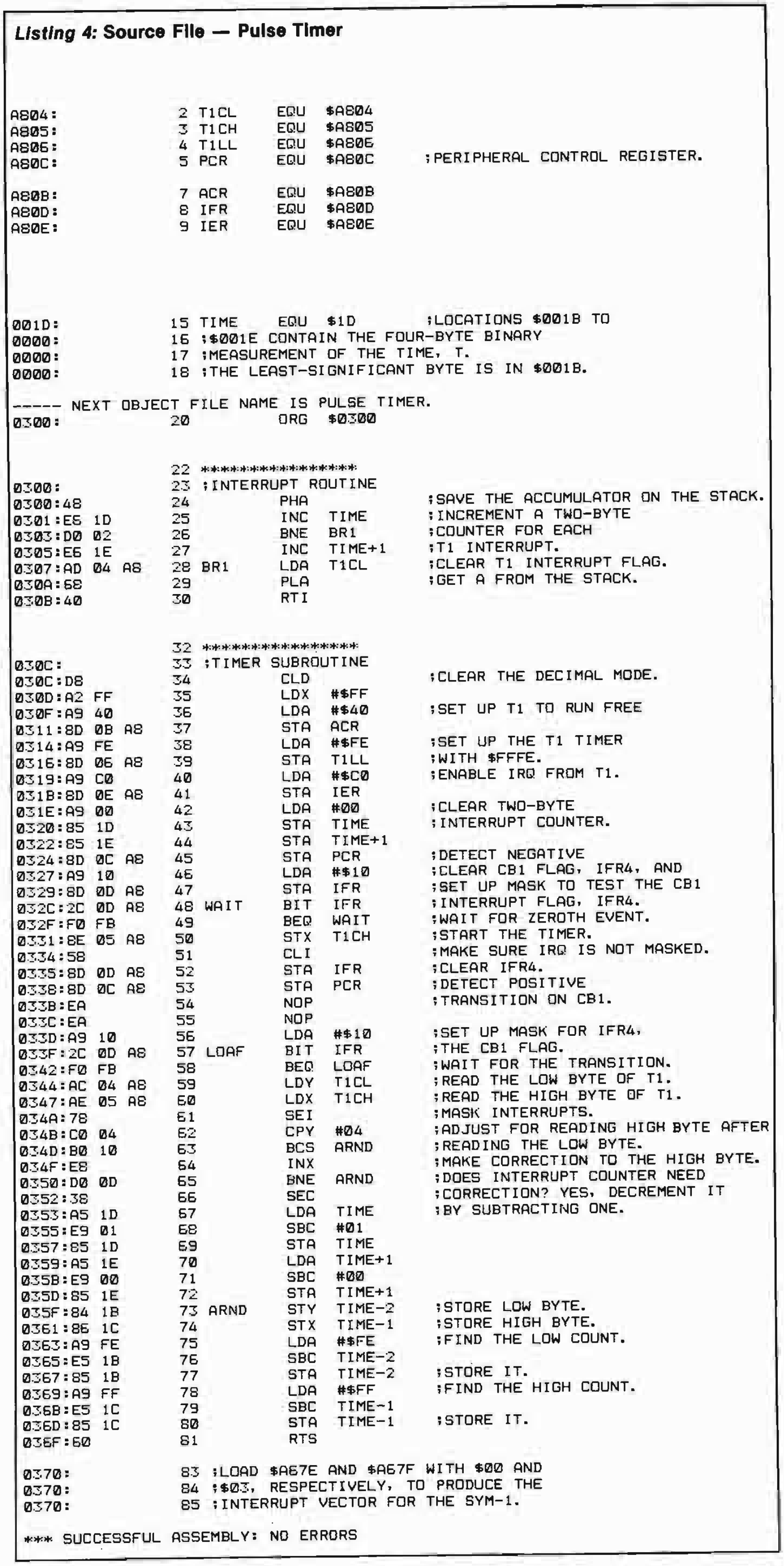
where Tc is the period of the microcomputer system clock, typically one microsecond, but approximately 0.977 microseconds for the Apple. The number 7 is a result of the fact that the loop that detects the pulse is seven clock cycles long.

If one or more interrupts from T1 have occurred, then an additional uncertainty is introduced because the last event may occur during an interrupt, but it will not be detected until the program returns to the BEQ LOAF loop in listing 1. In the case of the Apple, the IRQ-interrupt lasts about 55 clock cycles. This gives the following inequality relating the measured time,

Listing 3

```
1  REM  PRECISION TIMER PROGRAM
2Ø  PRINT "INPUT THE NUMBER OF EV
     ENTS."
25  PRINT "THIS NUMBER MUST BE LE
     SS THAN 65537."
3Ø  INPUT N
35 N = N - 1
36  REM  POKE N INTO TWO LOCATION
     S.
4Ø NHI =  INT (N / 256)
45  POKE 26,NHI
5Ø NLO = (N / 256 - NHI) * 256
55  POKE 25,NLO
56  REM  SET UP JUMP VECTOR.
6Ø  POKE 1Ø,76: POKE 11,12: POKE
     12,25
64  REM  CALL PRECISION TIMER SUB
     ROUTINE.
65 Z =  USR (Ø)
69  REM  CONVERT NUMBER OF CLOCK
     CYCLES FROM HEXADECIMAL TO D
     ECIMAL.
7Ø A =  PEEK (27)
75 B = 256 *  PEEK (28) + A
8Ø C = 65536 *  PEEK (29) + B
81 C = C / 1Ø22714
82  REM  CLOCK FREQUENCY = 1.Ø227
     14 MHZ.
85 D = (16777216 / 1Ø22714)* PEEK
     (3Ø)
9Ø T = D + C
1ØØ PERIOD = T / (N + 1)
11Ø F = 1 / PERIOD
12Ø  PRINT "THE PERIOD IS ";PERIO
     D;" SECONDS."
13Ø  PRINT "THE FREQUENCY IS ";F;
     " HERTZ."
14Ø  GOTO 65
```

Listing 4: Source File — Pulse Timer



```
A804:             T1CL     EQU   $A804
A805:             T1CH     EQU   $A805
A806:             T1LL     EQU   $A806
A80C:             PCR      EQU   $A80C        ;PERIPHERAL CONTROL REGISTER.

A80B:             ACR      EQU   $A80B
A80D:             IFR      EQU   $A80D
A80E:             IER      EQU   $A80E




001D:             TIME     EQU   $1D          ;LOCATIONS $001B TO
0000:             ;$001E CONTAIN THE FOUR-BYTE BINARY
0000:             ;MEASUREMENT OF THE TIME, T.
0000:             ;THE LEAST-SIGNIFICANT BYTE IS IN $001B.

----- NEXT OBJECT FILE NAME IS PULSE TIMER.
0300:                      ORG   $0300


                  ******************
0300:             ;INTERRUPT ROUTINE
0300:48                    PHA                ;SAVE THE ACCUMULATOR ON THE STACK.
0301:E6 1D                 INC   TIME         ;INCREMENT A TWO-BYTE
0303:D0 02                 BNE   BR1          ;COUNTER FOR EACH
0305:E6 1E                 INC   TIME+1       ;T1 INTERRUPT.
0307:AD 04 A8     BR1      LDA   T1CL         ;CLEAR T1 INTERRUPT FLAG.
030A:68                    PLA                ;GET A FROM THE STACK.
030B:40                    RTI


                  ******************
030C:             ;TIMER SUBROUTINE
030C:D8                    CLD                ;CLEAR THE DECIMAL MODE.
030D:A2 FF                 LDX   #$FF
030F:A9 40                 LDA   #$40         ;SET UP T1 TO RUN FREE
0311:8D 0B A8              STA   ACR
0314:A9 FE                 LDA   #$FE         ;SET UP THE T1 TIMER
0316:8D 06 A8              STA   T1LL         ;WITH $FFFE.
0319:A9 C0                 LDA   #$C0         ;ENABLE IRQ FROM T1.
031B:8D 0E A8              STA   IER
031E:A9 00                 LDA   #00          ;CLEAR TWO-BYTE
0320:85 1D                 STA   TIME         ;INTERRUPT COUNTER.
0322:85 1E                 STA   TIME+1
0324:8D 0C A8              STA   PCR          ;DETECT NEGATIVE
0327:A9 10                 LDA   #$10         ;CLEAR CB1 FLAG, IFR4, AND
0329:8D 0D A8              STA   IFR          ;SET UP MASK TO TEST THE CB1
032C:2C 0D A8     WAIT     BIT   IFR          ;INTERRUPT FLAG, IFR4.
032F:F0 FB                 BEQ   WAIT         ;WAIT FOR ZEROTH EVENT.
0331:8E 05 A8              STX   T1CH         ;START THE TIMER.
0334:58                    CLI                ;MAKE SURE IRQ IS NOT MASKED.
0335:8D 0D A8              STA   IFR          ;CLEAR IFR4.
0338:8D 0C A8              STA   PCR          ;DETECT POSITIVE
033B:EA                    NOP                ;TRANSITION ON CB1.
033C:EA                    NOP
033D:A9 10                 LDA   #$10         ;SET UP MASK FOR IFR4,
033F:2C 0D A8     LOAF     BIT   IFR          ;THE CB1 FLAG.
0342:F0 FB                 BEQ   LOAF         ;WAIT FOR THE TRANSITION.
0344:AC 04 A8              LDY   T1CL         ;READ THE LOW BYTE OF T1.
0347:AE 05 A8              LDX   T1CH         ;READ THE HIGH BYTE OF T1.
034A:78                    SEI                ;MASK INTERRUPTS.
034B:C0 04                 CPY   #04          ;ADJUST FOR READING HIGH BYTE AFTER
034D:B0 10                 BCS   ARND         ;READING THE LOW BYTE.
034F:E8                    INX                ;MAKE CORRECTION TO THE HIGH BYTE.
0350:D0 0D                 BNE   ARND         ;DOES INTERRUPT COUNTER NEED
0352:38                    SEC                ;CORRECTION? YES, DECREMENT IT
0353:A5 1D                 LDA   TIME         ;BY SUBTRACTING ONE.
0355:E9 01                 SBC   #01
0357:85 1D                 STA   TIME
0359:A5 1E                 LDA   TIME+1
035B:E9 00                 SBC   #00
035D:85 1E                 STA   TIME+1
035F:84 1B        ARND     STY   TIME-2       ;STORE LOW BYTE.
0361:86 1C                 STX   TIME-1       ;STORE HIGH BYTE.
0363:A9 FE                 LDA   #$FE         ;FIND THE LOW COUNT.
0365:E5 1B                 SBC   TIME-2
0367:85 1B                 STA   TIME-2       ;STORE IT.
0369:A9 FF                 LDA   #$FF         ;FIND THE HIGH COUNT.
036B:E5 1C                 SBC   TIME-1
036D:85 1C                 STA   TIME-1       ;STORE IT.
036F:60                    RTS

0370:             ;LOAD $A67E AND $A67F WITH $00 AND
0370:             ;$03, RESPECTIVELY, TO PRODUCE THE
0370:             ;INTERRUPT VECTOR FOR THE SYM-1.

*** SUCCESSFUL ASSEMBLY: NO ERRORS
```

Tm, the true time, T, and the period of the clock, Tc:

$$Tm - 7Tc \leq T \leq Tm + 62Tc$$

illustrating that the time to process the interrupt routine, expressed in clock cycles, must be added to 7Tc to give the upper limit for T.

The precision of the measurement is 7% for an interval of 100 microseconds, 0.7% for an interval of 1000 microseconds, 0.07% for an interval of 10,000 microseconds, etc. If T exceeds 65535 microseconds, the uncertainty increases. For example, if T is 100,000 microseconds, the precision of the measurement is approximately 62/100,000 or 0.062%.

Of course, it is being assumed, perhaps incorrectly, that the system clock frequency is known with an accuracy that exceeds the precision. Experience shows that the clock frequencies may be in error by as much as several hundred parts per million, or a relative uncertainty of approximately 0.03%. The user should be aware that the absolute accuracy of the system clock frequency may be an important factor in determining the accuracy of the results.

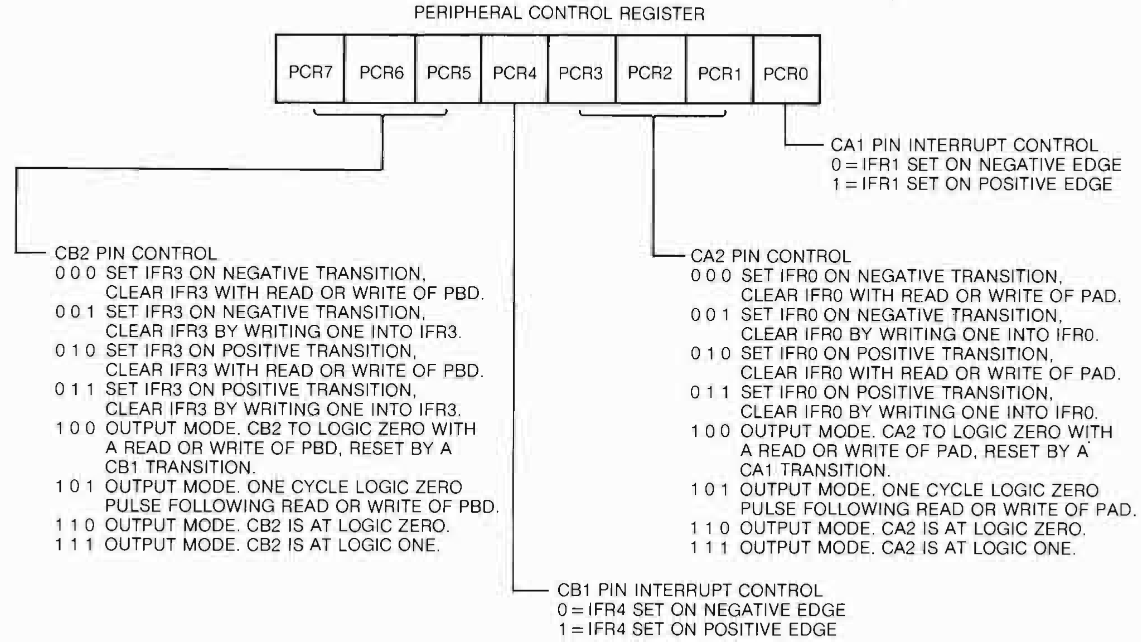
Applications

We will assume that the machine-language timing program (listing 1 or listing 2) is in place with the appropriate interrupt vectors loaded, and that a suitable driving routine (listing 3, for example) is also in place.

A. Frequency Counting

Connect the source of the TTL-level pulses to the PB6 pin on the 6522. The pulse source must produce a square wave. Applying a SINE wave to a Schmitt-trigger circuit (a 74LS14 or a 555 timer configured as a Schmitt trigger) will convert it to a square wave. Enter the number of pulses you wish to count. The program will measure the time required for the N pulses and output the frequency, f = N/T. For sub-audio frequencies you may wish to count only a few pulses, while for higher frequencies a larger N will give more precise results. Although the maximum number of pulses to be counted is 65536, the maximum frequency of the pulses should not exceed about 35,000 Hz. The reason for this maximum is a result of the fact that there are about 28 clock cycles between the time that T2 is loaded for the first time (line 45 in listing 1) and the time that it is

***Figure 7:* Diagram of the Peripheral Control Register (PCR)**



CA1 PIN INTERRUPT CONTROL
0 = IFR1 SET ON NEGATIVE EDGE
1 = IFR1 SET ON POSITIVE EDGE

CB2 PIN CONTROL
0 0 0 SET IFR3 ON NEGATIVE TRANSITION, CLEAR IFR3 WITH READ OR WRITE OF PBD.
0 0 1 SET IFR3 ON NEGATIVE TRANSITION, CLEAR IFR3 BY WRITING ONE INTO IFR3.
0 1 0 SET IFR3 ON POSITIVE TRANSITION, CLEAR IFR3 WITH READ OR WRITE OF PBD.
0 1 1 SET IFR3 ON POSITIVE TRANSITION, CLEAR IFR3 BY WRITING ONE INTO IFR3.
1 0 0 OUTPUT MODE. CB2 TO LOGIC ZERO WITH A READ OR WRITE OF PBD, RESET BY A CB1 TRANSITION.
1 0 1 OUTPUT MODE. ONE CYCLE LOGIC ZERO PULSE FOLLOWING READ OR WRITE OF PBD.
1 1 0 OUTPUT MODE. CB2 IS AT LOGIC ZERO.
1 1 1 OUTPUT MODE. CB2 IS AT LOGIC ONE.

CA2 PIN CONTROL
0 0 0 SET IFR0 ON NEGATIVE TRANSITION, CLEAR IFR0 WITH READ OR WRITE OF PAD.
0 0 1 SET IFR0 ON NEGATIVE TRANSITION, CLEAR IFR0 BY WRITING ONE INTO IFR0.
0 1 0 SET IFR0 ON POSITIVE TRANSITION, CLEAR IFR0 WITH READ OR WRITE OF PAD.
0 1 1 SET IFR0 ON POSITIVE TRANSITION, CLEAR IFR0 BY WRITING ONE INTO IFR0.
1 0 0 OUTPUT MODE. CA2 TO LOGIC ZERO WITH A READ OR WRITE OF PAD, RESET BY A CA1 TRANSITION.
1 0 1 OUTPUT MODE. ONE CYCLE LOGIC ZERO PULSE FOLLOWING READ OR WRITE OF PAD.
1 1 0 OUTPUT MODE. CA2 IS AT LOGIC ZERO.
1 1 1 OUTPUT MODE. CA2 IS AT LOGIC ONE.

CB1 PIN INTERRUPT CONTROL
0 = IFR4 SET ON NEGATIVE EDGE
1 = IFR4 SET ON POSITIVE EDGE

***Figure 8:* A Circuit to Debounce a Switch for a Stopwatch**

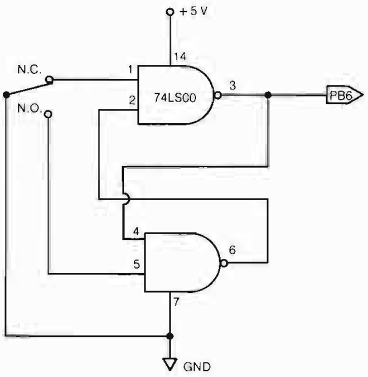


***Figure 9:* A Temperature-to-Frequency Converter**

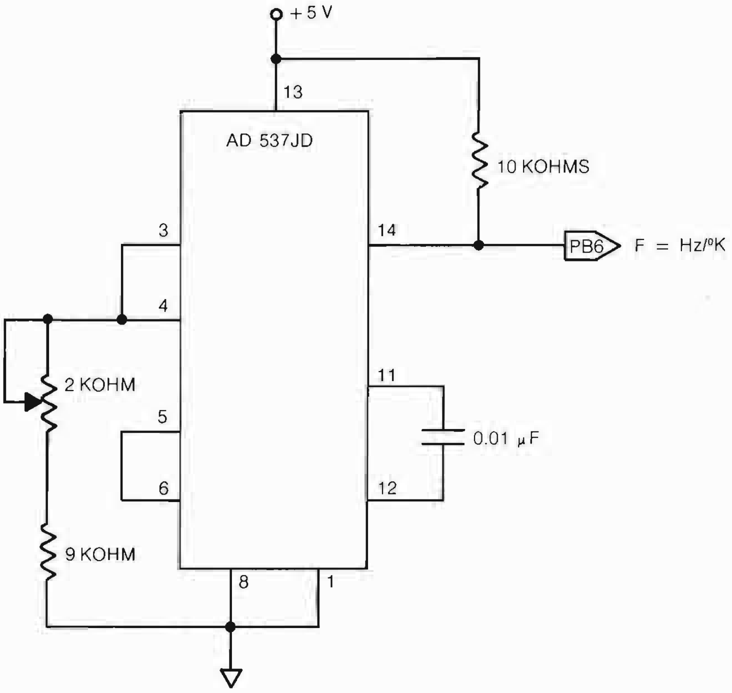
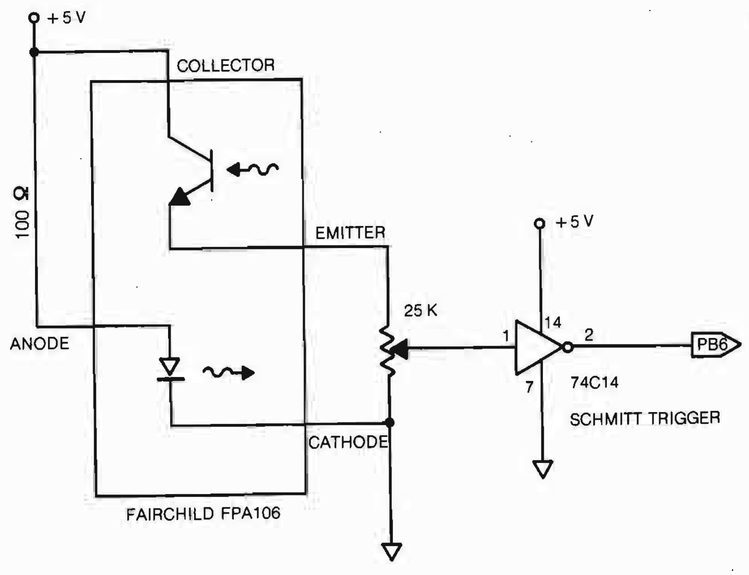


Figure 10: **A Photo-diode, Phototransistor Array**



reloaded (line 55 in listing 1). If a pulse occurs in this interval, it will not be counted, and the degree of error that is then introduced depends on the total number of pulses that are counted. If the pulse count is large, say 10,000 or more, this error may be negligible, but the user should be aware of its existence.

B. A Stop Watch

Enter "1" when the program requests the number of events. A suitable circuit to generate starting and stopping pulses with a mechanical switch is shown in figure 9. Each time the switch is connected to its N. O. (normally open) position, the output of the circuit goes to logic zero. The first closure initiates the timing sequence, the next closure terminates it. You may wish to modify the driving program to reflect the fact that you are measuring time rather than frequency.

C. Measuring Temperature

To measure temperature a T/F integrated circuit is used. In this case an Analog Devices (Route 1 Industrial Park, Box 280, Norwood, MA 02062) AD 537 was used in its temperature-to-frequency mode. The circuit diagram is given in figure 10. The AD 537 specification sheet, which you should request when you order the device, contains the necessary information to convert the pulse train from the device to a Kelvin, Celsius, or Fahrenheit temperature to be output from your microcomputer.

It should be clear that any voltage-to-frequency converter can be used with a variety of transducers to make measurements of physical quantities with these programs.

D. Tachometry

Another application involves measuring the rotation rate of a gear, fan, or wheel. A photodiode and a phototransistor make a suitable pickup device. Refer to the circuit in figure 11. A reflective surface on the rotating object passes near the diode-transistor pair once each rotation. Light emitted by the photodiode is reflected to the phototransistor and it conducts. The voltage across the 25 kohm potentiometer rises during each light pulse. A CMOS Schmitt trigger, the 74C14, will help to clean up the potentially noisy waveform from the phototransistor, producing a clean negative pulse for each pass of the reflecting surface. The pulses are counted and timed by the programs, and the rotation rate is the same as the frequency of the pulses.

If a piece of paper that had alternate dark and light strips on it were passed near the diode-transistor array, the pulse train frequency appearing on the PB6 output would be directly proportional to the velocity. With suitable modifications, therefore, the circuit in figure 11 and the programs in the listings are capable of measuring velocity.

The applications just suggested should generate some ideas for your own applications. The focus of this paper has been the assembly language programs that can be used to make precise measurements of either time or frequency with the simplest possible hardware (one connection to the PB6 pin) requirements.

Acknowledgement

Dr. Don Geilker of the Department of Physics at William Jewell College in Liberty, Missouri, provided the ideas that led to this work. His approach to the software was somewhat different than that described here, but the basic idea of using the 6522 to make the N/T measurement is his.

References

1. "6522 Chip Setup Time," Kosinski, J.T., and Suitor, R.F., *The Best of Micro*, Volume 2, pg. 93.

2. "Interfacing the Apple to 6500 Family Peripherals," Paul, D., and Wisman, J., *Compute!*, August 1981, pg. 74.

3. *6502 Assembly Language Programming*, Leventhal, L.A., Osborne/McGraw-Hill, 1979, pp. 11-36.

4. *Programming and Interfacing the 6502, with Experiments*, De Jong, M.L., Howard W. Sams, 1980, pg. 210.

5. The Apple II programs are part of a forthcoming book, *Apple II Assembly Language*, De Jong, M.L., Howard W. Sams, 1982.

Dr. De Jong may be contacted at The School of the Ozarks, Point Lookout, MO 65726.

Chances are, when you bought your first disk drive, it was an Apple. Now that you're ready for a second, take a look at Quentin.

Our Apple*-Mate™ 5¼" Disk Drive is fully software transparent with Apple's DOS 3.3 operating system in full and half track operation.

Add it to your present drive for greater capacity and faster access. Just plug it in and go to work.

And the Apple-Mate has these High Performance advantages:

ON TRACK HEAD SEEK

A precision lead screw positions the head onto the correct track. Time-consuming retries and disk-to-disk copying errors are virtually eliminated.

SIEMENS† DISK DRIVE

The apple-beige unit is built around the highly reliable Siemens system with over 10,000 lifetime hours. Shielded connecting cable also attached.

LONG TERM DEPENDABILITY

MTBF (Mean Time Between Failures)—8,500 power-on hours, and the unit has a one-year warranty.

COUNT ON QUENTIN FOR QUALITY

Quentin Research was building disk systems for the computer industry when Apple was a little bud on the big computer tree. We're known for product reliability and stand behind every system we sell you.

But the best news may be the price—only $335.00 (40 tracks).

A special introductory offer when you order Apple-Mate directly from us.

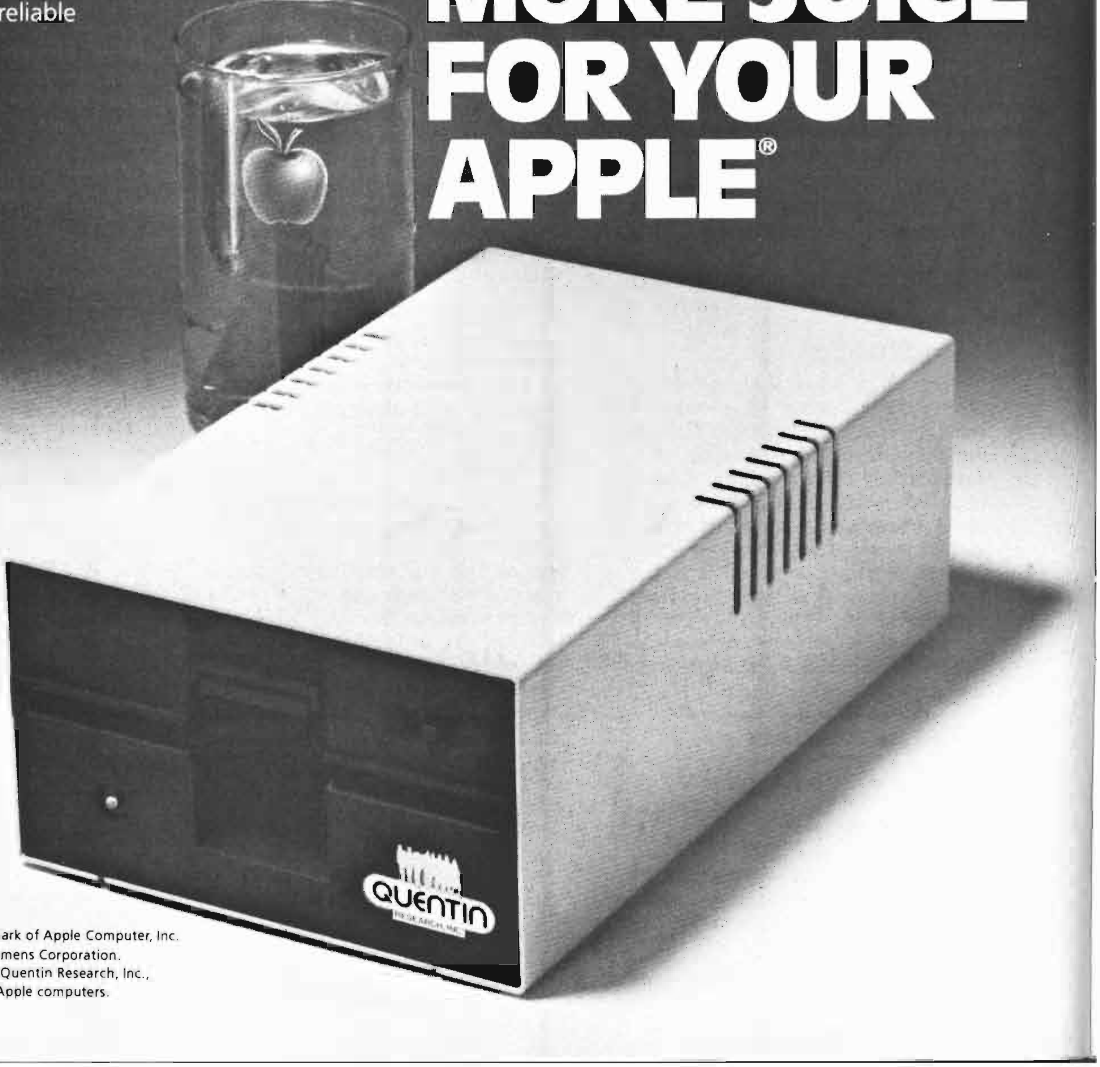
So when you're ready to boost the juice on your Apple, add-on the Quentin Apple-Mate.

To order: Check, money order, Visa or Mastercard number. Calif. residents add 6% sales tax. Allow one week delivery.

QUENTIN
RESEARCH, INC.

19355 Business Center Drive
Northridge, California 91324
(213) 701-1006

MORE JUICE FOR YOUR APPLE®



Special
Introductory
Price:
$335.00

® Apple is a registered trademark of Apple Computer, Inc.
†Siemens is a trademark of Siemens Corporation.
*Apple-Mate is a trademark of Quentin Research, Inc., which does not manufacture Apple computers.

Removing Frustrating Interference

by Patrick E. Hamel

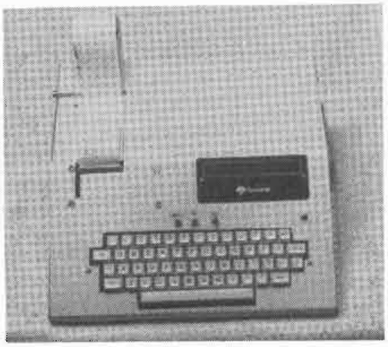
Several techniques for minimizing interference with single board computers are discussed. The AIM 65 is used as an example.

RFI (radio frequency interference) is a two-way street. Radiations from your computer can cause interference to your radio and television reception, while radiations from an older television set, amateur or citizens band radios, or even your video display can cause loss of data.

By using some basic principles we can avoid the cost of one of the new "FCC-accepted" computers and enjoy interference-free computing. The first principle is that wires do not really act like antennas (transmit or pick up interference) if there is a ground plane provided near them. This means that either a metal case, special ground layer on the circuit board, or shielded wires will remove or reduce interference.

If your circuit board has a special grounding layer, it probably has no radiation problem. If you make custom boards, leave one side as the ground plane. But how do you provide a ground plane for an existing board without shorting out the components? The safest way is to mount the board about 1/16 inch above a metal case, after insuring that the components will not short out if the case on the board flexes. For checking clearances, I recommend one of the $1.00 dental mirrors sold at the drugstore.

For the plastic case computer, I suggest a sandwich of poster-board, fanned-out stranded wire, aluminum foil, more fanned-out stranded wire, and more poster-board glued together. Be sure to hook the ends of the wires to the ground connection to shield the board.



The circuit board itself is probably not the major source of interferences; the flat-cables we use to hook up external tapes, disks, or displays usually cause or pick up most of the interference. The simplest answer is to pay extra and order shielded flat cable (there is such a thing) with each add-on.

It is also very easy to shield existing flat cable. Almost any drug store sells aluminum tape for muffler or ducting repair; simply cover the cable with the tape (both sides). To ground the shield either solder a wire to a paper clip or use alligator clips from the tape to your system ground.

A second principle is to keep the signals inside the case — mount displays and keyboards inside the enclosure rather than on the outside surface.

AIM 65 — An Example

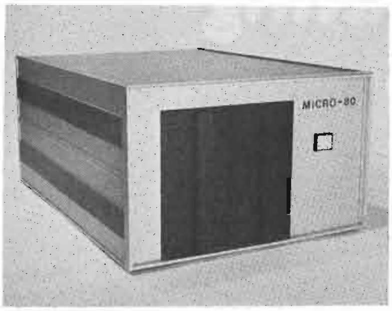
The scanning keyboard on the AIM can cause interference on a TV set at about 60 feet if an 18-inch cable is used between the main board and keyboard. This fact led to development of the enclosure in the picture. I ordered the case from a local TV parts store and did all work with hand tools. The paper roll hanger flip-up hatch may not be a good idea to duplicate because it limits room for expansion. The use of microphone connectors for the tapes is the result of many bad experiences with non-audio-rated connectors.

Now I can record, load, and compute one foot from an operating amateur transmitter, while the kids watch TV, all interference free.

Pat Hamel has been involved with computers since 1963, when he became an instructor on the old tube-type "Sage." He is currently involved with PLC Applications in support of the Shuttle Program. The AIM serves as a personal accounting device and development tool. You may contact Mr. Hamel at 1157 E. Old Pass Rd., Long Beach, MS 39560.

MICRO

NEW FROM D & N MICRO PRODUCTS, INC.



MICRO-80 COMPUTER

Z80A CPU with 4MHz clock and CP/M 2.2 operating system. 64K of low power static RAM. Calendar real time clock. Centronics type parallel printer interface. Serial interface for terminal communications, dip switch baud rates of 150 to 9600. 4″ cooling fan with air intake on back of computer and discharge through ventilation in the bottom. No holes on computer top or side for entry of foreign object. Two 8″ single or double sided floppy disk drives. IBM single density 3740 format for 243K of storage on each drive. Using double density with 1K sectors 608K of storage is available on a single sided drive or 1.2 meg on a double sided drive. Satin finish extruded aluminum with vinyl woodgrain decorative finish. 8 slot backplane for expansion. 48 pin buss is compatible with most OSI boards. Uses all standard IBM format CP/M software.

Model 80-1200 2 8″ single sided drives, 1.2 meg of storage	$2995
Model 80-2400 2 8″ double sided drives, 2.4 meg of storage	$3495
Option 001 Serial printer port, dip switch baud rate settings	$ 95

Software available in IBM single density 8″ format.

Microsoft	
Basic-80	$289
Basic Compiler	$329
Fortran-80	$410
Cobol-80	$574
Macro-80	$175
Edit-80	$105
Mu Simp/Mu Math	$224
Mu Lisp-80	$174

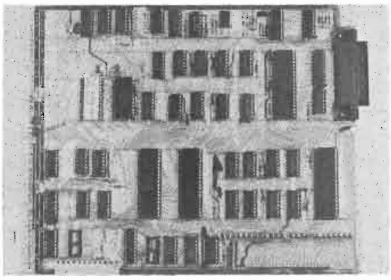
Digital Research	
PL/1-80	$459
Mac	$ 85
Sid	$ 78
Z-Sid	$ 95
C Basic-2	$110
Tex	$ 90
DeSpool	$ 50
Ashton-Tate	
dBase II	$595

Micropro	
Wordstar	$299
Mail-Merge	$109
Spellstar	$175
Super Sort I	$195
Pascal	
Pascal/MT +	$429
Pascal Z	$349
Pascal M	$355

Convert almost any static memory OSI machine to CP/M® with the D & N-80 CPU Board.

Z80A CPU with 4MHz clock. 2716 EPROM with monitor and bootstrap loader. RS-232 serial interface for terminal communications or use as a serial printer interface in a VIDEO system. Disk controller is an Intel 8272 chip to provide single or double density disk format. 243K single density or 608K double density of disk storage on a single sided 8″ drive. A double sided drive provides 1.2 meg of storage. DMA used with disk controller to unload CPU during block transfers from the disk drives. Optional Centronics type parallel printer port complete with 10 ft. cable. Optional Real Time Calendar Clock may be set or read using 'CALL' function in high level languages. Power requirements are only 5 volts at 1.4 amps. Available with WORDSTAR for serial terminal systems.

D & N-80	serial	$695
D & N-80	serial w/Wordstar	$795
D & N-80	video	$695
Option 001	parallel printer and real time calendar clock	$ 80



D & N-80 CPU BOARD

OTHER OSI COMPATIBLE HARDWARE

IO-CA10X Serial Printer Port $125
Compatible with OS-65U and OS-65D software

IO-CA9 Parallel Printer Port $175
Centronics standard parallel printer interface with 10 ft. flat cable

BP-580 8 Slot Backplane $ 47
Assembled 8 slot backplane for OSI 48 pin buss

24MEM-CM9	**$380**	**24MEM-CM9F**	**$530**
16MEM-CM9	**$300**	**16MEM-CM9F**	**$450**
8MEM-CM9	**$210**	**8MEM-CM9F**	**$360**
BMEM-CM9F	**$ 50**	**FL470**	**$180**

24K memory/floppy controller card supports up to 24K of 2114 memory chips and an OSI type floppy disk controller. Available fully assembled and tested with 8, 16, or 24K of memory, with floppy controller (F). Controller supports 2 drives. Needs separated clock and data inputs. Available Bare (BMEM-CM9F) or controller only (FL-470). Ideal way to upgrade cassette based system

C1P-EXP Expansion Interface $ 65
Expansion for C1P 600 or 610 board to the OSI 48 pin buss. Requires one slot in backplane. Use with BP-580 backplane

BIO-1600 Bare IO card $ 50
Supports 8K of memory, 2 16 bit parallel ports may be used as printer interfaces. 5 RS-232 serial ports, with manual and Molex connectors

DSK-SW Disk Switch $ 29
Extends life of drive and media. Shuts off minifloppy spindle motor when system is not accessing the drive. Complete KIT and manual

D & N Micro Products, Inc.
3684 N. Wells St.
Fort Wayne, Ind. 46808
(219) 485-6414

VISA MasterCard

TERMS $2.50 shipping, Foreign orders add 15%.
Indiana residents add 4% sales tax.

Disk Drives and Cables

8″ Shugart SA801 single sided	$395
8″ Shugart SA851 double sided	$585
FLC-6 6ft. cable from D & N or OSI controller to 8″ disk drive	$ 69
5 1/4″ MPI B51 with cable, power supply and cabinet	$450
FLC-5 1/4 8 ft. cable for connection to 5 1/4 drive and D & N or OSI controller, with data separator and disk switch	$ 75

Okidata Microline Printers

ML 82A Dot Matrix Printer $534
120 CPS, 80/120 columns, 9.5″ paper width, friction or pin feed

ML 83A Same as 82A except $895
16″ paper width, 132/232 columns with tractor feed

ML 84 Same as 82A except 200 CPS, $1152
16″ paper width, 132/232 columns, 2K buffer, dot addressable graphics, with tractor feed

An Overview of Apple DOS

by David P. Tuttle and Dr. Thomas Cleaver

This overview attacks the mystery of providing general information on the functional blocks of code in DOS. This article will enable the Apple user to manipulate DOS, permit DOS modification, and allow machine-language access to DOS commands.

Apple has made its Disk Operating System (DOS) user-friendly. One need know nothing of the internal functions of DOS in order to execute RUN, SAVE, MON, or any other DOS command. But an understanding of DOS internals *may* be required if you are concerned with:

- Determining memory use by DOS.
- Executing DOS commands from the monitor — or from an assembly-language program.
- Customizing DOS to use your own command names and error messages.
- Rewriting DOS to make it uncopyable.
- Retrieving information from a "clobbered" disk.

The purpose of this article is to discuss the internal workings of DOS and to catalog its functions. Toward this end, we have spent a good deal of time poking and prying into the code. However, there are still some blank spots and ambiguities in our understanding and there may even be some errors. Therefore, we ask you to be charitable in your assessment of our work.

In order to keep things simple and to minimize documentation, we have made certain assumptions and simplifications:

1. All addresses and data are in hex.
2. The version of DOS used is 3.3 (but most of the code is identical to DOS 3.2).
3. Memory size is 48K.
4. MAXFILES not changed (default = 3).
5. Master diskette is used.
6. Disk controller is in slot 6.
7. Language system is not installed.

Included in this review will be the "boot-up" sequence, the "idle" sequence, a list of contents and uses for the buffers, and a list of routines by address and function. Also included is a short explanation of how data is actually stored on the disk.

The "Boot-Up" Sequence

To initiate the DOS boot-up procedure, just turn on the power switch (if you have the autostart ROM). From the monitor, one may either type 6 CTRL-P <Return>, 6-CTRL-K <Return>, or C600G <Return>. Six is the slot number where the DOS ROM (Read Only Memory) card resides, and $C600 is the start of its memory locations.

The DOS ROM at $C600 starts the disk spinning and then reads track 0, sector 0 of the disk and stores it at $800-8FF. The code beginning at $801 is then executed. $801-84A reads in $3600-3FFF (the Read Write Track Sector (RWTS) Routine) by using DOS ROM at $C65C. Data at $350-$3FF are altered during this procedure. At $84A, a jump to $3700 is encountered.

$3700-3747 loads in the rest of DOS (on tracks 0, 1, and 2) at $1600-35FF. Included is an initialization routine in user Buffer #1 at $1B00. At $3747, a jump to $1B03 is encountered. $1B00-1C25 simply does some initialization when retrieving DOS from the disk; this code is never used again. A jump to $1D84 is encountered at $1B61 or $1C25 ($1C25 jumps to $1E25 which, in turn, is a jump to $1D84). $1D84 is the "hard entry point" to DOS. The entire DOS package is then moved from $1600-3FFF to $9600-BFFF. This is the top of RAM (Random Access Memory).

Next, the type of BASIC to be used is found (Applesoft or Integer BASIC). At $9E20-9E40 DOS checks to see if this is the first pass through this code since boot-up. If it is, then the greeting program (usually "Hello") is run. This greeting program is the one that was entered when the disk was INITialized. DOS enters the name of the greeting program into *File Name* Buffer #1 ($AA75-AA92), and then loads #06 into $AA5F (which is the command code number for the RUN command). It then jumps to $A180 to execute the RUN command of the greeting program whose name is now in File Name Buffer #1. DOS then utilizes the subroutines at $A180 (a) to match the command code number (in $AA5F, see table 2) with the appropriate entry point; (b) to put that entry point on the top of the stack; and (c) to jump to that entry point (plus one) by executing an RTS (return). After RUNning the greeting program, DOS jumps to $9E81, the DOS idle routine.

DOS Idle Sequence

When the cursor is flashing and the Apple is waiting for you to do something, the Apple is "idling." DOS normally idles at $9E81-9EBF. This is the input character routine. Here the cursor is put on the screen, characters are echoed to the screen from the keyboard, and characters are stored in the key-in buffer. For most commands keyed in, this also loads #$03 in $AA52 (which selects the machine state of DOS, see table 2), which in turn selects DOS #3. The program eventually jumps to $9EBD. Every time a character is to be output, this routine is called. The output "hook" at $0036, 0037 also points to $9EBD to output a character in the accumulator. A hook is a pointer at an address, such as $0038,0039 for the input hook, which points to another address for an indirect jump. $9EBD-9ED0 pushes an address from the state machine table (table 2, $9D10-$9D1D) onto the stack and then RTS's to jump to that address plus one. This selects DOS state 0-6 by indexing down the table ($AA52 contains the indexing number). DOS state #3 (< $AA52 = 3 >) is normally selected.

DOS state #3 ($9F2F) can do one of three things:

a) output unconditionally (print something)

b) output on the condition that MON C bit is set

c) decode key-in buffer as command *via* $9F15 (DOS #1)

DOS #3 checks for a < carriage return >. If it finds one, it then jumps to $9F15. The code at $9F15-9F22 eventually causes a jump to $9FCD, which decodes the string of characters in the key-in buffer.

Command decoding occurs at $9FCD. If errors are encountered, the program jumps to the error-processing routine at $A6D2. Otherwise, $AA5F is given the appropriate "command code number." $AA5F is then used to index down table 3 to get the corresponding entry point of the command to be executed. The program next jumps to $A180, where it pushes the command entry point (high and low order bytes) on the stack, and then jumps to the entry point plus one by executing an RTS.

The DOS Tables

The tables included with this article should provide enough information to understand the internal workings of DOS. You should find them useful to trace the path of a command as it is executed by DOS, or to modify DOS for your own purposes.

References

1. *Fort Worth Apple User's Group (FWAUG)*, June - October 1980.

2. Roe, David, "Sixteen *vs.* Thirteen," *LAUGHS*, Louisville Apple Users Group, 2, 4, Sept. 1980.

David P. Tuttle and Dr. Thomas Cleaver may be contacted at the Department of Electrical Engineering, University of Louisville, Louisville, KY 40292.

***Table 1:* Miscellaneous Addresses in DOS**

Note: The contents are stored in two consecutive bytes, low order byte first.

Table Address (Entry Point)	Contents	Description
9D00	9CD3	File Buffer #1
9D02	9E81	Input character routine
9D04	9EBD	Output character routine
9D06	AA75	Filename Buffer #1
9D08	AA93	Filename Buffer #2
9D0A	AA60	LENGTH of load
9D0C	9D00	Beginning of DOS
9D0E	B5BB	End of system buffer

***Table 2:* State Machine Table (categorizes the functions of DOS)**

$AA52 is used to index down $9D10 - 9D1C and find the entry point. $9EC0 - $9ED0 takes entry point, shoves it on the stack, and then jumps to it (plus one) by executing an RTS.

Table Address	Contents (Entry Point)	Description
9D10	9EEB minus 1	DOS #0(< $AA52 > = 0): Default value on DOS entry (set at $9DDA). Also used at front of line output from a program.
9D12	9F12 – 1	DOS #1 (< $AA52 > = 1): Outputting CTRL-D line from program; collect the line for decoding.
9D14	9F23 – 1	DOS #2 (< $AA52 > = 2): Outputting normal line from program. Print to the output device.
9D16	9F2F – 1	DOS #3 (< $AA52 > = 3): Output a character being echoed from the input routine (keyboard or EXEC file). Can do one of three things: a) output unconditionally, b) output on the condition that MON C bit is set, c) decode keyin buffer as command *via* $9F15.
9D18	9F52 – 1	DOS #4 (< $AA52 > = 4): "WRITE" is active, middle of line, states 4 and 5 work together to output to the disk until a line comes along with a CTRL-D in the front.
9D1A	9F61 – 1	DOS #5 (< $AA52 > = 5): "WRITE" is active, front of line, then go to DOS #4.
9D1C	9F71 – 1	DOS #6 (< $AA52 > = 6): Echoing character input from "READ" file. Ignore character for DOS command purposes.

***Table 4:* Vectors for Various Languages**

Vectors are used by DOS to interface with the supported languages (Applesoft ROM, Applesoft RAM, and Integer BASIC). DOS uses these addresses to jump into the language when RUNning (or CHAINing in the case of Integer BASIC) a new program, or when processing errors. Errors may occur during program execution or at the command level. (Applesoft also has a 6th vector to relink the pointers in each program line so the program source doesn't need to be loaded with the program start address each time. This is used throughout DOS.)

Current Language (Applesoft or Integer BASIC)

Table Address	Contents (Entry Point)	
9D56	(Moved in from	CHAIN entry
9D58	below tables	RUN entry
9D5A	when needed,	Error entry
9D5C	initially zero)	Hard entry
9D5E		Soft entry
9D60		Recompute links: (Applesoft only)

Integer BASIC (moved in above when needed)

Table Address	Contents (Entry Point)	
9D62	E836	CHAIN
9D64	A4E5	RUN
9D66	E3E3	Error
9D68	E000	Hard entry
9D6A	E003	Soft entry
	0000	Not used

Applesoft ROM (moved in above when needed)

Table Address	Contents (Entry Point)	
9D6C	A4FC	CHAIN (actually RUN)
9D6E	A4FC	RUN
9D70	D865	Error
9D72	E000	Hard entry
9D74	D43C	Soft entry
9D76	D4F2	Recompute links

Applesoft RAM (disk version moved in above when needed)

Table Address	Contents (Entry Point)	
9D78	A506	CHAIN
9D7A	A506	RUN
9D7C	1067	Error
9D7E	9D84	Hard entry
9D80	0C3C	Soft entry
9D82	0CF2	Recompute links

Note: All addresses in hex, low order then high order bytes at the addresses in the table.

Table 5: Miscellaneous Variables

ADDRESS	CONTENTS (TYPICAL)	DESCRIPTION
AA4F	982D	Current file buffer pointer
AA51	00	Input state
AA52	0006	Output state
AA53	FDF0	Output hook
AA55	FD18	Input hook
AA57	03	No. of buffers
AA58	03	-
AA59	00	Save S register
AA5A	00	Save X register
AA5B	00	Save Y register
AA5C	00	Save A register
AA5D	06	Line buffer displacement
AA5E	00	MON-NOMON Values
AA5F	00-36	Command code #
AA60	11B4	Block length for LOAD and BLOAD
AA62	00	Holds AA5F for a time
AA63	00	Temporary 1A
AA64	00	Temporary 2A
AA65	00	Command input option
AA66	0000	Command volume
AA68	0001	Command drive
AA6A	0006	Command slot
AA6C	0001	Command L value (length)
AA6G	0000	Command R value (record)
AA70	0000	Command B value (byte)
AA72	0800	Command A value (address)
AA74	0C	CIO bits
AA75	30	File name buffer #1
AA93	30	File name buffer #2 (for RENAME)
AAB1	03	Number of default file buffers
AAB2	84	Command chain (CTRL-D)
AAB3	00	EXEC file state
AAB4	0000	EXEC file buffer pointer
AAB6	00	Applesoft/Integer BASIC switch. 00 - Integer BASIC 40 - ROM Applesoft 80 - RAM Applesoft
AAB7	00	Applesoft begin run switch $00-NO. $40,$80=YES
AAB8		"Applesoft" in modified ASCII (bit 7 high of each byte)
AAC1	B7E8	RWTS buffer (IOB)
AAC3	B3BB	VTOC buffer
AAC5	B4BB	SYS. buffer
AAC7	C000	Top of RAM(+1)(last byte of DOS)

Table 6: I/O Package Commands

This table is used at $AB14 to $AB1E to jump to the correct I/O routine. $B5BB is used to choose which I/O routine will be

TABLE ADDRESS	CONTENTS (ENTRY POINT)	DESCRIPTION
AAC9	B37F minus 1	Good return (dummy)
AACB	AB22-1	OPEN file
AACD	AC06-1	CLOSE file
AACF	AC58-1	READ from file
AAD1	AC70-1	WRITE to file
AAD3	AD2B-1	DELETE file
AAD5	AD98-1	Print CATALOG
AAD7	ACEF-1	LOCK a file
AAD9	ACF6-1	UNLOCK a file
AADB	AC3A-1	RENAME a file
AADD	AD12-1	POSITION file
AADF	AE8E-1	Format disk (INIT)
AAE1	AD18-1	VERIFY file
AAE3	B37F-1	Good return (dummy)

Table 7: Read Commands

This table is used at $AC58 to $AC69 to jump to the correct read routine. $B5BC is used to index to the correct entry point address.

TABLE ADDRESS	CONTENTS (ENTRY POINT)	DESCRIPTION
AAE5	B37F-1	Good return
AAE7	ACBA-1	Read next byte
AAE9	AC96-1	Read next block
AAEB	AC87-1	Read specific byte
AAED	AC93-1	Read specific block
AAEF	B37F-1	Good return (dummy)

Table 8: Write Commands

This table is used at $AC70 to $AC86 to jump to the correct write routine. Again $B5BC determines which routine to jump to.

TABLE ADDRESS	CONTENTS (ENTRY POINT)	DESCRIPTION
AAF1	B37F-1	Good return
AAF3	ACBE-1	Write next byte
AAF5	ACCA-1	Write next block
AAF7	ACBB-1	Write specific byte
AAF9	ACC7-1	Write specific block
AAFB	B37F-1	Good return (dummy)

Note: Low order byte first then high for the addresses in the table.

***Table 9:* Encoded Nibbles (Ref. 2)**

Based solely upon deduction from a disassembled listing of the new controller card PROM, it appears that the Language System (and so 3.3 too) uses 6-bit nibbles instead of 5-bit nibbles. These nibbles are used as the intermediary stage in translating the bytes recorded on the disk surface into real memory bytes.

Under DOS 3.2, data is written to the disk using only 8-bit bytes which meet the following criteria:

Bit 7 is always set
No two consecutive bits are 0

Exactly 34 bytes meet these criteria. Their values fall between $AA and $FF. Two bytes, $D5 and $AA, are given special significance and are not used by 3.2 for storing data. They are used to signal the beginning of a sector's address or data segment. That leaves 32 distinct bytes, exactly the number needed to identify uniquely all possible 5-bit nibbles. They are mapped by the controller card as follows:

Byte/5-Bit Nibble		Byte/5-Bit Nibble		Byte/5-Bit Nibble		Byte/5-Bit Nibble	
AB	0 = 00000	AD	1 = 00001	AE	2 = 00010	AF	3 = 00011
B5	4 = 00100	B6	5 = 00101	B7	6 = 00110	BA	7 = 00111
BB	8 = 01000	BD	9 = 01001	BE	10 = 01010	BF	11 = 01011
D6	12 = 01100	D7	13 = 01101	DA	14 = 01110	DB	15 = 01111
DD	16 = 10000	DE	17 = 10001	DF	18 = 10010	EA	19 = 10011
EB	20 = 10100	ED	21 = 10101	EE	22 = 10110	EF	23 = 10111
F5	24 = 11000	F6	25 = 11001	F7	26 = 11010	FA	27 = 11011
FB	28 = 11100	FD	29 = 11101	FE	30 = 11110	FF	31 = 11111

(In addition to its use in data representation, $FF is the 'filler' which is written everywhere there isn't anything else.)

However, DOS 3.3 codes data into bytes that meet these criteria:

Bit 7 is always set... that's the same
At least two adjacent bits *must* be set (i.e., have value 1)
No more than two consecutive bits may be clear (have value 0)
There must be no more than one pair of consecutive bits clear

Many more bytes meet these requirements. In fact, 64 of them do, and that's precisely the number needed for a unique one to one mapping of the 64 different 6-bit nibbles.

Table stored at $BA29 - BA68

Byte/6-Bit	Nibble	Byte/6-Bit	Nibble	Byte/6-Bit	Nibble	Byte/6-Bit	Nibble
96	000000	97	000001	9A	000010	9B	000011
9D	000100	9E	000101	9F	000110	A6	000111
A7	001000	AB	001001	AC	001010	AD	001011
AE	001100	AF	001101	B2	001110	B3	001111
B4	010000	B5	010001	B6	010010	B7	010011
B9	010100	BA	010101	BB	010110	BC	010111
BD	011000	BE	011001	BF	011010	CB	011011
CD	011100	CE	011101	CF	011110	D3	011111
D6	100000	D7	100001	D9	100010	DA	100011
DB	100100	DC	100101	DD	100110	DE	100111
DF	101000	E5	101001	E6	101010	E7	101011
E9	101100	EA	101101	EB	101110	EC	101111
ED	110000	EE	110001	EF	110010	F2	110011
F3	110100	F4	110101	F5	110110	F6	110111
F7	111000	F9	111001	FA	111010	FB	111011
FC	111100	FD	111101	FE	111110	FF	111111

It takes $199 5-bit nibbles to encode the data in $100 8-bit bytes. It only takes $156 6-bit nibbles to encode the same data. The saving in disk space in each sector, plus a slight reduction in the spacing between sectors, frees up enough room for three additional sectors. This is why DOS 3.2 has 13 sectors per track and DOS 3.3 has 16.

Table 10: DOS Buffers and Their Contents

STARTING ADDRESS	ENDING ADDRESS	DESCRIPTION
9600	9852	USER FILE Buffer #3
9853	9AA5	USER FILE Buffer #2
9AA6	9CFF	USER FILE Buffer #1 These 3 buffers contain the contents of the current data sector, current track/sector, list sector, and miscellaneous information about the file.
9D00	9D83	Miscellaneous address vectors used within DOS. See Tables 1, 2, 3, and 4.
A884	A907	DOS modified ASCII command table (bit seven always high on last character, bit seven of the other characters is clear.) e.g., 52, 55, CE is RUN.
A908	A940	DOS command parameter validity table. Checks validity of various parameters with various commands. Set=yes it does, zero=not allowed. First Byte: Bit 7 - BASIC uses this command if no file name (LOAD, RUN, SAVE) Bit 6 - Needs no parameters Bit 5 - Uses file name Bit 4 - Uses 2 file names Bit 3 - PR# and IN# commands Bit 2 - Maxfiles only (2 & 3 allow a number as the only parameter) Bit 1 - Not valid in direct mode (OPEN, READ) Bit 0 - Command writes to disk Second Byte: Bit 7 - Takes C, I, O parameters Bit 6 - Takes V parameter Bit 5 - Takes D parameter Bit 4 - Takes S parameter Bit 3 - Takes L parameter Bit 2 - Takes R parameter Bit 1 - Takes B parameter Bit 0 - Takes A parameter
A941	A94A	Parameter prefix names (V, D, S, L, R, B, A, C, I, O)
A94B	A954	Parameter prefix byte A94B $40 for V parameter A94C $20 D A94D $10 S A94E $08 L A94F $04 R A950 $02 B A951 $01 A A952 $C0 C A953 $A0 I A954 $90 O
A955	A970	Parameter prefix ranges (VDSLRBA in the normal order). The valid ranges for each parameter. First the lowest (minimum) value and then the highest (maximum) valid value.
A971	AA3E	Modified ASCII error messages. (Bit seven of last character is set). Example: 45,4E,44,20,4F,46,20,44,41,54,C1 = END OF DATA.
AA3F	AA4E	Error message offsets, number of bytes beyond A971 that an error message starts. One offset per error message.
AA4F	AA74	Miscellaneous variables. See Table 5.
AA75	AA92	File name Buffer #1, holds file name of file being used at present time.
AA93	AAAF	File name Buffer #2, holds file name while Applesoft is set up.
AAB0	AAC8	See Table 5.
AAC9	AAE4	I/O package commands, see Table 6.
AAE5	AAF0	Read command, see Table 7.
AAF1	AAFC	Write command, see Table 8.
B397	B3A6	Miscellaneous variables.
B3A7	B3AE	"T,I,A,B,S,R,A,B" in modified ASCII (Bit 7 high) for output or possibily input comparison purposes.
B3AF	B3BA	"DISK VOLUME" In modified ASCII. (Bit 7 high for each character) for output or input comparison.
B3BB	B4BA	VTOC (Volume Table of Contents). See DOS Manual P. 132. This is loaded whenever a disk access is made.
B4BB	B5BC	System buffer - last accessed sector from disk before moving to normal program memory location or out to screen if catalog.
B5BD	B5D0	File manager parameters.
B5D1	B5FE	File manager scratchpad.
B64D	B65D	Data for use in Boot in code, possibly timer decrementing or indexing numbers list of decreasing odd then even hex words.
B6B3	B6CF	Not used.
B6E8	B6FE	Not used.
B7DF	B7E7	Scratchpad.
B7E8	B7FF	IOB buffer, includes device characteristics table (DCT) at B7FB for RWTS.
BA11	BA28	Miscellaneous data.
BA29	BA68	Encoded nibbles, Table 9.
BA69	BAFF	Decoding or encoding buffer.
BB00	BC55	I/O encoded nibble buffer; it takes $156 6-bit nibbles to encode $100 of data.
BFA8	B7C7	Miscellaneous data.

Table 11: DOS Code Functions Divided Into Blocks By Functions

FROM	TO	DESCRIPTION
9600	9852	USER FILE Buffer #3
9853	9AA5	USER FILE Buffer #2
9AA6	9CFF	USER FILE Buffer #1 Contains content of the current data sector, carries track/sector, list sector and miscellaneous information about file.
9D00	9D0F	Miscellaneous addresses used within DOS address constants (see Table 1).
9D10	9D1D	Addresses used in state machine that routes output characters (used from 9EC0 to 9ED0, $AA52 is used to choose which one) (DOS 0 to 6). See Table 2.
9D1E	9D55	DOS command branch vectors used from A186 to A192 with AA5F used to choose which one to call. See Table 3.
9D56	9D61	Vectors used by DOS to interface with the various support languages (see Table 4). (Current language - Applesoft or Integer BASIC).
9D62	9D6B	Integer BASIC (moved in when needed).
9D6C	9D77	Applesoft ROM (moved in when needed).
9D78	9D83	Applesoft RAM (disk version)(moved in when needed).
9D84	9DBE	Which BASIC? Hard entry point.
9DBF	9DD0	DOS soft entry - $3D0 jumps here, as does reset with an autostart ROM. (Routine to re-initialize DOS).
9DD1	9DE9	Both hard and soft entries join.
9DEA	9E1F	Initialize DOS buffers and set vectors for RAM Applesoft called from DOS KEYIN routine on a hard entry or FP command. (But only when there is no Applesoft ROM card and Applesoft must come off the disk.)
9E20	9E50	$AA5F is 0 if this is the first character input since the Boot. That is because $AA5F holds the command number and 0 is for INIT. The DOS image still has 0 left from when it was created. This signals to RUN the HELLO program and other beginning things.
9E51	9E80	TABLE OF COMMANDS, etc. that is copied into $3D0 to 3FF on boot up (such as JMP and entry point.)
9E81	9EBC	INPUT CHARACTER ROUTINE - DOS comes here every time a program uses JSR $FD1B or $FD0C to input a character. That includes every BASIC input statement or every line typed to the BASIC prompt (] or >) (Handles input hook).
9EBD	9EEA	This routine pushes an address from the state machine table onto the stack and then RTS's to jump to that address plus 1. If DOS is active the output hook points to this address. A hook is a pointer in one address that points to another address i.e., every character output causes this routine to be called with that character in the ACC.
9EEB	9F11	DOS #0 ENTRY (<$AA52> =0) 1. Default value on DOS entry (set at $9DDA). Also used at front of line ouputted from a program. 2. Checks for some special cases as follows: a. If Applesoft begin run switch is set then clear it and rejoin "RUN" command. b. If 2nd character printed after inputting from 'READ' FILE is '?' then just echo (Assuming MON I). c. If 2nd character printed after input from 'EXEC' file is the prompt, then just echo (ANY MON). d. If charater is CTRL-D, collect line by using STATE2 next. INPUT SWITCH ($AA51) is clear if input is coming from the keyboard; it is set if input comes from a 'READ' or 'EXEC' file. It is also set on boot entry. If input is from a file we test the character being outputted.
9F12	9F22	DOS #1 ENTRY (<$AA52> = 1) Outputting CTRL-D line from program, so we collect the line for decoding.
9F23	9F2E	DOS #2 ENTRY (<$AA52> = 2) Outputting normal line from program, so just print to output device, usually the screen.
9F2F	9F51	DOS #3 ENTRY (<$AA52> = 3) Come here to output a character being echoed from the input routine (keyboard or' EXEC file) so we do one of 3 things: 1. O) output unconditionally via $9FA4 2. M) maybe output (if MONC bit set) via $9F9D 3. D) decode KEYIN buffer as command via $9F15.

(Continued)

Table 11 *(continued)*

```
                          This depends on:
                          1. what character is being outputted
                          2. if BASIC is running (active)
                          3. if an 'EXEC' file is active.

                          CHARACTER IS

                              <CR>                NOT <CR>
                          BASIC ACTIVE          BASIC ACTIVE
                           NO      YES           NO     YES
          E        NO      D        O      E  NO  O       O
          X                                X
          E       YES      D        D      E YES  M       M
          C                                C
```

9F3F	9F48	Only used when program echoes a <CR> after getting a character from keyboard or 'EXEC file'.
9F49	9F4B	Here the carry represents: "Is BASIC running but there is no active EXEC file?" CC=YES C5=NO
94FC	9F51	Put <CR> in buffer at position following string inputted and then try to decode it as a command. Also comes here for keyboard entry to], > or * prompt or any
9F52	9F60	DOS #4 ENTRY (<$AA52> = 4) 'Write' is active, middle of line. States 4 and 5 work together to output to the disk until a line comes along with a CTRL-D on the front.
9F61	9F70	DOS #5 ENTRY (<$AA52> = 5) 'Write' is aotive, front of line. Then go to
#4. 9F71	9FC7	DOS #6 ENTRY (<$AA52> = 6) Echoing character input from 'READ' file. Ignore character for DOS command purposes. Come here on first character outputted after running Applesoft BASIC program from disk.
9F83	9F8A	Return from DOS command execution.
9F8B	9F94	If not CTRL-D, clear out the buffer and pretend the user didn't type anything but a <CR>. This is designed to work well with routine line 'GETLN' in the Apple II ROM ($FD6A).
9F95	9F9E	Set up the accumulator with the correct bit to test the 'MON' output MODE: $40 for command echo $10 for DISK output echo $20 for DISK input echo
9F9F	9FA3	Test the bit and output if it is set in the MON byte at $AA5E.
9FA4	9FB2	Call the real output routine but keep control of the registers.
9FB3	9FC4	Finally leave DOS to idle routine at $9E81. Restore stack, A,Y,X. Save I/O hooks.
9FC5	9FC7	This is called to output the character in the accumulator to the output device (usually the screen).
9FC8	A17F	Reads keyboard, checks for command validity. If not valid start processing error via routine at $A0C9 to A6D2. Decodes ASCII from keyboard, converts to command code #, then if the command is valid, jumps to $A180 where command code number indexes table to appropriate address to execute oommand.
A180	A192	Command code number points to proper place in table $9D1F whioh has appropriate vectors for command, pushes command address on stack. Then jumps to that address by executing an RTS.
A193	A1A3	Look at keyboard buffer, oompare to <RETURN>
A1A4	A1AD	Compare keyboard character to $A0 (blank).
A1AE	A1B8	Put 0's into $B5BA-$B5C9.
A1B9	A1D5	Miscellaneous subroutine.
A1D6	A202	Decimal conversion.
A203	A228	Hexadecimal conversion.
A229	A22D	"PR#"
A22E	A232	"IN#"
A233	A23C	"MON"
A23D	A250	"NO MON"
A251	A262	"MAX FILES" - 3 on boot up - up to 16 files.
A263	A270	"DELETE".
A271	A280	LOCK/UNLOCK - software write protect and VERIFY.
A281	A297	"RENAME"
A298	A2A2	"APPEND"
A2A3	A2A7	"OPEN"
A2A8	A2E9	Command handler.
A2EA	A330	"CLOSE"
A331	A35C	"BSAVE"
A35D	A38D	"BLOAD"
A38E	A396	"BRUN"
A397	A3D4	"SAVE"
A3D5	A3DF	BLOAD, BSAVE, BRUN, routine affeoting only binary file operations.

(Continued)

Table 11 *(continued)*

A3E0	A412	Subroutines whioh work with BRUN, BLOAD and BSAVE.
A413	A479	"LOAD"
A47A	A4AA	"LOAD" assisting subroutine.
A4AB	A4D0	Miscellaneous subroutine.
A4D1	A4EF	"RUN"
A4F0	A50F	"CHAIN"
A510	A54G	"READ/WRITE" entry points.
A54F	A56D	"INIT"
A56E	A579	"CATALOG"
A57A	A590	Language deoode.
A59E	A5B1	"INT"
A5B2	A5C5	Set ROMs for Applesoft or integer BASIC.
A5C6	A5DC	"EXEC"
A5DD	A60D	"POSITION"
A60E	A671	Output to disk file routine.
A672	A69C	Get character from EXEC file.
A69D	A6A7	Make EXEC file currently open from $9615.
A6A8	A6C3	VERIFY.
A6C4	A719	Error processing, print errors.
A71A	A742	Called from $A2DE. Append transfer data to $B5BF to $B5C4.
A743	A74D	READ $AA75 - output string called by $A208 Append.
A74E	A75A	Modify $B5A9 - $B5AF.
A75B	A763	Turn in and out switches OFF from $9E1D.
A764	A791	Check output string starting at $AA75.
A792	A7A9	Part of clone routine from $A316, $A31B.
A7AA	A7AE	Part of clone from $A325, $A773.
A7AF	A7C3	Part of clone routine from $A2FC, $A320.
A7C4	A7D3	Check file type.
A7D4	A845	Initialize buffer from $9E0C.
A846	A850	Integer BASIC pointer setting.
A851	A883	Set I/O hooks. Branch from 3EA.
A884	A908	Command name table. NOTE: The last byte of each command name has the high (7th) bit set. The other bytes have it clear. (ex. 49 4E A3 = IN#). The search used is sequential ASCII codes (see Table 10). NOTE: this is where you can customize commands.
A909	A940	Parameter validity table. Checks validity of various parameters with various commands with 2 bytes (see Table 10).
A941	A94A	Parameter prefix names (VDSLRBACIO). (See Table 10).
A94B	A954	Parameter prefix bits (see Table 10).
A955	A970	Parameter prefix ranges (see Table 10). For the various prefix parameters (VDSLRBA is the normal order). This gives the range of possible values. First the lowest (minimum) value that is valid then the highest (maximum) valid value.
A971	AA3E	ASCII error message table.
AA3F	AA4E	Error message offsets.
AA4F	AAB7	Miscellaneous variables (ADDR) AA5F = COMMAND?
AAB8	AAC0	Name of FP basic file, "APPLESOFT".
AAC1	AAC8	4 miscellaneous addresses (RWTS IOB buffer) (VTOC buffer) (SYS buffer) (Top of RAM)
AAC9	AAE4	I/O package commands (see Table 6). This table is used at $AB14 to $AB1E to jump to the correct I/O routine. $B5BB is used to choose which I/O routine will be called.
AAE5	AAF0	READ commands (see Table 7). This table is used at $AC58 to $AC69 to jump to correct READ routine. The value of $B5BC is used to get the correct entry and a jump is made there.
AAF1	AAFC	WRITE commands (see Table 8). This table is used at $AC70 to AC86 to jump to the correct WRITE routine. The value of $B5BC is used to specify which routine will be jumped to
AAFD	B396	I/O package. Entry point for I/O package.
AAFD	AB1E	Chooses which I/O routine by using $B5BB to store I/O routine # to index through table at $AAC9.
AB1F	AC05	OPEN file.
AC06	AC39	CLOSE file.
AC3A	AC57	RENAME file.
AC58	AC69	READ from file - directs DOS to correct READ routine.
AC6A	AC86	WRITE to file - directs DOS to input WRITE routine.
AC87	ACB7	READ specific/next byte/block entry points.
ACB8	ACEE	WRITE specific/next byte/block entry points.
ACEF	AD11	LOCK/UNLOCK a file.
AD12	AD17	POSITION file ($B300).
AD18	AD2A	VERIFY file.
AD2B	AD97	DELETE file.
AD98	AE41	Print CATALOG. Takes file information from disk - sends in the buffer, then out to the screen.
	AE39	Responsible for pause during catalog listing.
AF06	B396	Various subroutines called by DOS commands.
B397	B6FF	Data area.

(Continued)

Table 11 *(continued)*

B397	B398	Track and sector address for the most recently read catalog sector.
B3A7	B3AE	T, I, A, B file type characters.
B3AF	B3BA	Character string "DISK VOLUME".
B3BB	B4BA	VTOC buffer, master track/sector bit map seotor or volume table of contents (VTOC).
B4BB	B5BA	System buffer - last accessed director sector. This last access may have been a catalog command or other DOS command requiring a directory search.
B5BB		1st byte beyond the system buffer. The routine in page 3 at $03DC loads Y and A registers to point here.
B5BC		Used to choose which I/O routine.
B5BD	B5FE	Miscellaneous data.
B600	B6B1	Relocation "boot up". Checks size of machine.
B700	B78C	General codes entry point for R.W.T.S. bootstrap routine.
B793	B7B4	Miscellaneous subroutine.
B7BF	B7C1	Miscellaneous subroutine.
B7C2	B7D5	Miscellaneous subroutine.
B7D6	B7DF	Miscellaneous subroutine.
B7DF	B7E7	Miscellaneous data.
B7E8	B7FF	I/O block.
B800	B829	New subroutine, prenibblize to write ($B800 3.2).
B82A	B8B7	New subroutine. "Write nibbles" (encode data).
	B86A	Reference addresses which actually control the disk interface device select address.
B8B8	B8C1	Routine at set mode.
B8C2	B8D8	Routine "post nibble" ($39C1 3.2) decode into real world data.
B8DC	B943	Routine "read nibbles" data ($B8FD 3.2).
B944	B99F	Routine "read next address field".
B9A0	B9FC	Routine to step R/W head up or down.
B9FD	BA10	Delay routine based on the last 2 bytes of DCT.
BA11	BA28	Data or buffer.
BA29	BA68	Encoded nibbles (64).
BB00	BC55	Buffer. Encoded nibbles to/from disk buffer.
BC56	BCC3	Motor running and on track. Format this track for INIT or writing to disk.
BCC4	BCDE	WRITE byte.
BCD5	BCDE	WRITE nibble.
BCDF	BCFF	Miscellaneous data.
BD00	BD18	STEP 1 - Determine new slot #. STEP 2 - ($BD17) If not same go to $BD19. If same, go to STEP 4 ($BD34).
BD19	BD33	STEP 3 - Not same slot, wait for motor to turn off.
BD34	BD63	STEP 4 - Check if motor is currently on, save results, STEP 5 ($BD4D). Turn motor on regardless of previous state.
BD64	BD84	STEP 6 - Determine if previous drive = current drive. If yes, go to STEP 8 $BD92.
BD85	BD8F	STEP 7 - If not same drive, WAIT for new drive to come up to speed, and set test result from STEP 4 to false ($BA00 relay delay loop x 7).
BD90	BD96	STEP 8 - Jump to track seek routine ($BE5A).
BD97	BD9D	STEP 9 - Was motor on in STEP 4? YES, go to STEP 11 ($BDAB). NO, go to STEP 10.
BD9E	BDAA	STEP 10 - wait for motor to come up to speed.
BDAB	BDB0	STEP 11 - if get "NULL" command, go to STEP 26 = ($BE0B) - (if no error turn off motor, if error occurred, indicate appropriate one). If not go to STEP 12 = ($BDB1).
BDB1	BDB4	STEP 12 - if given "FORMAT DISKETTE" command go to STEP 28 (- $BEAF). If not, go to STEP 13 ($BDB5).
BDB5	BDBB	STEP 13 - Given "WRITE SECTOR" command? If yes, prenibblize the data.
BDBC	BDC0	STEP 14 - Allow 48 retries for writing or reading process (RETRYCOUNT = 48).
BDC1	BDC6	STEP 15 = Read next field address.
BDC7	BDC8	STEP 16 - Was error encountered? NO = STEP 22 ($BDED). YES = CONTINUE to STEP 17.
BDC9	BDCB	STEP 17 - Decrement "RETRYCOUNT" (0578).
BDCC	BDCD	STEP 18 - $BDCC. Is RETRYCOUNT = 0. If no, go to STEP 15 - ($BDC1). (Read next address field, try again). If yes, jump to STEP 19.
BDCE	BDD1	STEP 19 - Find current track, save wanted track.
BDD2	BDD6	JSR SETTRK ($BE95). Recalibrate all over again.
BDD7	BDE0	Is $06F8 = 0? (Initially 02) YES go to STEP 27 ($BE04). Indicate error. No - Set SEEKCNT = 04 (0478).

Table 11 *(continued)*

BDE1	BDE9	STEP 20 - Recalibrate out to track 00. Then to desired track.
BDEA	BDEC	STEP 21 - Go back to 14 (try again).
BDED	BDF3	STEP 22 - Check if on right track. Yes - STEP 23 = $BE10
BDF4	BE03	No, not correct TRK, get desired track, JSR SETTRK, decrement SEEKCNT (from 040. If it has been less than 4 tries, go to TRKSEEK ROUTINE ($BE5A), then to STEP 14. If 4th time - RECAL. Then STEP 14.
BE04	BE0F	STEP 27 - (06F8 = 0). DRIVE ERROR, JMP to $BE48. Indicate error, turn off motor.
BE10	BE25	STEP 23 - CORRECT VOLUME? No - STEP 27 ($BE07), indicate error, turn off motor. Yes - go to STEP 24.
BE26	BE31	STEP 24 - Correct sector? No - STEP 17 ($BDC9) Yes - Go to STEP 25 and READ or WRITE.
BE32	BE37	STEP 25 - Do read or write operations.
BE38	BE50	Check for BAD READ, if bad - STEP 17 ($BDC9). STEP 26 - If ok. GIVE NO ERROR MESSAGE. Turn off motor. EXIT at $BE58.
BE51	BE59	WRITE subroutine.
BE5A	BE6A	TRACK SEEKING ROUTINE.
BE6B	BE80	Doesn't switch off all stepper motor lines, jump to $B9A0 to STEP R/W. Head up or down (update slot dependent locations with track.
BE8E	BE94	SET Y = SLOT #.
BE95	BEAE	"SETTRK" (Sets track location) (Slot dependent).
BEAF	BF0C	STEP 28 - FORMAT DISK routine.
BF0D	BFA7	RWTS miscellaneous read or write routine.
BFA8	BFB7	Blank.
BFB8	BFC7	Miscellaneous data.
BFC8	BFFF	Miscellaneous subroutines not connected with RWTS.

IMAGE PRINTER SERIES

Sensible Software is proud to introduce our new series of high resolution screen dumps. IMAGE PRINTERS provide a simple way to transfer high resolution graphic images onto paper. Each program in the series has unique features that give you full control of the printing. Some of the included options are:

- Full control over the area of the HIRES screen to be printed. You graphically pick the area for the utmost ease and accuracy.
- One-step printout of the picture with the ability to pause or abort the printing at any time.
- Menu-driven. All options are invoked with single keystrokes. IMAGE PRINTERS are extremely easy to use.
- Multiple image sizes, 6 different sizes for letter quality printers, 4 sizes for other printers.
- Creation of an inverse (negative) image for reverse printing.
- The ability to save the compressed and inverse images to disk.
- One time configuring for your printer and interface card. Why answer all those questions about your printer each time you want to print a picture?
- The images may be printed anywhere on the page.
- IMAGE PRINTERS support most popular interface cards, such as cards from Apple, California Computer Systems, Epson, and Mountain Computer. (The SSM AIO Serial Card and user-written 'driver' routines may be used with the letter quality printers.)

There are three separate versions of IMAGE PRINTERS, each one tailored to take full advantage of a different printer.

IMAGE PRINTER—LETTER QUALITY. For all popular letter quality printers (Diablo, NEC, Qume, etc.)

IMAGE PRINTER—EPSON. For the popular Epson MX-70, MX-80 and MX-100.

IMAGE PRINTER—NEC PC-8023A. For the NEC dot-matrix printer.

All versions are available for $40.00 ea.

Please specify version desired.

THE BUG

A sophisticated, yet easy to use diagnostic aid for getting "the bugs" out of your assembly language programs.

If you are a novice just getting started with assembly-language programming, you will find The BUG helpful in developing your understanding of how the Apple's 6502 internal processor operates. The many display options of The BUG will permit you to try out your assembly-language programs at the speed that is most comfortable for you. The BUG will also make it easy for you to see the effect of your program on the Apple as it executes.

If you are a professional programmer, you will also find that The BUG can improve your efficiency by reducing the time you spend identifying and solving complex, assembly-language programming errors. You will particularly appreciate the fact that The BUG offers the easiest to use and most extensive breakpointing capability of any "debugger" available for the Apple. Up to 13 different breakpoints can be specified to halt program execution when either: 1) a particular program location is reached, 2) one of the 6502 registers reaches a specified value, or 3) one of the bits in the 6502 status register reaches a specified value.

Another key feature of The BUG that serious programmers will appreciate is the ability to AUTOMATICALLY run lower-level subroutines at FULL SPEED. You no longer have to keep debugging the portions of your program that you already have working.

This is not the least expensive "debugger" program for the Apple, but we challenge you to find more capability for less money!

The BUG is supplied with a 40+ page user guide and is designed for use with DOS 3.3 on either the Apple II or Apple II Plus computer only **$50.00**

BUILD USING

. . . provides an easy to use print-using routine plus similar functions for strings. Creating charts, reports and general screen formatting becomes a simple task. BUILD USING is written entirely in machine language and provides a simple means of avoiding garbage collection (those unnecessary delays that slow down your programs). With BUILD USING, you can choose how many digits should be displayed to right and left of the decimal point, and you can even fill the leading positions with the character of your choice. For example, you can print the number '157.23' as '157.2', or '0000157.230', or '*****$157. AND 23/100 DOLLARS', or hundreds of other ways (including exponential formats). Working with strings is just as easy: it's a snap to convert names from 'John' and 'Doe' to 'Doe, J.'. Also included are three levels of error trapping, so you can trap and correct numbers or strings that cannot fit in your specified format.

Utilities like BUILD USING are usually difficult to use because they must be located in one memory location (usually between DOS and the DOS file buffers); they cannot be used with your favorite editor or other special routines. BUILD USING does not have this limitation, as it can be easily located in many different memory locations: 1) the "normal" between DOS and DOS file buffers, 2) at HIMEM, 3) APPENDED to your Applesoft program, or 4) anywhere else in memory. Appending BUILD USING to your program is as simple as EXECing a TEXT file. BUILD USING uses the "CALL" command thereby leaving the ampersand vector free for your own use.

BUILD USING requires Applesoft in ROM (Language cards are find), DOS 3.3 and a minimum of 32K only **$30.00**

Sensible Software

6619 Perham Drive Dept. M
West Bloomfield, Michigan
48033 • (313) 399-8877
Visa and Mastercard
Welcome
Please add $1 25
postage and
handling per
diskette

Converting Apple Pictures to a Standard Bit-Mapped Format

by David Lubar

This program will allow conversion of Apple II format high-resolution pictures to a standard bit-mapped format. It may be run on the source or destination machine.

Requires

Apple II or Apple II Plus
24K or greater without DOS
36K or greater with DOS

My first experience with data transfer between computers was in the form of digital entry [the digits being my typing fingers]. This method suffered both a low baud rate and a high error rate. Later, the appearance in MICRO of a program to write KIM tapes on the Apple improved matters. The latest step is in the form of several programs that allow the PET, MTU-130, and other 6502 machines to read and write Apple binary format. This format has become a sort of standard among the people with whom I swap data. Frank Covitz, who wrote the tape routines we use [see "PET Communicates with Apple" in this issue], actually transfers data from an MTU-130 to his PET using Apple format.

Once we were able to transfer data, the big question became what to transfer. At first, our choice was limited mostly to music data for the music interpreter Frank wrote. The music code is machine independent, and can go from one 6502 computer to another with no problem. Since Frank had the MTU Visible Memory in his PET, and high-resolution capability on the MTU-130, I decided to see if pictures done on the Apple hi-res screen could be translated into a standard bit-mapped format. As generated, an Apple picture won't fit a standard bit map in three different ways. First, the screen lines aren't stored in memory in contiguous order. The top line starts at 2000, the second at 2400, the third at 2800. After reaching 3C00, it goes to 2080. The second problem is that only seven of the eight bits map to the screen. The high bit is a color switch. Topping it off, on the Apple a byte is plotted on the screen with the low bit in the left-most position. Most bit maps plot each byte with the high bit to the left.

Of the three problems, two can be solved with lookup tables or calculations. The toughest part seemed to be packing eight Apple bytes, each with seven bits of information, into seven bytes with eight bits of information. After playing with the problem for a while, I found there was a simple algorithm for the required manipulation.

Let's look at the three translations, starting with the simplest.

Unmapping the Screen

There are two ways to access vertical locations on the Apple screen methodically. A base calculation can be done which will turn any Y coordinate into the correct address. This is the method used in the internal routines in Applesoft. A faster method is to use a table containing the low and high byte for the start of each screen line. Since the table is not that long, and contains a fair amount of repetition [making entry easy], I went with the table. Normally, the high byte entries point right to the hi-res screen. In the interest of machine independence, I subtracted the page number from the standard high byte table. This allows the user to put the source data at any page boundary. The page number is added to each high byte pulled from the table. To use the table, just put the desired screen line value in X and pull the entries out of the table, putting them into a pair of zero-page pointers. That takes care of unraveling the screen.

Mirror Image

Reversing a byte is not difficult. You just put the byte in the accumulator, then rotate the accumulator one way and rotate a byte of memory the other way. Do this eight times and the memory location will contain the reversed byte. The only thing to keep in mind is that the color switch that was originally the high bit has been flipped to the low position. [Note: if you have an application that requires quick reversal of bytes, this method might prove too slow. In such cases, use a lookup table containing reversed bytes stored in order of their original value. That way, the desired value can be used as an index into the table.]

Packing Them Down

This is the really interesting part. How do you shift everything over and drop the switch bit? The first thing I realized was that it would make sense to work with eight Apple bytes at a time. They would fit neatly into seven normal bytes. Some form of rotation would be required to achieve this. The end result would be the same as if you took one bit from the second byte and moved it to the first, then shifted the second byte over, took two bits from the third byte, and so on. Here's the method I used. First, assume there are eight bytes that haven't been reversed yet. Take the eight and perform a ROL. This knocks off the high bit. Now ROL bytes eight and seven. This moves a bit from the eighth byte to the seventh and removes the high bit from the seventh. Repeat the procedure with bytes eight, seven, and six, and so on until it has been done eight times. The result is that byte eight contains none of the original information, while bytes seven through one contain eight plotting pixels.

There is only one slight complication. The reversal has to be performed

Table 1

```
0900- 00 04 08 0C 10 14 18 1C
0908- 00 04 08 0C 10 14 18 1C
0910- 01 05 09 0D 11 15 19 1D
0918- 01 05 09 0D 11 15 19 1D
0920- 02 06 0A 0E 12 16 1A 1E
0928- 02 06 0A 0E 12 16 1A 1E
0930- 03 07 0B 0F 13 17 1B 1F
0938- 03 07 0B 0F 13 17 1B 1F
0940- 00 04 08 0C 10 14 18 1C
0948- 00 04 08 0C 10 14 18 1C
0950- 01 05 09 0D 11 15 19 1D
0958- 01 05 09 0D 11 15 19 1D
0960- 02 06 0A 0E 12 16 1A 1E
0968- 02 06 0A 0E 12 16 1A 1E
0970- 03 07 0B 0F 13 17 1B 1F
0978- 03 07 0B 0F 13 17 1B 1F
0980- 00 04 08 0C 10 14 18 1C
0988- 00 04 08 0C 10 14 18 1C
0990- 01 05 09 0D 11 15 19 1D
0998- 01 05 09 0D 11 15 19 1D
09A0- 02 06 0A 0E 12 16 1A 1E
09A8- 02 06 0A 0E 12 16 1A 1E
09B0- 03 07 0B 0F 13 17 1B 1F
09B8- 03 07 0B 0F 13 17 1B 1F
09C0- 00 00 00 00 00 00 00 00
09C8- 80 80 80 80 80 80 80 80
09D0- 00 00 00 00 00 00 00 00
09D8- 80 80 80 80 80 80 80 80
09E0- 00 00 00 00 00 00 00 00
09E8- 80 80 80 80 80 80 80 80
09F0- 00 00 00 00 00 00 00 00
09F8- 80 80 80 80 80 80 80 80
0A00- 28 28 28 28 28 28 28 28
0A08- A8 A8 A8 A8 A8 A8 A8 A8
0A10- 28 28 28 28 28 28 28 28
0A18- A8 A8 A8 A8 A8 A8 A8 A8
0A20- 28 28 28 28 28 28 28 28
0A28- A8 A8 A8 A8 A8 A8 A8 A8
0A30- 28 28 28 28 28 28 28 28
0A38- A8 A8 A8 A8 A8 A8 A8 A8
0A40- 50 50 50 50 50 50 50 50
0A48- D0 D0 D0 D0 D0 D0 D0 D0
0A50- 50 50 50 50 50 50 50 50
0A58- D0 D0 D0 D0 D0 D0 D0 D0
0A60- 50 50 50 50 50 50 50 50
0A68- D0 D0 D0 D0 D0 D0 D0 D0
0A70- 50 50 50 50 50 50 50 50
0A78- D0 D0 D0 D0 D0 D0 D0 D0
```

before packing the bits; otherwise the screen becomes rather garbled. Since the algorithm was designed to strip the high bit, and since some machines might also plot bits from low to high, I decided to maintain transportability by adding a step to the reversal procedure. After each byte is reversed, it undergoes one LSR. This kicks off the low bit (which was formerly the high bit color switch) and puts a dummy unused bit back at the high end. In this way, using the reversal portion as a separate module that can be left in or removed, depending on the mapping of the destination computer, the packing algorithm will always work.

Once the bytes have survived the above transformations, only one more step is necessary. The packing has left a gap every eight bytes. The picture data has to be shifted down. Once this is done, the information is in pure bit-mapped order. The following program is designed to put an Apple picture on a PET with the MTU Visible Memory. It can easily be adapted for other computers.

Apple to PET

The top routine, ZSAVE, merely saves the zero-page locations used by the program. Since 6502 machines vary in free locations, this approach can save a bit of grief. Next, the pointers for source and destination are set up. Note the LDA #$20 on the line labeled SETUP. This is the page number where the source data is stored. If you put the source at another location, just change the 20 to the correct page number.

Starting at the label MAIN, the program gets the location of a picture line from the table. The X register is used to keep track of the line being processed. Next, to simplify matters, the line is moved to a standard buffer. Since the buffer address is fixed, operations such as rotations and shifts are much less of a headache. The buffer is only 40 bytes long (this is the number of bytes on one line of Apple hi-res), so it can be put almost anywhere in memory.

With a line in the buffer, the next step is the SWAP section. This just reverses each byte and performs a shift to move the data back to the seven lower bits. If your display maps from low to high, delete this section.

The packing is performed next. Since the operation has to be performed on the eighth byte, then the eighth and seventh, and so on, a counter and pointer are required. The counter keeps track of the number of bytes being manipulated, the pointer shows which byte to rotate. When the counter has cycled from one through eight, the eight bytes have been packed. Since each line contains forty bytes, the operation has to be done five times.

Listing 1

```
0800              *APPLE TO PET VM GRAPHIC CONVERTER
0800              *USES LOOKUP TABLE FOR APPLE DATA
0800              *LOCATION $04 MUST HOLD PAGE
0800              *WHERE APPLE SCREEN IMAGE STARTS
0800              *
0800                        ORG $800
0800              *
0000              DEST      EPZ $00              ;POINTER TO PET VM DISPLAY
0002              SRC       EPZ $02              ;POINTER TO SOURCE, REQUIRES 8
K OF FREE RAM
0004              PAGE      EPZ $04              ;OFFSET FOR PAGE WHERE APPLE D
ATA IS STORED
0005              XSAVE     EPZ $05
0006              COUNTER   EPZ $06
0007              POINTER   EPZ $07
0B00              BUFFER    EQU $B00             ;STICK IT AFTER TABLE
0900              THI       EQU $900             ;HI BYTE LOOKUP TABLE
09C0              TLO       EQU $9C0             ;LO BYTE TABLE
9000              VM        EQU $9000            ;START OF VM MEMORY MAP
0800              *
0800              *SAVE ZERO PAGE VALUES
0800              *
0800 78           ZSAVE     SEI
0801 A2 08                  LDX #$08
0803 B5 00        ZLOOP     LDA $0,X
0805 48                     PHA
0806 CA                     DEX
0807 10 FA                  BPL ZLOOP
0809 A9 20        SETUP     LDA #$20             ;PAGE WHERE APPLE IMAGE IS STO
RED. CAN BE CHANGED
080B 85 04                  STA PAGE
080D A9 00                  LDA #VM
080F 85 00                  STA DEST             ;PET SCREEN
0811 A9 90                  LDA /VM
0813 85 01                  STA DEST+1
0815 A2 00                  LDX #$0              ;X COUNTS VERTICAL SCREEN LINE
S
0817 BD C0 09     MAIN      LDA TLO,X            ;POINT TO APPLE
081A 85 02                  STA SRC              ;IMAGE USING LOOKUP TABLE
081C BD 00 09               LDA THI,X            ;OFFSET BY PAGE WHERE
081F 18                     CLC                  ;DATA IS STORED
0820 65 04                  ADC PAGE
0822 85 03                  STA SRC+1
0824 86 05                  STX XSAVE
0826              *
0826              *FIRST, ONE SCREEN LINE IS MOVED
0826              *INTO A BUFFER AREA
0826              *
0826 A0 27                  LDY #$27             ;40 BYTES PER LINE
0828 B1 02        FILLBUFF  LDA (SRC),Y
082A 99 00 0B               STA BUFFER,Y
082D 88                     DEY                  ;BUFFER
082E 10 F8                  BPL FILLBUFF
0830              *
0830              *EACH BYTE HAS TO BE REVERSED
0830              *
0830 A2 27                  LDX #$27             ;40 BYTES AGAIN
0832 A0 08        SWAP      LDY #$08             ;HAVE TO DO 8 PAIRS OF ROTATIO
NS
0834 BD 00 0B               LDA BUFFER,X
```

(Continued)

Listing 1 *(Continued)*

```
0837 2A          SWLOOP   ROL
0838 7E 00 0B             ROR BUFFER,X
083B 88                   DEY
083C D0 F9                BNE SWLOOP
083E 5E 00 0B             LSR BUFFER,X         ;DROP OLD HI BIT AND PUT DUMMY
 AT HI END
0841 CA                   DEX
0842 10 EE                BPL SWAP
0844             *
0844             *FOLLOWING SECTION SHIFTS EIGHT
0844             *APPLE BYTES INTO 7 BIT-MAPPED BYTES. METHOD
0844             *IS TO ROL 8TH BYTE, THEN 8TH
0844             *AND 7TH, THEN 8TH, 7TH AND 6TH
0844             *AND SO ON
0844             *
0844 A9 07                LDA #$07             ;POINTS TO 8TH BYTE IN BUFFER
0846 85 07                STA POINTER
0848 A9 01       LOOP1    LDA #$01             ;NUMBER OF ITERATIONS
084A 85 06                STA COUNTER
084C A6 07       LOOP2    LDX POINTER
084E A4 06                LDY COUNTER
0850 3E 00 0B    LOOP3    ROL BUFFER,X         ;START MOVING BITS
0853 CA                   DEX
0854 88                   DEY
0855 D0 F9                BNE LOOP3            ;NOT FINISHED WITH PASS
0857 E6 06                INC COUNTER          ;ADD ANOTHER ITERATION
0859 A5 06                LDA COUNTER
085B C9 09                CMP #$09             ;ONLY DONE 8 TIMES
085D D0 ED                BNE LOOP2            ;NOT FINISHED
085F A5 07                LDA POINTER          ;DONE WITH A GROUP
0861 18                   CLC                  ;GET ANOTHER GROUP OF 8 BYTES
0862 69 08                ADC #$08
0864 85 07                STA POINTER
0866 C9 2F                CMP #$2F             ;ONLY WANT TO DO FIRST $27 BYT
ES
0868 D0 DE                BNE LOOP1            ;BACK FOR MORE
086A             *
086A             *BYTES ARE NOW SHIFTED, BUT
086A             *THERE IS A GAP EVERY 8 BYTES
086A             *NEXT STEP IS TO MOVE EVERYTHING
086A             *DOWN TO REMOVE THE GAPS
086A A2 00                LDX #$00
086C A0 00                LDY #$00
086E A9 07                LDA #$07             ;USED TO INDICATE WHICH
0870 85 06                STA COUNTER          ;BYTES TO SKIP
0872 B9 00 0B    LOOP4    LDA BUFFER,Y
0875 9D 00 0B             STA BUFFER,X
0878 E8                   INX
0879 C8                   INY
087A C4 06                CPY COUNTER          ;CHECK FOR THOSE TO SKIP
087C D0 F4                BNE LOOP4
087E C8                   INY                  ;SKIP OVER A BYTE
087F A9 08                LDA #$08             ;ADJUST COUNTER
0881 18                   CLC
0882 65 06                ADC COUNTER
0884 85 06                STA COUNTER
0886 C9 2F                CMP #$2F             ;NO NEED TO GO PAST $27
0888 D0 E8                BNE LOOP4
088A             *
088A             *40 APPLE BYTES ARE NOW SQUEEZE
088A             *DOWN TO 35 BIT-MAPPED BYTES
088A             *TIME TO PUT THEM ON THE SCREEN
088A             *
088A A6 05                LDX XSAVE            ;RESTORE VERTICAL POINTER
088C A0 22                LDY #$22             ;HAVE TO MOVE 35 BYTES
088E B9 00 0B    LOOP5    LDA BUFFER,Y
0891 91 00                STA (DEST),Y
0893 88                   DEY
0894 10 F8                BPL LOOP5
0896             *
0896             *LINE IS ON SCREEN
0896             *NOW THE DESTINATION HAS TO BE ADUSTED
0896             *
0896 A5 00                LDA DEST
0898 18                   CLC
0899 69 28                ADC #$28             ;MOVE UP 40 BYTES
089B 85 00                STA DEST
089D 90 02                BCC NEXT
089F E6 01                INC DEST+1
08A1 E8          NEXT     INX                  ;GET READY FOR NEXT LINE OF SO
URCE
08A2 E0 C0                CPX #$C0             ;APPLE SCREEN ENDS AT $BF
08A4 F0 03                BEQ ZBACK
08A6 4C 17 08             JMP MAIN
08A9             *
08A9             *RESTORE ZERO PAGE
08A9             *
08A9 A2 00       ZBACK    LDX #$0
08AB 68          ZBLOOP   PLA
08AC 95 00                STA $0,X
08AE E8                   INX
08AF E0 09                CPX #$09
08B1 D0 F8                BNE ZBLOOP
08B3 58                   CLI
08B4 60                   RTS
08B5             *TABLE STARTS ON NEXT PAGE BOUNDARY
08B5             *IT IS SHOWN AS A MEMORY DUMP, BUT NOT
08B5             *LISTED IN THIS SOURCE CODE
08B5                      END
```

This is achieved by adding eight to the pointer and resetting the counter to one. Once the program reaches the end of this section, the line has been packed.

After this, the gaps have to be handled. The counter is now used to determine which bytes to skip. On the first pass through the routine labeled LOOP4, X and Y are equal. In essence, a byte is just taken from its location and put back there. When the value of Y is equal to the value in the counter, it is time to skip over a byte. Once this is done, the counter is increased to the next value to be skipped, and a check is made to see if the counter has gone past the end of the buffer.

Finally, the line is read to be put on the destination screen. This is performed in the area of the label LOOP5. Once the line is on the screen, the destination value and the pointer to the source line have to be adjusted. One of the joys of normal bit mapping is that the next line can be accessed with a simple addition. In the case of the PET VM, each line contains 40 bytes. Add 40 to the present pointer and you are on the next line. (Note that this code places the picture starting in the upper left edge of the screen. To center the image, the starting value of the destination has to be adjusted. To do this for any bit map, take the horizontal byte size, subtract 35 (the number of bytes in the packed Apple line), then divide by two. This gives the offset from the left edge. Add this value to the start of the destination and the picture will be centered horizontally. To center vertically, subtract 192 from the vertical resolution of your bit map, multiply by the number of bytes in a line, and divide by two. Add this number to the starting location for vertical centering).

Once the destination has been adjusted, the source pointer is incremented and checked against the top value. The Apple screen has 192 lines (numbered from $00 to $BF). When X equals $C0, the program is finished.

Using the Program

One feature of this program is that it can be used on either the source or destination computer. If you have an Apple and want to generate a binary file that will load directly into the bit map of a different computer, use the following steps. Load the hi-res picture into page one of hi-res. (Note to non-Apple users: this confusing bit of terminology has nothing to do with page one of memory. Page one of hi-res is at $2000, page two at $4000.) Set the destination to $4000 and leave the page pointer at $20. This can be accomplished by changing $812 from $90 to $40 (or, if you are reassembling the program, by changing line 17 to VM EQU $4000). When the program is finished, the translated data will be in memory from $4000 to $5E00. This data can be loaded directly into the memory map of another computer.

To use the program on the destination machine, load the program into $800. (If it requires relocation, just change the value of the JMP near the end and the location of the buffer and table.) Next, put the Apple hi-res data into memory starting at any page boundary. Eight K of free space is required. If the source is stored somewhere other than $2000, change the value in the line labeled SETUP to the correct page of memory. Then, if your hardware requires, enable the visible memory. A few seconds after you run the program, the picture will be on your screen.

Though the Apple uses color, and pure bit maps are monochrome, the picture will contain a nice assortment of gray scales, especially if the original image made use of some of the dithered colors that can be produced on the Apple.

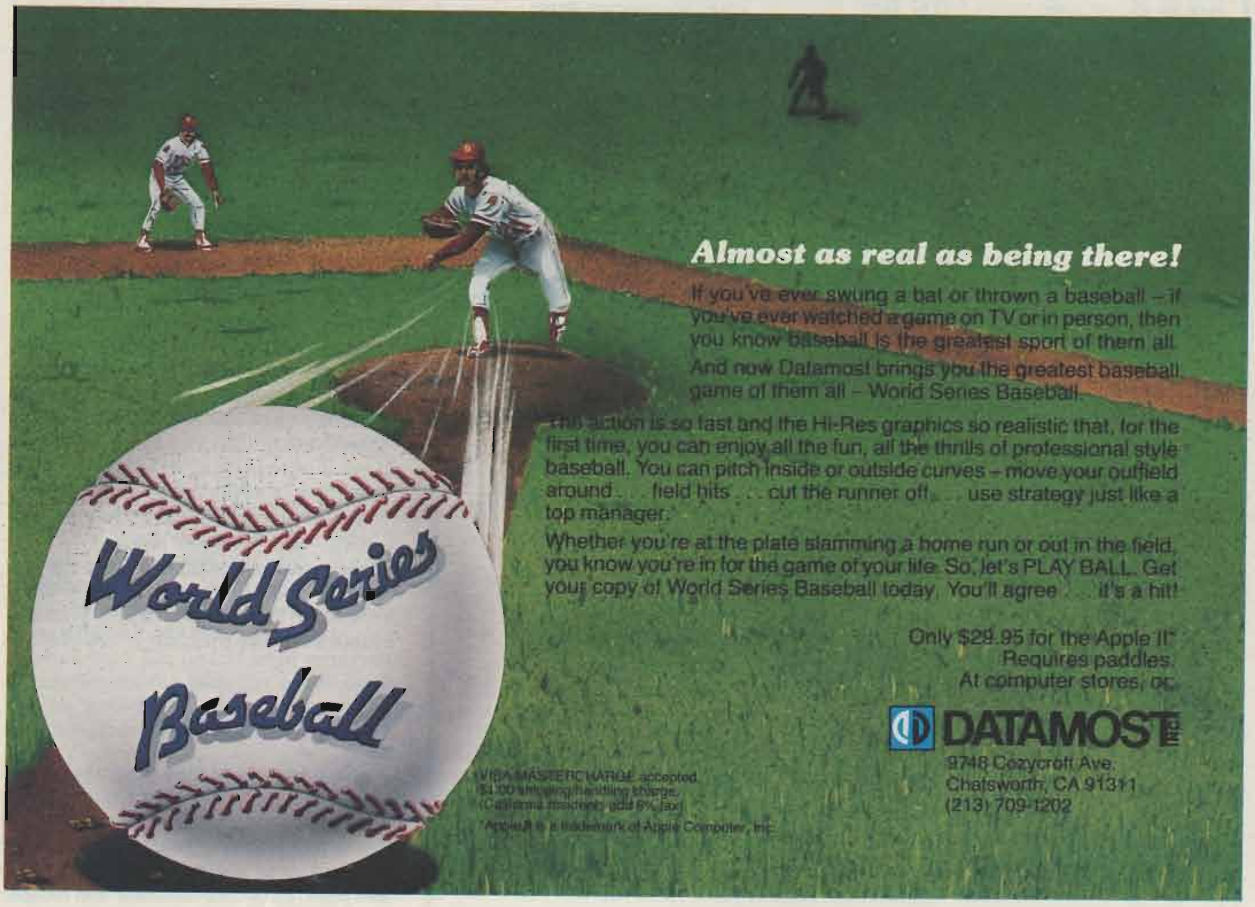
This method should also work if you want to put an Apple picture on the Atari. Use GRAPHICS 8. Again, you can either translate the picture while it is in the Apple, and then load it above the display list, or put the source into the Atari and move it up into display memory.

If your machine is unable to read Apple tapes, data can still be transferred using a modem or other form of interface. For those of you with a PET and the MTU VM, the routine written by Frank Covitz to read Apple tapes is included in this issue. The routine consists of a basic driver and a machine-language routine that simulates Apple tape input on the PET. To use it, enter the start and end address of the tape data in hex. The routine will read the tape and indicate whether the load was good.

As an interesting exercise, you might want to try reversing the procedure, producing an Apple image from a standard bit map. That way we can all swap pictures.

David Lubar may be contacted at 1809
Cedarwood Drive, Piscataway, NJ 08854.

MICRO™



Commodore Computers Bloom with CP/M®, Hard Disk and Multi-User Capability.

COMMODORE DEALERS

Expand your customer base by including the vast range of CP/M software development and applications!

Small Systems Engineering provides everything you need to enter the CP/M marketplace. Excellent dealer margins. Advertising and marketing support. CP/M software on Commodore formats. Corvus products. Watch for innovative products from SSE to support your Commodore Computer line.

Now your Commodore can blossom into a full-blown CP/M-based or multi-user business system with the addition of one of these innovative peripherals. Your Commodore plus the SOFTBOX gives you a single-user CP/M system. Commodore plus HARDBOX puts Pet software on Corvus Disk Drives and multiplexing units.

RUN CP/M WITH SOFTBOX ON FLOPPY AND HARD DISK.

Add SOFTBOX. The Z80-based computer which connects to your Commodore PET or CBM through the IEEE bus.

Now you have the ability to run any CP/M software or application, giving you a universe of computing capabilities you could only wish for until now.

Simply by plugging the SMALL SYSTEMS SOFTBOX into your Commodore Computer's IEEE-488 port and loading the CP/M disk, your computer will run under the world's most popular disk operating system. No internal connections or modifications to your computer are necessary.

Applications packages designed to work with specific terminals (like Lear Siegler ADM3A, Televideo 912 or Hazeltine 1500) need no modification to work with your computer's screen, since the SOFTBOX allows your computer to emulate any of these devices.

Interfaces that come with your SOFTBOX let you add the Corvus Hard Disk and any RS-232C peripheral. You can run a printer, and even talk to another CP/M based system.

90-Day Warranty. Suggested retail $895
Complete line of CP/M software available.

HARDBOX GIVES PET DOS MULTI-USER AND HARD DISK CAPABILITY.

The HardBox introduces your PET software to the world of high-capacity, high speed disk storage. It acts as an intelligent controller for up to four Corvus Winchester disk drives. Each Corvus drive can have a capacity of 6, 10 or 20 million bytes.

The HardBox is designed to be software-compatible with PET DOS version 1 or 2, so that it will work with existing programs. The HardBox is designed to appear to the PET as a fast, high-capacity floppy disk unit.

When used with the Corvus range of drives, the HardBox provides much more than just a hard disk interface. By using an 8-way multiplexing unit, up to eight PET users, each equipped with a HardBox, may simultaneously address the same hard disk. By using two levels of multiplexing, up to 64 users may be accommodated. The hard disk may be divided logically into user areas.

A comprehensive set of utility programs are provided with the hardbox for file transfer, hard disk configuration, mirror backup, system maintenance etc.

90-Day Warranty. Suggested retail $695.

SSE

SMALL SYSTEMS ENGINEERING
222 B View Street
Mountain View, CA 94041
(415) 964-8201

CP/M is a registered trademark of Digital Research.
© 1982 Small Systems Engineering Corp.

PET Communicates with Apple

by Frank Covitz

This article describes how the Apple II and PET can communicate using the Apple II cassette format.

Requires

PET with BASIC - 3.0

The Commodore PET and the Apple use the same microprocessor, the excellent 6502, running at *ca.* 1 MHz. As such, you might think it would be relatively easy to transport data from the one to the other. Alas, this is not the case! Of course, data could be transferred *via* modems or by physically connecting the two through some sort of I/O port, but this would limit the applicability to those who had the required hardware. Fortunately, there is a simpler (and cost-free) way: through the cassette port.

The programs given in this article are to be used within the Commodore machine, and allow the PET to write and read Apple binary tapes. With binary format, any type of data can be transmitted, including ASCII as well as machine code. The tape format is such that the total number of bytes must be known (so that the checksum will come out right). The data to be written to tape must be in the PET's memory (I conveniently start data at $2000), and conversely, on read, the start and end addresses must be specified. (The start address is arbitrary, but the total number of bytes to be read must agree with what is on the tape.)

I have routinely dumped both machine-code programs and data as well as ASCII text with no hitches. However, on read, it appears to be necessary to start the tape a few seconds into the leader portion. Obviously, this system could work quite well without the actual invervention of tape, if the two machines can be brought side-by-side, and an electrical hook to the tape port on the PET is made.

Listing 1

```
100 SA=1749:EA=SA+2:CH=SA+4:A=6*256
110 INPUT"START ADDRESS(HEX)";A$
120 GOSUB200:POKE SA,L%:POKE SA+1,H%
130 INPUT"END ADDRESS(HEX)";A$
140 GOSUB200:POKE EA,L%:POKE EA+1,H%
180 SYS(A)
181 A$="GOOD"
182 IF PEEK(CH)=255THENA$="BAD"
183 PRINTA$+" LOAD"
190 END
200 D=0:FORL=0TOLEN(A$)-1:B$=MID$(A$,LEN(A$)-L,1):B=ASC(B$)
210 IFB>47ANDB<58THENB=B-48
220 IFB>64THENB=B-55
230 D=D+B*16↑L:NEXTL
240 H%=D/256:L%=D-256*H%:RETURN
```

***Listing 2:* Appleread**

```
0800                  ;
0800                  ; APPLEREAD
0800                  ;
0800                  ;
0000                  A1        EPZ $00           ;ADDRESS POINTER
0002                  A2        EPZ $02           ;ENDAD POINTER
0004                  CHKSUM    EPZ $04
00D4                  DEVICE    EPZ $D4           ;CURRENT DEVICE
00F9                  MOT1      EPZ $F9           ;MOTOR STATUS
E810                  PIAL      EQU $E810
E811                  CRA       EQU $E811
F812                  STTAPE    EQU $F812
FCA8                  MOTOFF    EQU $FCA8
0005                  CRCT      EPZ $05
0000                  ZERO      EPZ $00
0600                            ORG $600

0600                  ;
0600                  ;
0600 A9 01            START     LDA #1            ;TURN ON MOTOR #1
0602 85 D4                      STA DEVICE
0604 20 12 F8                   JSR STTAPE
0607 78               STRT1     SEI               ;STOP INTERRUPTS
0608 20 C4 06                   JSR DELAY         ;DELAY 3.5 SEC
060B A2 06                      LDX #6
060D B5 FF            STRT2     LDA $FF,X         ;SAVE LOC'NS 0-5
060F 48                         PHA               ;ON STACK
0610 CA                         DEX
0611 D0 FA                      BNE STRT2
0613 AD D5 06                   LDA STAD          ;INITIALIZE POINTERS
0616 85 00                      STA A1
0618 AD D6 06                   LDA STAD+1
061B 85 01                      STA A1+1
061D AD D7 06                   LDA ENDAD
0620 85 02                      STA A2
0622 AD D8 06                   LDA ENDAD+1
0625 85 03                      STA A2+1
0627 AD 11 E8                   LDA CRA           ;SAVE CONTROL REG
062A 48                         PHA
062B 29 C6                      AND #%11000110    ;FORCE BITS 5430 TO 0
062D 09 02                      ORA #%00000010    ;FORCE BIT 1 HIGH
062F 8D 11 E8                   STA CRA
0632 2C 10 E8                   BIT PIAL          ;RESET CA1
0635 20 AE 06         READ      JSR RDBIT         ;FIND AN EDGE
0638 A9 FF                      LDA #$FF
063A 85 04                      STA CHKSUM
063C 20 AB 06                   JSR RD2BIT        ;GET BIT
063F A2 64            RD0       LDX #100          ;GET AT LEAST 100 SYNC'S IN A
ROW
0641 A0 1E            RD1       LDY #$23-CRCT
0643 20 AE 06                   JSR RDBIT
0646 90 F7                      BCC RD0           ;NOT SYNC, START OVER
0648 CA                         DEX
0649 D0 F6                      BNE RD1
```

The assembly source below and simple BASIC drivers were created by close examination of the Apple ROM source code (from the "red book"), with particular analysis of the timing, since all are derived *via* software loops. Since the code is essentially Apple copyrighted, permission for this article was granted by Apple Corp.

When used with the BASIC driver, the prompts for start and end addresses are given, and on load the system returns with "GOOD LOAD" if the checksums agree. Any "BAD LOAD" messages I have gotten were generally the result of giving the wrong addresses. So, if you send someone an Apple binary tape, be sure you give the start and end addresses of the data on the tape. *Note:* this should be START ADDRESS, END ADDRESS *not* START ADDRESS, END ADDRESS + 1.

I have used the same scheme for transferring data from KIM to Apple. The KIM can only conveniently write Apple tapes; because of the KIM's PLL circuitry for tape input, it cannot read them. Since the PET can read it and the KIM can write it, I have frequently used Apple tape format to transfer KIM data to the PET — without an Apple in sight!

Frank Covitz may be contacted at Deer Hill Road, Lebanon, NJ 08833.

Listing 2 *(Continued)*

```
064B A0 1F      RD2     LDY #$24-CRCT     ;LOOK FOR SYNC'S
064D 20 AE 06           JSR RDBIT         ;EDGE ONLY
0650 B0 F9              BCS RD2           ;LOOP ON SYNC'S
0652 20 AE 06           JSR RDBIT         ;SKIP NEXT HALF CYCLE
0655 A0 36              LDY #$3B-CRCT     ;INDEX FOR 0/1 TEST
0657 20 9D 06   RD3     JSR RDBYTE        ;GET A BYTE
065A 81 00              STA (A1,X)        ;AND STASH IT
065C 45 04              EOR CHKSUM        ;MAINTAIN CHECKSUM
065E 85 04              STA CHKSUM
0660 20 8E 06           JSR NXTA1         ;UPDATE ADDRESS
0663 A0 30              LDY #$35-CRCT     ;COMPENSATE
0665 90 F0              BCC RD3           ;LOOP UNTIL DONE
0667 20 9D 06           JSR RDBYTE        ;READ CHECKSUM
066A A0 00              LDY #0            ;DEFAULT Y=0(GOOD)
066C C5 04              CMP CHKSUM
066E F0 02              BEQ *+4
0670 A0 FF              LDY #$FF          ;Y=$FF MEANS BAD
0672 68                 PLA               ;RESTORE CRA
0673 8D 11 E8           STA CRA
0676 A2 00              LDX #$0           ;RESTORE LOC'NS 0-5
0678 68         RESTOR  PLA
0679 95 00              STA ZERO,X
067B E8                 INX
067C E0 06              CPX #6
067E D0 F8              BNE RESTOR
0680 8C D9 06           STY ERROR         ;SAVE ERROR COND
0683 A9 01              LDA #1
0685 85 F9              STA MOT1
0687 A9 3D              LDA #$3D
0689 20 A8 FC           JSR MOTOFF        ;TURN OFF MOTOR
068C 58                 CLI
068D 60                 RTS
068E            ;
068E            ;  SUBROUTINES
068E            ;
068E A5 00      NXTA1   LDA A1
0690 C5 02              CMP A2
0692 A5 01              LDA A1+1
0694 E5 03              SBC A2+1          ;SET CARRY ON DONE
0696 E6 00              INC A1            ;UPDATE ADDRESS
0698 D0 02              BNE *+4
069A E6 01              INC A1+1
069C 60                 RTS
069D A2 08      RDBYTE  LDX #8
069F 48         RDBYT2  PHA
06A0 20 AB 06           JSR RD2BIT        ;READ TWO TRANSACTIONS
06A3 68                 PLA               ;FORM THE BYTE VIA CARRY FLAG
06A4 2A                 ROL
06A5 A0 35              LDY #$3A-CRCT     ;TIMING
06A7 CA                 DEX
06A8 D0 F5              BNE RDBYT2        ;DO 8 BITS
06AA 60                 RTS
06AB 20 AE 06   RD2BIT  JSR RDBIT
06AE 88         RDBIT   DEY               ;(2 STATES)
06AF 24 04              BIT CHKSUM        ;(WASTE 3 STATES)
06B1 2C 11 E8           BIT CRA           ;(4 STATES) WAIT FOR TRANSITIO
N
06B4 10 F8              BPL RDBIT         ;(3,2 STATES)
06B6 AD 11 E8           LDA CRA           ;(4) FLIP ACTIVE CA1 COND
06B9 49 02              EOR #%00000010    ;(2)
06BB 8D 11 E8           STA CRA           ;(4)
06BE 2C 10 E8           BIT PIAL          ;(4) RESET CRA BIT 7 .
06C1 C0 80              CPY #$80          ;(2) SET CARRY ON Y
06C3 60                 RTS               ;(6)
06C4 20 C7 06   DELAY   JSR DLY0          ;DELAY CA 2.2 SEC
06C7 A2 00      DLY0    LDX #0            ;DELAY 1.1 SEC
06C9 A0 00      DLY1    LDY #0
06CB 20 D4 06   DLY2    JSR DLY3          ;WASTE 12 STATES
06CE 88                 DEY               ;(2)
06CF D0 FA              BNE DLY2          ;(3)
06D1 CA                 DEX
06D2 D0 F5              BNE DLY1
06D4 60         DLY3    RTS               ;(3)
06D5            ;
06D5            ;RAM STORAGE
06D5            ;
06D5 00 00      STAD    DBY 00            ;DUMMY START ADDRESS
06D7 07 FF      ENDAD   DBY 00+$7FF       ;DUMMY END ADDRESS
06D9            ERROR   EQU *             ;0=GOOD LOAD, $FF=BAD
06D9                    END
```

Listing 3

```
10 SA=12*16+9:EA=SA+2:A=6*256
20 INPUT"START ADDRESS(HEX)";A$
30 GOSUB100:POKE SA,L%:POKE SA+1,H%
40 INPUT"END ADDRESS(HEX)";A$
50 GOSUB100:POKE EA,L%:POKE EA+1,H%
60 INPUT"TAPE DRIVE 1 OR 2";D
70 IF D=2 THEN A=A+3
80 SYS(A)
90 END
100 D=0:FORL=0TOLEN(A$)-1:B$=MID$(A$,LEN(A$)-L,1):B=ASC(B$)
110 IFB>47ANDB<58THENB=B-48
120 IFB>64THENB=B-55
130 D=D+B*16↑L:NEXTL
140 H%=D/256:L%=D-256*H%:RETURN
```

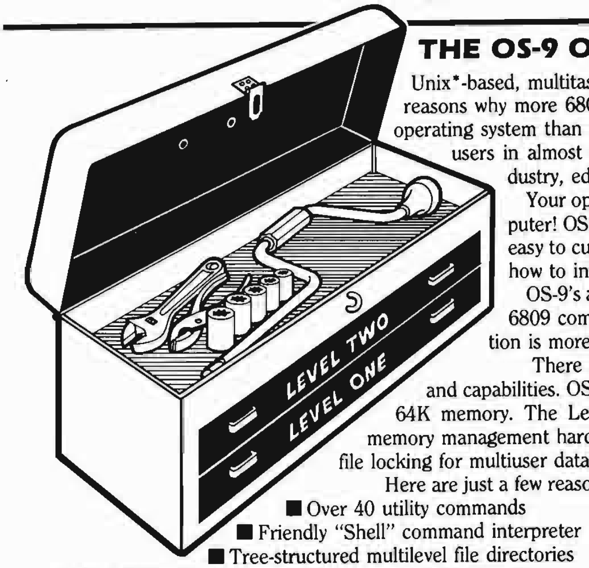
(Listing 4 on next page)

(Continued on page 51)

Listing 4

```
0800                ; APPLEWRITE
0800                ;
0800                ;
00C9                CURAD    EPZ $C9
00CB                ENDAD    EPZ $CB
E842                DDRB     EQU $E842
E840                PORTB    EQU $E840
F847                PROMPT   EQU $F847
FCA8                MOTOFF   EQU $FCA8
00D4                DEV      EPZ $D4
00F9                MOT1     EPZ $F9
00FA                MOT2     EPZ $FA
0008                TPBIT    EPZ %00001000
0800                ;
0600                         ORG $0600
0600 A9 01          START    LDA #1              ;DEFAULT TO TAPE #1
0602 2C                      BYT $2C             ;DUMMY BIT INSTRUCTION
0603 A9 02          START2   LDA #2              ;ENTRY FOR TAPE #2
0605 85 D4                   STA DEV             ;PUT IN DEVICE #
0607 20 47 F8                JSR PROMPT          ;PLAY & RECORD
060A 78                      SEI                 ;PREVENT INTERRUPTS
060B AD 42 E8                LDA DDRB            ;FORCE BIT 3 AS OUTPUT
060E 09 08                   ORA #TPBIT
0610 8D 42 E8                STA DDRB
0613 A9 08                   LDA #TPBIT
0615 49 FF                   EOR #$FF            ;FORCE A LOW AT START
0617 2D 40 E8                AND PORTB
061A 8D 40 E8                STA PORTB
061D 20 37 06                JSR WRITE           ;NOW WRITE THE TAPE
0620 A9 08                   LDA #TPBIT
0622 49 FF                   EOR #$FF            ;FORCE A LOW AT END
0624 2D 40 E8                AND PORTB
0627 8D 40 E8                STA PORTB           ;TURN OFF TAPE MOTOR
062A A9 01                   LDA #1              ;SET MOTOR STATUS
062C 85 F9                   STA MOT1
062E 85 FA                   STA MOT2
0630 A9 3D                   LDA #$3D
0632 20 A8 FC                JSR MOTOFF          ;TURN OFF TAPE MOTOR
0635 58                      CLI
0636 60                      RTS
0637 A9 40          WRITE    LDA #$40            ;WRITE 10 SEC. HEADER
0639 20 5F 06                JSR HEADR
063C A0 25                   LDY #$27-2          ;TIMING?
063E A2 00          WR1      LDX #0
0640 41 C9                   EOR (CURAD,X)       ;CHECKSUM
0642 48                      PHA                 ;SAVE CHECKSUM
0643 A1 C9                   LDA (CURAD,X)       ;GET BYTE FROM MEMORY
0645 20 56 06                JSR WRBYTE          ;PUT ON TAPE
0648 20 89 06                JSR NXTAD           ;ADVANCE INDEX
064B A0 1B                   LDY #$1D-2          ;TIMING?
064D 68                      PLA                 ;RECALL CHECKSUM
064E 90 EE                   BCC WR1             ;CARRY IS CLEAR UNTIL END
0650 A0 20                   LDY #$22-2          ;TIMING?
0652 20 56 06                JSR WRBYTE          ;WRITE CHECKSUM
0655 60                      RTS
0656                ;
0656                ; ** WRITE A BYTE TO TAPE
0656                ;
0656 A2 10          WRBYTE   LDX #$10            ;DO 16 BITS
0658                ;WRBYT2 ASLA ;SHIFT HIGH BIT TO CARRY
0658 0A             WRBYT2   ASL                 ;SHIFT HIGH BIT TO CARRY
0659 20 6C 06                JSR WRBIT           ;AND WRITE TO TAPE
065C D0 FA                   BNE WRBYT2          ;RESULT OF 'DEX'
065E 60                      RTS                 ;GO BACK FOR NEXT BYTE
065F                ;
065F                ; ** WRITE HEADER **
065F                ;
065F A0 49          HEADR    LDY #$4B-2          ;TIMING?
0661 20 71 06                JSR ZERDLY          ;650 USEC
0664 D0 F9                   BNE HEADR
0666 69 FE                   ADC #$FE
0668 B0 F5                   BCS HEADR
066A A0 1F                   LDY #$21-2          ;TIMING?
066C 20 71 06       WRBIT    JSR ZERDLY          ;ZERO BIT
066F C8                      INY
0670 C8                      INY
0671 88             ZERDLY   DEY                 ;DELAY LOOP
0672 D0 FD                   BNE ZERDLY
0674 90 05                   BCC WRTAPE          ;CARRY STATUS SET PREVIOUSLY
0676 A0 30                   LDY #$32-2
0678 88             ONEDLY   DEY                 ;ONE BIT
0679 D0 FD                   BNE ONEDLY
067B                ;
067B                ; ** WRITE TO TAPE HERE **
067B                ;
067B                ; MUST TOGGLE TAPE OUTPUT BIT
067B                ; A,X AND CARRY MUST BE PRESERVED
067B                ;
067B 48             WRTAPE   PHA                 ;SAVE A
067C AD 40 E8                LDA PORTB
067F 49 08                   EOR #TPBIT          ;FLIP BIT 3
0681 8D 40 E8                STA PORTB
0684 68                      PLA                 ;FETCH A
0685 A0 2A                   LDY #$2C-2          ;TIMING?
0687 CA                      DEX
0688 60                      RTS
0689                ;
0689                ; ** ADVANCE POINTERS **
0689                ;
0689 A5 C9          NXTAD    LDA CURAD
068B C5 CB                   CMP ENDAD           ;CHECK CURAD=ENDAD
068D A5 CA                   LDA CURAD+1
068F E5 CC                   SBC ENDAD+1         ;CLEVER USE OF CARRY
0691 E6 C9                   INC CURAD
0693 D0 02                   BNE NXT1
0695 E6 CA                   INC CURAD+1
0697 60             NXT1     RTS                 ;CARRY SET AT CURAD=ENDAD
0698                         END
```

Build performance into your system using the OS-9™ toolbox!



THE OS-9 OPERATING SYSTEM

Unix*-based, multitasking, modular, and versatile: these key features are some of the reasons why more 6809 computer manufacturers have selected OS-9 as their standard operating system than any other O.S. And OS-9 has been put to work by thousands of users in almost every conceivable computer application in business, science, industry, education, and government.

Your operating system should not be a barrier between you and your computer! OS-9 is very friendly and easy to use. Due to its modular structure it's easy to customize, plus its comprehensive documentation shows you exactly how to interface it to just about any I/O device.

OS-9's advanced features unleash the performance potential of almost any 6809 computer—large or small. In many respects the OS-9/6809 combination is more powerful than many *minicomputers!*

There are two basic versions of OS-9. Both have the same basic features and capabilities. OS-9 Level One runs on small to medium sized systems having up to 64K memory. The Level Two version runs on medium to large size systems having memory management hardware and up to 1 megabyte of memory, and includes record and file locking for multiuser database applications.

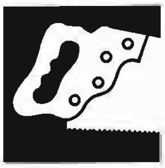
Here are just a few reasons why you should insist on OS-9 for your microcomputer system.

- Over 40 utility commands
- Friendly "Shell" command interpreter
- Tree-structured multilevel file directories
- Full timesharing support with log-in and file security
- Fast, secure random and sequential access files
- Comprehensive English language error messages
- Compact real-time multitasking executive
- Hardware or software memory management
- Device independent interrupt-driven I/O
- Fully ROMable for small control systems
- Standard versions available from manufacturers of most popular 6809 computers

OS-9 PASCAL™ LANGUAGE COMPILER

The OS-9 Pascal language compiler is the most complete and versatile Pascal available for the 6809. OS-9 Pascal has the unusual capability of generating P-code for interpretive execution while debugging OR highly optimized 6809 assembly language source code output for maximum speed. Another feature of OS-9 Pascal is its "virtual memory" P-code interpreter that lets you run incredibly large Pascal programs. OS-9 Pascal meets the ISO 7185.1 Standard and the complete Wirth/Jensen specification.

CIS COBOL™ COMPILER



6809 CIS Cobol is a compact, interactive and standard Cobol language compiler which is ideal for the most demanding business applications. Standard features are: ISAM, Debug, ACCEPT/DISPLAY, and Interprogram Communications modules. CIS Cobol is *the* preeminent microcomputer Cobol in the industry, and the OS-9 version retains full compatibility with CP/M applications software, CIS Cobol meets the ANSI 1974 Level One COBOL standard and is GSA certified. Also available is Micro Focus' FORMS 2, an optional automatic program generator that lets you interactively design screen-oriented applications with ease.

BASIC09™ STRUCTURED BASIC INTERACTIVE COMPILER

Basic09 is the fastest and most comprehensive full Basic language available for the 6809. It combines standard Basic with the best features of Pascal. It is a unique interactive compiler that combines compiler speed, interpreter friendliness, and superlative debugging facilities. RunB, a ROMable run-time system for compiled Basic09 programs is now available as an option.

C LANGUAGE COMPILER

C—the systems language of the future—is here today on OS-9. This is a complete implementation of the Unix Version 7 C language including INT, CHAR, SIGNED, UNSIGNED, FLOAT and LONG data types, structures, unions, standard C library, and a full preprocessor with macro definitions. Generates fully reentrant 6809 assembly language source code output.

For information contact your computer supplier, or

MICROWARE®

Microware Systems Corporation
5835 Grand Avenue, Des Moines, Iowa 50312
Phone 515-279-8844 • Telex 910-520-2535

*Unix is a trademark of Bell Laboratories. CIS Cobol is a trademark of Micro Focus, Inc. OS-9 and Basic09 are trademarks of Microware and Motorola, Inc.

EPROM PROGRAMMER FOR PET AND ATARI COMPUTERS

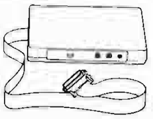
The BRANDING IRON is an EPROM programmer especially designed for PET and ATARI computers. Programs 2716 and 2532 type EPROMs. The PET version plugs into the cassette and I/O port and comes with software which adds the programmer commands to the PET monitor. The ATARI version plugs into controller jacks and comes with a full fledged machine language monitor which provides 30 commands for interacting with the computer and the BRANDING IRON.

PET – $75.00

ATARI – $119.95

CONVERT YOUR PET INTO A TERMINAL $129.95

RS232 Hardware and cable, and sophisticated terminal software. Upload and Download, communicates in ASCII, status line, built-in file translator. A complete package, all you need is a modem and we sell them too.



Need a Modem?
We sell the Signalman Direct Connect Modems for $89.50 (includes cable)

Eastern House
3239 Linda Dr.
Winston-Salem, N C 27106
(919) 924-2889 (919) 748-8446

CSE means OSI

Custom After Market Software for C1P and C4P machines

*Basic Enhancer:
Renumber, Auto Sequencer, Screen Control functions, and tape 1/0 system that is faster and has file names

C1P	$21.95
C4P	$29.95

Modified Monitor Rom Chip:
Now get indirect jump-capabilities just like those in the C1P and for no extra charge CSE will burn in your machines serial number $16.95

*NOTE: The C4P version of the Basic Enhancer includes the modified monitor Rom chip required for proper program functioning.

This is only a partial listing of our products. Write us for information on new disk programs or send $2 for catalog. Please include $2.00 shipping and handling with orders.

Computer Science Engineering
Box 50 • 291 Huntington Ave. Boston 02115

Olympic Sales Company

SERVING YOU SINCE 1947

CABLE: "OLYRAV" LSA
Main Showroom & Offices:
216 So. Oxford Avenue
(P. O. Box 74545)
Los Angeles, CA 90004
Telex: 67 34 77

WE HONOR VISA and MASTERCHARGE

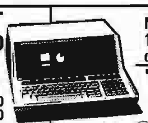
ORDER DESKS
open 6 days a week!
7:00 am to 7:00 pm Mon – Sat
Order Desks: (213) 739-1130

TOLL-FREE	TOLL-FREE
(outside Calif.)	(within Calif.)
800-421-8045	800-252-2153

Prices shown are for mail or phone orders; Walk-in slightly higher.

Goods subject to availability; OSC will meet or beat almost any advertised price as long as the competition has the goods on hand; not responsible for typographical errors; prices & specifications subject to change without notice; this ad supercedes all previous ads; min shpg & hndlg $5.95; Send $2 for 112-pg catalog

HEWLETT PACKARD



HP-87 $1885.00
32K Bytes standard

82907A	32K Mem. mod.	239.00
82908A	64K Mem. mod.	337.00
82909A	128K Mem. mod.	537.00
HP-85A	Computer	$1985.00
HP-125	Computer	1985.00
HP-41C	Prog. scientific	188.00
HP-41CV	Prog. scientific	248.00
HP-41	Card reader	169.00
HP-41	Printer	297.00
HP-41	Optical wand	99.00
82106A	41C Memory module	26.00
82170A	41C Quad mem. mod.	84.00
82160A	HP-IL interface	109.00
82161A	Cassette drive	437.00
82162A	HP-IL printer	397.00
HP-97	Prog. scien/printer	584.00
HP-67	Hand-held fully prog	309.95
HP-32E	Scientific	58.95
HP-33C	Prog. scientific	93.95
HP-34C	Adv. prog. scientific	114.00
HP-37E	Business	77.95
HP-38C	Prog. financial	119.00
HP-11C	Prog. scientific	114.00
HP-12C	Prog. financial	127.00
82905A	Impact printer	619.00
82901M	5¼" dual drive	1699.00
9895A	8" dual drive	4499.00
2631B	Impact printer	2999.00
2601A	Letter qual. ptr.	2999.00
85-15001	Mass storage ROM	124.00
85-15002	Plot/print ROM	124.00
85-15003	I/O ROM	239.00
85-15004	Matrix ROM	124.00
85-15005	Adv. prog. ROM	124.00
85-15007	Assembler ROM	239.00
82903A	16K mem. module	169.00
82937A	HP-IB interface	328.00
82939A	Serial interface	338.00
82940A	GPIO interface	408.00
82941A	BCD interface	408.00
82949A	Parallel ptr. int.	239.00
HP-7470A	Hi-speed color grphc pltr w/intrfc	1269.00

New TI-99/4A - 48K RAM
100% TI parts w/Extended Basic
or TI-LOGO *Re: $1300 Y/C:* $$689.95

Texas Instruments Home Computer

16K

TI-99/4A *Only* $299⁹⁵

	RETAIL	Your Cost
10" color monitor high res	374.95	339.95
32K memory module	399.95	314.95
Extended Basic	100.00	75.00
Speech synthesizer	149.95	129.95
Disk memory drive	499.95	394.95
Telephone coupler (modem)	224.95	189.95
Printer (solid state)	399.95	319.95
TI-LOGO	250.00	99.95

SANYO MONITORS

High resolution, number one seller!

		Retail	*Your Cost*
VM4509	9" B & W	190.00	159.95
VM4215	15" B & W (below our cost)	349.00	189.95
DM5109	9" Green *VM4215 limited offer*	200.00	169.95
DM8012	12" B & W	250.00	199.95
DM8112	12" Green	260.00	209.95
DMC6013	13" Color, hi quality	470.00	399.95
DMC6113	13" Color RGB hi res	995.00	799.95
DM2012	(NEW) 12" B & W	179.00	139.95
DM2112	(NEW) 12" Green	199.00	159.95

Free w/purchase of each Sanyo monitor one $49.00 LCD pen watch
(subject to avail. on new models & others)

commodore VIC-20

5K Personal Computer
(Expands to 32K) Works with any TV!
Retail: $300.00 Your SPECIAL Cost: $259.95
FREE with purch of VIC-20 one $49.50 LCD Pen Watch!
Commodore Datasette Recorder
for VIC-20 Your Cost: $68.95 We carry all peripherals, software & access. for the VIC-20 (we will beat any price!)

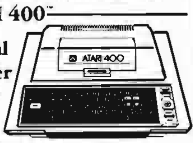
ATARI® IN STOCK COMPUTERS

FOR ATARI 400 & 800–
GHOST HUNTER (similar to Pac-Man)
Oisk $34.95 Cassette $29.95

Model		Retail	Your Cost
800	16K	1080.00	689.95
800	48K	1280.00	789.95
800	160K		
(800-32K + 128 RAM disk)			1299.95
810	Oisk drive	600.00	469.95
Atari	VisiCalc	200.00	169.95
825	Impact Ptr	1000.00	779.95

ATARI 400™

Personal Computer System



SUPER SPECIAL – ATARI 400 (16K)
Retail $595.00 Your Cost: $299.95
400 Lang. card opt. $49.95

ATARI *Retail* $225.00
Programmable Color TV Game
Your Cost: $139⁹⁵

Atari's *game cartridge*
Pac-man $27.95

I O S Prism Printers–
*& Software for Apple Computers**
**call for our LOW, LOW prices!*

Immediate delivery–	*Retail*	*Your Cost*
132 column (color)	1995.00	1795.00
80 column (B & W)	1196.00	999.00

While supply lasts
CLOSE-OUT *ON PRINTERS*

PAPER TIGER by I D S			
Model	*Retail*	*Our Cost*	*We Sell*
460 G	1094.00	733.00	699.95
445 G	894.00	599.00	579.95
445	795.00	556.00	536.00
CENTRONICS			
739-1	895.00	650.00	550.00
737	895.00	610.00	499.00

Epson PRINTERS

	Your Cost
MX70	$299.95
MX80 with graftrax	499.95
MX80 FT " "	599.00
MX100	749.00
Graphtrax chip	95.00

MATTEL ELECTRONICS INTELLIVISION
$229⁹⁵
Retail: $325.00
animated TV game!

AMDEK Leedex High Quality Monitors

Model	Description	Retail	Your Cost
100	12" B/W, 12 MHz	180.00	149.00
300 G	12" Green, 18 MHz	269.00	199.00
Color I	13" Color, NTSC comp. input,audio amp & spkr	499.00	359.00
Color II	13" Color,RGB input,hi res graphics, spkr	1295.00	799.00
Color III	13" Color, RG8 240 line resolution	699.00	459.00

Apple Disk Sector Map Utility

by Clyde Camp

This machine-language program produces a display that shows which physical sectors on a disk are actually used by the Apple's lo-res graphics capability.

DISKMAP
requires:
Apple II or Apple II Plus
with DOS 3.3

It is rarely necessary to know where on a disk data and programs are physically located. In fact, keeping track of all of the allocation details is one of the major reasons for having a Disk Operating System (DOS) in the first place. Most users don't want or need to know actual sector allocation because it obstructs more important functions such as program development and debugging.

There are times, however, when that knowledge is useful. For instance, you may need this information when a disk has been in use for a while and the programs are scattered in random sectors rather than in contiguous sectors. This happens as programs are deleted or saved and the disk management routines split up programs according to their own internal sector allocation algorithms. This program will display graphically just how broken up the sector allocations really are. In severe cases, program load/save time and data access time in random files can deteriorate significantly. At this point you're better off to stop and transfer your programs to a new disk using FID or a simple LOAD/SAVE. This will concatenate your programs onto contiguous sectors and decrease your access times accordingly.

Configuration

The program was written for a 48K Apple II Plus or Apple II with the Applesoft ROM card, but should work on any system with 48K using DOS 3.3. If you have an assembler (I used the S-C Assembler Version 4.0), the program is easily relocated by altering the .TA and .OR pseudo-ops. I strongly recommend you acquire an assembler (if you do not have one) if you need to relocate this program. The program is far too complex to relocate manually.

Installation

Assuming you have one of the above system configurations, the program should be installed as follows:

1. Boot up your system to get DOS installed.

2. Enter the MONITOR (CALL − 151) and key in the hex data per listing 1. *Do not* begin at address $9900 as the listing shows; this could zap DOS. Instead, begin at location $900.

3. After keying in the data, carefully check it using the MONITOR hex or symbolic dump routines (see Chapter 3 in the *Apple II Reference Manual).*

4. Save the program on disk by:
   ```
   BSAVE DISKMAP,A$900,L$320
   ```

5. Convert the disk to a master disk using the DOS MASTER CREATE utility. The program will not operate with a slave diskette.

6. Return to BASIC by typing FP (or INT) and initialize a scratch disk using a null (non-existent) HELLO program. Save the DISKMAP program on the scratch disk as well, using the procedure in step 4. Use this scratch disk in all the following steps; if you have made a mistake you will not zap a good disk.

7. Run the program by:
   ```
   BRUN DISKMAP,A$9900
   ```

8. If all goes well, you should get the blinking BASIC cursor back after a moment. If you do not get the cursor and/or strange things start happening to the screen or disk, turn the system *off,* insert the DOS 3.3 master disk, and reboot the system from scratch. Then BLOAD DISKMAP, re-enter the MONITOR, and look for the erroneous data entry.

9. Assuming that all goes well and Applesoft's "}" or Integer's ">" prompt reappears, proceed as follows.

10. With the scratch disk still in the drive, type "&" followed by a carriage return. At this point you should get a split lo-res screen full of garbage with text on the four lines as shown in figure 1.

11. Now try out the modes of operation described in the Operation section of this article. After "C", you should get figure 1. After a "1", you should get figure 2. An "E" command should exit the program with the disk map intact and with the BASIC cursor at the bottom of the screen. Now "DELETE HELLO" and "SAVE HELLO".

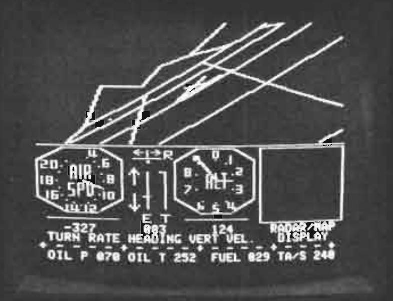
12. Re-enter the utility *via* "&" and type "2". The result should be figure 3.

    If everything is still operating correctly, quit the program by typing in "Q". The screen should clear and you should re-enter Applesoft or Integer BASIC.

13. Save the program on the scratch disk by:
    ```
    BSAVE DISKMAP,A$9900,L$320
    ```

14. Catalog the disk to make sure all is OK, then save the program on your permanent disk in a similar way.

from SubLOGIC... quality software for your Apple II.



AIR SPD
ALT
E T
-327
003
124
TURN RATE HEADING VERT VEL.
RADAR/MAP DISPLAY
OIL P 070 OIL T 252 FUEL 029 TA/S 240

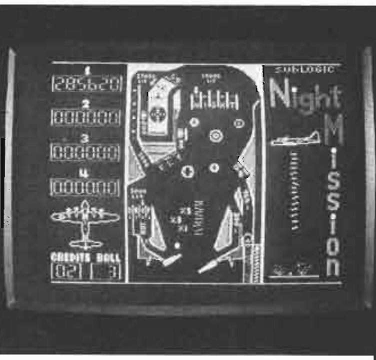
A2-FS1

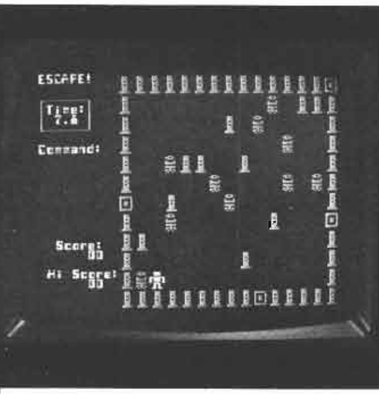
FLIGHT SIMULATOR

Combines superior flight simulation with the best animated 3D graphics available. Practice take-offs and landing, other aerial maneuvers, declare war on the enemy. 16K cassette, **$25.00**. 32K disk, **$33.50.**

A2-PB1

PINBALL – The ultimate arcade simulation program, an exciting pinball game with the ball and flipper precision to make increased skill pay off. Includes 10 different play modes and 100 user-adjustable modes. 48K disk, **$29.95.**





A2-SG1

ESCAPE! – The simple game of captive pursuit. A gem for game purists . . . A classic for your game collection. 48K disk, **$29.95.**

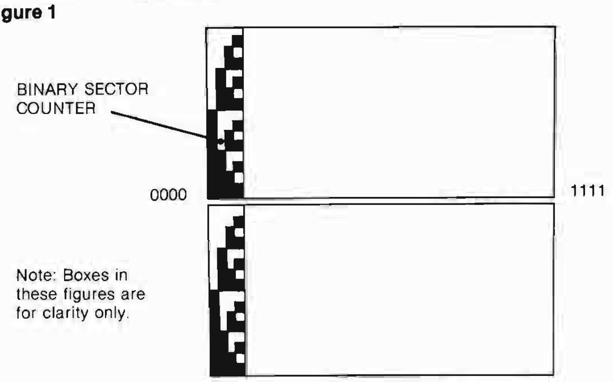
See them today at your dealer . . .

or for direct orders add $1.50 and specify UPS or first class mail. Illinois residents add 5% sales tax. VISA and MasterCard accepted. Descriptive brochures of most products listed here are available on request

"Apple" is the registered trademark of Apple Computer Inc.

subLOGIC
Communications Corp.
713 Edgebrook Drive
Champaign, IL 61820
(217) 359-8482
Telex: 206995

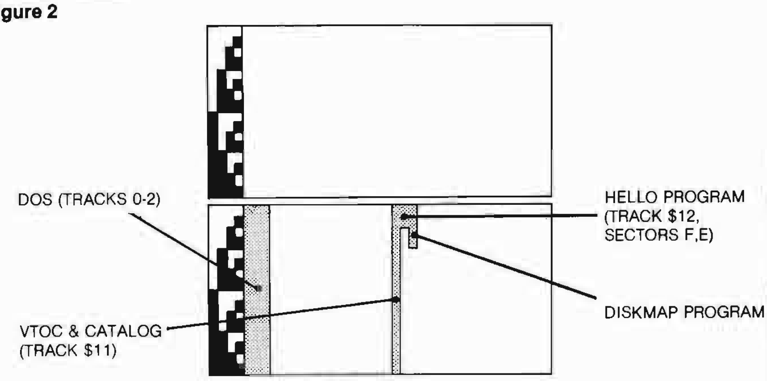
Figure 1



Note: Boxes in these figures are for clarity only.

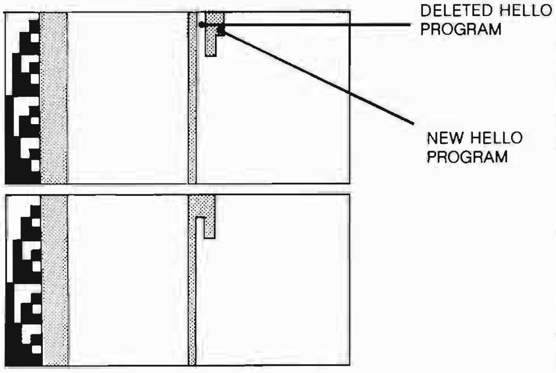
```
HEX 00000000000000001111111111111111222
TRK 0123456789ABCDEF0123456789ABCDEF012
BLACK = USED     SHADED = FREE
          MODE = 1,2,C,E,Q ? C
```

Figure 2



```
HEX 00000000000000001111111111111111222
TRK 0123456789ABCDEF0123456789ABCDEF012
BLACK = USED     SHADED = FREE
          MODE = 1,2,C,E,Q ? 1
```

Figure 3



```
HEX 00000000000000001111111111111111222
TRK 0123456789ABCDEF0123456789ABCDEF012
BLACK = USED     SHADED = FREE
          MODE = 1,2,C,E,Q ? 2
```

Operation

Once the program is debugged per the previous procedure (designed to minimize trauma caused by clobbered disks), operation is very easy.

After the program is initialized by BRUN DISKMAP, it is entered by use of the Applesoft "&" command, either from inside a program or from the keyboard. Once this is done the following commands are recognized by the DISKMAP program. Refer to figure 2 for clarification.

1 - Read the disk and place its map into the MAP #1 position.

2 - Read the disk and place its map into the MAP #2 position.

C -Clear and initialize the graphics screen.

E - End the program and return to the calling program without clearing the graphics display.

Q -Quit the program and return to the calling program full text mode.

The program, hidden between DOS and its buffers, is unaffected by FP,INT,NEW,CLEAR,HIMEN,LOMEM, or any other user or program action short of rebooting the system *via* PR#6 or its equivalent.

How the Program Works

The program is broken into two major sections. The first is a setup routine which is executed once when the program is BRUN the first time. This routine (ENTRY) performs three functions:

1. "POKEs" in a fake HIMEM (lines 1860-1890)

2. "POKEs" in the "&" JUMP vector (lines 1900-1950)

3. Moves the DOS sector buffers so as to hide the program between the buffers and DOS itself (lines 1960-2000)

After performing these tasks, the program exits back into whatever mode the Apple was in when DISKMAP was BRUN.

Lines 2060-2230 define the I/O Buffer (IOB) and Device Characteristics Table (DCT) required by the RWTS routine which is accessed in the program by GETVTC in lines 2240-2270.

The second part of the program is the section that is executed when the "&" Applesoft command in encountered. It is composed of a primary control loop routine (START) and three major subroutines (CODE, MAP, and NEWTRK). The operation of each of these will be discussed in the following paragraphs.

All program references refer to listing 1 (a working knowledge of 6502 machine language is assumed). This particular assembler utilizes '/SYM' to denote the most significant byte of the 16-bit word 'SYM' and '#SYM' for the least significant byte.

Main Control Loop (START)

This routine begins by stuffing the value $3CF onto the top of the stack. This step will be interpreted by the RTS in line 2730 as a return to location $3D0, which is the DOS warm start vector.

Lines 2520-2600 then set the screen to the mixed lo-res mode, set normal video, and position the cursor at text screen line 20 in preparation for the text message.

The main control loop begins at START2 by clearing the text screen and printing the text message. Then it waits for a keystroke input using the BASIC KEYIN routine. After line 2640 saves the key pressed as part of the text message, lines 2650-2730 test for and execute the 'E' and 'Q' commands which terminate the Disk Mapper and return to the calling program *via* the RTS at line 2730.

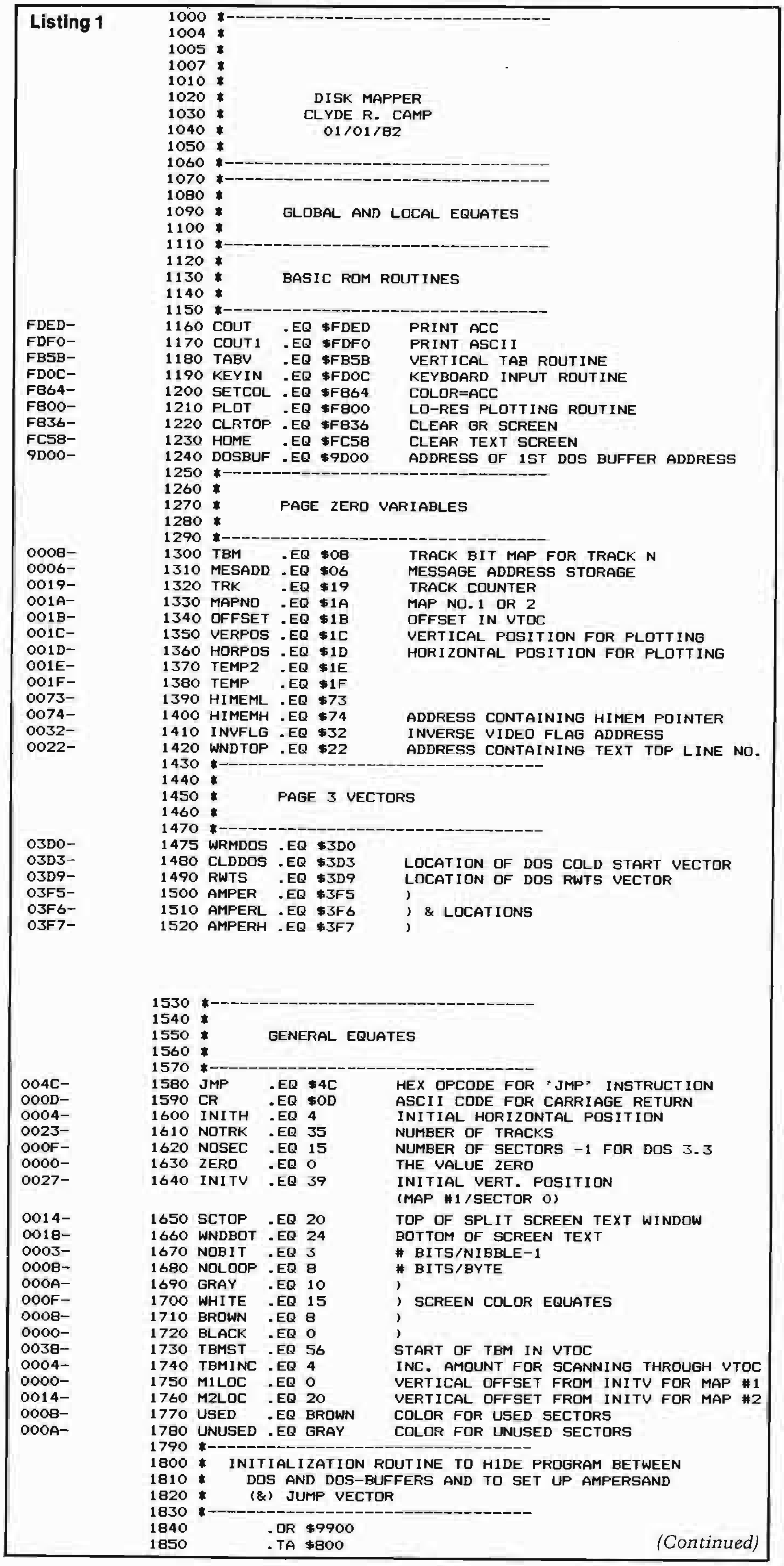
Lines 2740-2780 test for and execute the 'C' command by clearing the graphics screen and re-plotting the black and white binary code sector reference for the two maps.

Lines 2790-2890 test for the '1' or '2' commands and store the appropriate vertical offset into MAPNO for use by the plotting routines. Invalid command keys are detected by line 2860.

Lines 2900-2920 then call subroutines to get the VTOC from the disk, plot its configuration in the appropriate map location, and loop back for the next command.

Binary Code Plotter (CODE)

This subroutine is called by the 'C' command after the graphics screen is cleared by line 2760 and plots a binary sector counter on the left side of the



Listing 1

```
       1000 *-----------------------------------
       1004 *
       1005 *
       1007 *
       1010 *
       1020 *         DISK MAPPER
       1030 *        CLYDE R. CAMP
       1040 *          01/01/82
       1050 *
       1060 *-----------------------------------
       1070 *-----------------------------------
       1080 *
       1090 *       GLOBAL AND LOCAL EQUATES
       1100 *
       1110 *-----------------------------------
       1120 *
       1130 *       BASIC ROM ROUTINES
       1140 *
       1150 *-----------------------------------
FDED-  1160 COUTT   .EQ $FDED     PRINT ACC
FDF0-  1170 COUT1   .EQ $FDF0     PRINT ASCII
FB5B-  1180 TABV    .EQ $FB5B     VERTICAL TAB ROUTINE
FD0C-  1190 KEYIN   .EQ $FD0C     KEYBOARD INPUT ROUTINE
F864-  1200 SETCOL  .EQ $F864     COLOR=ACC
F800-  1210 PLOT    .EQ $F800     LO-RES PLOTTING ROUTINE
F836-  1220 CLRTOP  .EQ $F836     CLEAR GR SCREEN
FC58-  1230 HOME    .EQ $FC58     CLEAR TEXT SCREEN
9D00-  1240 DOSBUF  .EQ $9D00     ADDRESS OF 1ST DOS BUFFER ADDRESS
       1250 *-----------------------------------
       1260 *
       1270 *       PAGE ZERO VARIABLES
       1280 *
       1290 *-----------------------------------
0008-  1300 TBM     .EQ $08       TRACK BIT MAP FOR TRACK N
0006-  1310 MESADD  .EQ $06       MESSAGE ADDRESS STORAGE
0019-  1320 TRK     .EQ $19       TRACK COUNTER
001A-  1330 MAPNO   .EQ $1A       MAP NO.1 OR 2
001B-  1340 OFFSET  .EQ $1B       OFFSET IN VTOC
001C-  1350 VERPOS  .EQ $1C       VERTICAL POSITION FOR PLOTTING
001D-  1360 HORPOS  .EQ $1D       HORIZONTAL POSITION FOR PLOTTING
001E-  1370 TEMP2   .EQ $1E
001F-  1380 TEMP    .EQ $1F
0073-  1390 HIMEML  .EQ $73
0074-  1400 HIMEMH  .EQ $74       ADDRESS CONTAINING HIMEM POINTER
0032-  1410 INVFLG  .EQ $32       INVERSE VIDEO FLAG ADDRESS
0022-  1420 WNDTOP  .EQ $22       ADDRESS CONTAINING TEXT TOP LINE NO.
       1430 *-----------------------------------
       1440 *
       1450 *       PAGE 3 VECTORS
       1460 *
       1470 *-----------------------------------
03D0-  1475 WRMDOS  .EQ $3D0
03D3-  1480 CLDDOS  .EQ $3D3      LOCATION OF DOS COLD START VECTOR
03D9-  1490 RWTS    .EQ $3D9      LOCATION OF DOS RWTS VECTOR
03F5-  1500 AMPER   .EQ $3F5      )
03F6-  1510 AMPERL  .EQ $3F6      ) & LOCATIONS
03F7-  1520 AMPERH  .EQ $3F7      )


       1530 *-----------------------------------
       1540 *
       1550 *       GENERAL EQUATES
       1560 *
       1570 *-----------------------------------
004C-  1580 JMP     .EQ $4C       HEX OPCODE FOR 'JMP' INSTRUCTION
000D-  1590 CR      .EQ $0D       ASCII CODE FOR CARRIAGE RETURN
0004-  1600 INITH   .EQ 4         INITIAL HORIZONTAL POSITION
0023-  1610 NOTRK   .EQ 35        NUMBER OF TRACKS
000F-  1620 NOSEC   .EQ 15        NUMBER OF SECTORS -1 FOR DOS 3.3
0000-  1630 ZERO    .EQ 0         THE VALUE ZERO
0027-  1640 INITV   .EQ 39        INITIAL VERT. POSITION
                                  (MAP #1/SECTOR 0)
0014-  1650 SCTOP   .EQ 20        TOP OF SPLIT SCREEN TEXT WINDOW
0018-  1660 WNDBOT  .EQ 24        BOTTOM OF SCREEN TEXT
0003-  1670 NOBIT   .EQ 3         # BITS/NIBBLE-1
0008-  1680 NOLOOP  .EQ 8         # BITS/BYTE
000A-  1690 GRAY    .EQ 10        )
000F-  1700 WHITE   .EQ 15        ) SCREEN COLOR EQUATES
0008-  1710 BROWN   .EQ 8         )
0000-  1720 BLACK   .EQ 0         )
0038-  1730 TBMST   .EQ 56        START OF TBM IN VTOC
0004-  1740 TBMINC  .EQ 4         INC. AMOUNT FOR SCANNING THROUGH VTOC
0000-  1750 M1LOC   .EQ 0         VERTICAL OFFSET FROM INITV FOR MAP #1
0014-  1760 M2LOC   .EQ 20        VERTICAL OFFSET FROM INITV FOR MAP #2
0008-  1770 USED    .EQ BROWN     COLOR FOR USED SECTORS
000A-  1780 UNUSED  .EQ GRAY      COLOR FOR UNUSED SECTORS
       1790 *-----------------------------------
       1800 *  INITIALIZATION ROUTINE TO HIDE PROGRAM BETWEEN
       1810 *    DOS AND DOS-BUFFERS AND TO SET UP AMPERSAND
       1820 *     (&) JUMP VECTOR
       1830 *-----------------------------------
       1840         .OR $9900
       1850         .TA $800
```

(Continued)

Listing 1 *(Continued)*

```
9900- A9 00    1860 ENTRY  LDA #ENTRY   )
9902- 85 73    1870        STA HIMEML   ) POINT HIMEM TO ENTRY POINT
9904- A9 99    1880        LDA /ENTRY   )
9906- 85 74    1890        STA HIMEMH   )
9908- A9 DA    1900        LDA #START   >
990A- 8D F6 03 1910        STA AMPERL   > SET UP AMPERSAND JUMP TO 'START'
990D- A9 9A    1920        LDA /START   >
990F- 8D F7 03 1930        STA AMPERH   >
9912- A9 4C    1940        LDA #JMP     >
9914- 8D F5 03 1950        STA AMPER    >
9917- A9 00    1960        LDA #ENTRY   )
9919- 8D 00 9D 1970        STA DOSBUF   ) MOVE DOS BUFFERS TO PROTECT PROGRAM
991C- A9 98    1980        LDA /ENTRY-1 )
991E- 8D 01 9D 1990        STA DOSBUF+1 )
9921- 4C D3 03 2000        JMP CLDDOS   ) COLD START DOS
               2010 *---------------------------------
               2020 *
               2030 *      DOS BUFFER,IOB,DCB, AND CALLING ROUTINE
               2040 *
               2050 *---------------------------------
9924-          2060 BUF    .BS 256      SECTOR STORAGE BUFFER
9A24- 01       2070 IOB    .DA #$01     IOB TYPE
9A25- 60       2080 IBSLOT .DA #$60     DISK SLOT
9A26- 01       2090 IBDRVN .DA #$01     DISK DRIVE
9A27- 00       2100        .DA #$00     VOLUME
9A28- 11       2110        .DA #$11     TRACK NUMBER
9A29- 00       2120        .DA #$00     SECTOR NUMBER
9A2A- 35 9A    2130        .DA DCT      DCT ADDRESS
9A2C- 24 99    2140        .DA BUF      BUFFER ADDRESS
9A2E- 00 00    2150        .DA $00      NOT USED
9A30- 01       2160        .DA #$01     READ CODE
9A31- 00       2170        .DA #$00     ERROR CODE
9A32- FE       2180        .DA #$FE     VOLUME NUMBER
9A33- 60       2190 IOBPSN .DA #$60     SLOT NUMBER
9A34- 01       2200 IOBPDN .DA #$01     DRIVE NUMBER
9A35- 00       2210 DCT    .DA #$00     DEVICE TYPE CODE
9A36- 01       2220        .DA #$01     PHASES/TRACK
9A37- EF D8    2230        .DA $D8EF    TIME COUNT
9A39- A9 9A    2240 GETVTC LDA /IOB     )
9A3B- A0 24    2250        LDY #IOB     ) ROUTINE TO READ TRK/SEC
                                          DEFINED BY IOB
9A3D- 20 D9 03 2260        JSR RWTS     )
9A40- 60       2270        RTS
               2280 *---------------------------------
               2290 *
               2300 *      THE FOLLOWING IS THE
               2310 *      TEXT MESSAGE FOR THE PROGRAM
               2320 *
               2330 *      ".AS -" SETS THE HIGH ORDER BIT IN THE
               2340 *          ASCII BYTE
               2350 *
               2360 *---------------------------------
9A41- C8 C5 D8
9A44- A0 B0 B0
9A47- B0 B0 B0
9A4A- B0 B0 B0
9A4D- B0 B0 B0
9A50- B0 B0 B0
9A53- B0 B0 B1
9A56- B1 B1 B1
9A59- B1 B1 B1
9A5C- B1 B1 B1
9A5F- B1 B1 B1
9A62- B1 B1 B1
9A65- B2 B2 B2 2370 M1     .AS -"HEX 00000000000000001111111111111111222"
9A68- A0 D4 D2
9A6B- CB A0 B0
9A6E- B1 B2 B3
9A71- B4 B5 B6
9A74- B7 B8 B9
9A77- C1 C2 C3
9A7A- C4 C5 C6
9A7D- B0 B1 B2
9A80- B3 B4 B5
9A83- B6 B7 B8
9A86- B9 C1 C2
9A89- C3 C4 C5
9A8C- C6 B0 B1
9A8F- B2       2380        .AS -" TRK 0123456789ABCDEF0123456789ABCDEF012"
9A90- A0 C4 C1
9A93- D2 CB A0
9A96- A8 C2 D2
9A99- CF D7 CE
9A9C- A9 A0 BD
9A9F- A0 D5 D3
9AA2- C5 C4 A0
9AA5- A0 CC C9
9AA8- C7 C8 D4
9AAB- A0 A8 C7
9AAE- D2 C1 D9
9AB1- A9 A0 BD
9AB4- A0 C6 D2
9AB7- C5 C5    2390        .AS -" DARK (BROWN) = USED  LIGHT (GRAY) = FREE"
```

(Continued)

screen in two locations (one for each disk map). It uses black for binary '0' and white for binary '1'. This is necessary as the lo-res graphics screen cannot have text imbedded in the graphics portion. The graphical representation of a 4-bit counter is useful for locating a specific sector within the map.

The 'CODE' routine consists of two nested loops which are executed twice with different values of MAPNO (once for each map position). The outer loop is executed 16 times as it counts from 15 down to zero, representing the 16 sectors within a track. The inner loop is executed four times for each sector count as it plots black or white for each bit in the nibble representing the sector count.

Mapping Routine (MAP)

This routine, the heart of the program, reads each word in the Track Bit Map portion of the VTOC saved at BUF. It then plots each word as a vertical track using the NEWTRK subroutine.

Lines 3390-3460 initialize the parameters necessary to plot the appropriate map. The TBM consists of 140 bytes beginning at byte TBMST of the VTOC. These are arranged in blocks of four bytes per track with

BYTE #1 = status of tracks 15-8
(1 = free 0 = used)

BYTE #2 = status of tracks 7-0
(1 = free 0 = used)

BYTE #3 = unused (all zero)

BYTE #4 = unused (all zero)

Lines 3470-3520 get the first two bytes for the appropriate track and save them in variable locations TBM and TBM + 1 for use by the NEWTRK routine which is then called by line 3530.

After NEWTRK plots the status of the given track, lines 3540-3640 calculate the next track, re-initialize the necessary parameters, and loop if there are more sectors to do. Otherwise, line 3640 returns to the START routine for the next command.

Track Plotting Routine (NEWTRK)

This routine plots the status of a single track based on the data passed to TBM and TBM + 1. It consists of two pieces of inline code, identical except for whether or not they access TBM or TBM + 1 as the data source. Lines 3740-3890 are for sectors 0-7 (TBM + 1) and lines 3900-4050 are for sectors 8-15 (TBM).

Line 3740 sets up the 'X' register as the loop counter for 8 bits in the byte. Lines 3750-3760 set the plotting color to the 'UNUSED' color. This is done by use of the BASIC SETCOL routine which expects to see the desired color in the 'A' register.

Next, the 'Y' register is loaded with the horizontal position for this track (HORPOS) and lines 3780-3800 test the next bit in TBM + 1 to determine its value. If the bit is a '1' (unused) then the routine continues, otherwise lines 3810-3820 switch the color to 'USED' prior to continuing.

Lines 3830-3850 then calculate the desired vertical position based on the map (1 or 2) and the sector. Line 3860 then calls the BASIC PLOT routine which plots one lo-res point with the color last set by SETCOL at the horizontal location in 'Y', and the vertical position in 'A'. This routine destroys the 'A' data but does not disturb 'X' or 'Y'.

Lines 3870-3890 alter the vertical position to the next sector and test to see if all bits in the byte (TBM + 1) have been plotted. If so, line 3890 falls through to an identical routine for TBM.

Program Modifications

Of the several modifications that could be made to the program, the most likely would be to access another disk drive or slot. This can be done by changing the disk slot number at IBSLOT and IBDRVN. The values for IBDRVN are $01 or $02 and the values for IBSLOT are $n0 where n is the slot number that the disk interface card is plugged into. The locations IOBPSN and IOBPDN should also be changed to match IBSLOT and IBDRVN respectively. All of these locations are in the IOB (lines 2080-2200).

The program will also work with DOS 3.2, provided that NOSEC (lines 1620 and 3020) is changed from 15 ($0F) to 12 ($0C) and NOLOOP (line 3740 *only*) is changed from 8 to 5.

The colors chosen for 'USED' and 'UNUSED' were selected to be visually acceptable on both color and black-white monitors. Any other colors may be used, depending on personal preference, by changing lines 1770, 1780, 3750, 3810, 3910, and 3970. A word of caution: What looks good in color does not necessarily look good in black and white due to the Apple's hardware.

Listing 1 *(Continued)*

```
9AB9- A0 A0 A0
9ABC- A0 A0 A0
9ABF- A0 A0 A0
9AC2- A0 A0 A0
9AC5- CD CF C4
9AC8- C5 A0 BD
9ACB- A0 B1 AC
9ACE- B2 AC C3
9AD1- AC C5 AC
9AD4- D1 A0 BF
9AD7- A0       2400         .AS -"            MODE = 1,2,C,E,Q ? "
9AD8- A0       2410 LSTKEY  .DA #$A0     LAST KEY COMMAND (PART OF TEXT MESSAGE)
9AD9- 0D       2420         .DA #13      HEX C.R. (END OF TEXT MESSAGE)
               2430 *-----------------------------------
               2440 *
               2450 *       ENTRY POINT OF PROGRAM FOLLOWING "&"
               2460 *
               2470 *-----------------------------------
9ADA- A9 03    2480 START   LDA /WRMDOS-1
9ADC- 48       2490         PHA            ) FAKE A RETURN TO DOS WARM START
9ADD- A9 CF    2500         LDA #WRMDOS-1     BY PUSHING $03CF ON TOP OF STACK
9ADF- 48       2510         PHA            )
9AE0- AE 53 C0 2520         LDX $C053    SET MIXED TEXT/GRAPHICS
9AE3- AE 50 C0 2530         LDX $C050    SET GRAPHICS MODE
9AE6- AE 56 C0 2540         LDX $C056    SET LO RESOLUTION
9AE9- A2 FF    2550         LDX #$FF
9AEB- 86 32    2560         STX INVFLG   SET NORMAL VIDEO
9AED- A2 14    2570         LDX #SCTOP
9AEF- 86 22    2580         STX WNDTOP   SET TEXT WINDOW TOP TO LINE 20
9AF1- A9 18    2590         LDA #WNDBOT
9AF3- 20 5B FB 2600         JSR TABV     VTAB TO LINE 23
9AF6- 20 58 FC 2610 START2  JSR HOME     CLEAR TEXT
9AF9- 20 10 9C 2620         JSR MM1      PRINT MESSAGE
9AFC- 20 0C FD 2630         JSR KEYIN    GET A COMMAND
9AFF- 8D D8 9A 2640         STA LSTKEY   SAVE COMMAND IN MESSAGE
9B02- C9 C5    2650         CMP #$C5     WAS IT 'E'?
9B04- F0 0B    2660         BEQ END      IF 'E' THEN END WITHOUT CLEARING SCREEN
9B06- C9 D1    2670         CMP #$D1     WAS COMMAND A 'Q' ?
9B08- D0 0B    2680         BNE NOTQ     IF NOT KEEP LOOKING
9B0A- AE 51 C0 2690         LDX $C051    IF SO SET TEXT MODE
9B0D- A2 00    2700         LDX #M1LOC   GET VERTICAL OFFSET FOR MAP #1
9B0F- 86 22    2710         STX WNDTOP
9B11- 20 58 FC 2720 END     JSR HOME     CLEAR SCREEN TEXT
9B14- 60       2730         RTS          RETURN TO CALLING PROG. VIA FAKE RETURN
9B15- C9 C3    2740 NOTQ    CMP #$C3     WAS COMMAND A 'C'
9B17- D0 09    2750         BNE NOTC     IF NOT KEEP LOOKING
9B19- 20 36 F8 2760         JSR CLRTOP   OTHERWISE CLEAR GRAPHICS SCREEN
9B1C- 20 44 9B 2770         JSR CODE     DISPLAY BINARY CODE FOR SECTOR REFERENCE
9B1F- 4C F6 9A 2780         JMP START2   AND LOOP TO NEW COMMAND
9B22- C9 B1    2790 NOTC    CMP #$B1     WAS IT A '1' (FOR MAP #1)
9B24- D0 05    2800         BNE NOT1     IF NOT KEEP LOOKING
9B26- A2 00    2810         LDX #M1LOC   VERTICAL OFFSET FOR MAP #1
9B28- 4C 36 9B 2820         JMP VTOC2    AND GO GET THE TRACK MAP
9B2B- A2 A0    2830 NOT1    LDX #$A0     PUT A ASCII BLANK IN X
9B2D- C9 B2    2840         CMP #$B2     WAS IT A '2'
9B2F- 8E D8 9A 2850         STX LSTKEY   STORE A BLANK FOR LAST KEY
9B32- D0 C2    2860         BNE START2   IF NOT THEN BAD COMMAND SO LOOP
9B34- A2 14    2870         LDX #M2LOC   GET VERTICAL OFFSET FOR MAP #2
9B36- 86 1A    2880 VTOC2   STX MAPNO    SAVE VERTICAL OFFSET AS MAP NUMBER
9B38- 8D D8 9A 2890         STA LSTKEY   STORE VALID KEY AS PART OF MESSAGE
9B3B- 20 39 9A 2900 VTOC    JSR GETVTC   GET VOLUME TABLE OF CONTENTS FROM DISK
9B3E- 20 80 9B 2910         JSR MAP      GO PLOT THE MAP BASED ON VTOC DATA
9B41- 4C F6 9A 2920         JMP START2   AND LOOP FOR NEW COMMAND
               2930 *-----------------------------------
               2940 *
               2950 *       ROUTINE TO PUT BINARY COUNTER ON LEFT
               2960 *         SIDE OF SCREEN FOR MAP SECTOR REFERENCE
               2970 *
               2980 *-----------------------------------
               2990 CODE
9B44- A2 00    3000         LDX #M1LOC   SET UP OFFSET FOR MAP #1
9B46- 86 1A    3010         STX MAPNO    AND SET MAP #
9B48- A2 0F    3020 CODE2   LDX #NOSEC   16 SECTORS FOR DOS 3.3 (13 FOR DOS 3.2)
9B4A- 86 1F    3030 NXTNIB  STX TEMP     STORE SECTOR COUNT IN TEMP
9B4C- A0 03    3040         LDY #NOBIT   COUNTER FOR BITS IN HEX NIBBLE (0-F)
9B4E- 86 1E    3050         STX TEMP2    SAVE AS BIT COUNTER
9B50- A9 00    3060 NXTBIT  LDA #BLACK   SETUP FOR BIT=0
9B52- 20 64 F8 3070         JSR SETCOL   SET COLOR TO BLACK
9B55- 18       3080         CLC          GET RID OF CARRY IF ANY
9B56- 66 1F    3090         ROR TEMP     ROTATE TO NEXT BIN IN COUNTER
9B58- 90 05    3100         BCC BLK      JUMP IF BIT IS ZERO
9B5A- A9 0F    3110         LDA #WHITE   IF NOT A ZERO IT MUST BE ONE SO...
9B5C- 20 64 F8 3120         JSR SETCOL   SET COLOR TO WHITE
9B5F- A9 27    3130 BLK     LDA #INITV   A = BOTTOM OF GRAPHICS SCREEN
9B61- 18       3140         CLC          CLEAR CARRY
9B62- E5 1E    3150         SBC TEMP2    SUBTRACT THE SECTOR COUNT
9B64- E5 1A    3160         SBC MAPNO    SUBTRACT OFF MAP # BIAS
9B66- 20 00 F8 3170         JSR PLOT     PLOT THE BIT (1=WHITE 0=BLACK)
9B69- 88       3180         DEY          DECREMENT BIT COUNTER FOR NIBBLE
9B6A- 10 E4    3190         BPL NXTBIT   LOOP  TO NEXT BIT OF NIBBLE
9B6C- CA       3200         DEX          DECREMENT THE SECTOR COUNTER
9B6D- 10 DB    3210         BPL NXTNIB   LOOP FOUR TIMES THROUGH NIBBLE (HEX 'F')
9B6F- A9 14    3220         LDA #M2LOC   VERTICAL BIAS FOR MAP #2
```

(Continued)

Listing 1 *(Continued)*

```
9B71- C5 1A    3230        CMP MAPNO     DID I JUST DO MAP #2?
9B73- F0 05    3240        BEQ GOON2     IF SECOND TIME THROUGH THEN QUIT
9B75- 85 1A    3250        STA MAPNO     ELSE PUT IT IN MAPNO
9B77- 4C 48 9B 3260        JMP CODE2     AND GO DO IT AGAIN FOR MAP #2
9B7A- A9 00    3270 GOON2  LDA #M1LOC    VERTICAL BIAS FOR MAP #1
9B7C- 85 1A    3280        STA MAPNO     SAVE IN MAPNO
9B7E- 60       3290        RTS           RETURN
9B7F- 60       3300        RTS           NOT USED (SPARE BYTE)
               3310 *--------------------------------
               3320 *
               3330 *      ROUTINE TO PLOT A 16 SECTOR BY 35 TRACK
               3340 *        MAP AT ONE OF TWO VERTICAL LOCATIONS
               3350 *        DEFINED BY (INITV-MAPNO)
               3360 *
               3370 *--------------------------------
               3380 MAP
9B80- A9 00    3390        LDA #ZERO     INITIALIZE TRACK COUNTER TO ZERO
9B82- 85 19    3400        STA TRK
9B84- A9 38    3410        LDA #TBMST    OFFSET = START OF TRACK BIT
                                         MAP IN VTOC
9B86- 85 1B    3420        STA OFFSET
9B88- A9 27    3430        LDA #INITV    INITIALIZE VERTICAL POSITION
9B8A- 85 1C    3440        STA VERPOS
9B8C- A9 04    3450        LDA #INITH    INITIALIZE HORIZONTAL POSITION
9B8E- 85 1D    3460        STA HORPOS
9B90- A4 1B    3470 LOOP1  LDY OFFSET    GET TBM OFFSET
9B92- B9 24 99 3480        LDA BUF,Y     GET 1ST BYTE OF BIT MAP FOR
                                         THIS TRACK
9B95- 85 08    3490        STA TBM       SAVE VALUE IN TBM
9B97- C8       3500        INY           BUMP INDEX TO NEXT BYTE
9B98- B9 24 99 3510        LDA BUF,Y     GET 2ND BYTE OF BIT MAP FOR
                                         THIS TRACK
9B9B- 85 09    3520        STA TBM+1     SAVE SECOND BYTE
9B9D- 20 B5 9B 3530        JSR NEWTRK    GO PLOT SECTOR STATUS FOR THIS TRACK
9BA0- A9 27    3540        LDA #INITV    RE-INIT. VERT. POSITION FOR
                                         NEXT TRACK
9BA2- 85 1C    3550        STA VERPOS
9BA4- E6 19    3560        INC TRK       BUMP THE TRACK COUNTER
9BA6- E6 1D    3570        INC HORPOS    BUMP HOR.POS. FOR NEXT TRACK
9BA8- A5 1B    3580        LDA OFFSET
9BAA- 69 04    3590        ADC #TBMINC   CALCULATE THE NEW OFFSET IN THE VTOC
9BAC- 85 1B    3600        STA OFFSET    AND SAVE IT
9BAE- A5 19    3610        LDA TRK       GET THE TRACK TO DO NEXT
9BB0- C9 23    3620        CMP #NOTRK    WAS THES THE LAST TRACK?
9BB2- D0 DC    3630        BNE LOOP1     IF NOT THEN LOOP
9BB4- 60       3640        RTS
               3650 *--------------------------------
               3660 *
               3670 *      ROUTINE CALLED BY 'MAP' TO PLOT A
               3680 *       (NOSEC) SECTOR TRACK AT
               3690 *       VERTICAL (INITV-MAPNO+VERPOS)
               3700 *       HORIZONTAL (INITH+(TRACK #))
               3710 *
               3720 *--------------------------------
               3730 NEWTRK
9BB5- A2 08    3740        LDX #NOLOOP   X = BIT/BYTE COUNTER
9BB7- E0 05    3742        CPX #5        CHECK FOR 13 SECTOR (X=5)
9BB9- D0 06    3743        BNE NEWBIT    SKIP NEXT 3 ROR'S IF 16 SECTOR
9BBB- 66 09    3744        ROR TBM+1     ONLY EXECUTED IF LINE 3740 MANUALLY
9BBD- 66 09    3745        ROR TBM+1       CHANGED TO 'LDX #5' FOR
                                         13 SECTOR DISK
9BBF- 66 09    3746        ROR TBM+1
9BC1- A9 0A    3750 NEWBIT LDA #UNUSED
9BC3- 20 64 F8 3760        JSR SETCOL    SET GR COLOR TO 'UNUSED' COLOR
9BC6- A4 1D    3770        LDY HORPOS    Y = HORIZONTAL POSITION
9BC8- 18       3780        CLC           CLEAR THE CARRY
9BC9- 66 09    3790        ROR TBM+1
9BCB- B0 05    3800        BCS FREE      IF BIT IN TBM+1 IS 1 THEN
                                         SECT. IS FREE
9BCD- A9 08    3810        LDA #USED
9BCF- 20 64 F8 3820        JSR SETCOL    OTHERWISE SET GR COLOR TO 'USED'COLOR
9BD2- A5 1C    3830 FREE   LDA VERPOS    NOW CALCULATE TRUE VERTICAL POSITION
9BD4- 18       3840        CLC
9BD5- E5 1A    3850        SBC MAPNO
9BD7- 20 00 F8 3860        JSR PLOT      A=VERT;Y=HOR.;SO PLOT SECTOR CELL
9BDA- C6 1C    3870        DEC VERPOS    DOWN COUNT THE VERTICAL
                                         POSITION OFFSET

9BDC- CA       3880        DEX
9BDD- D0 E2    3890        BNE NEWBIT    KEEP GOING IF BYTE NOT FINISHED
9BDF- A2 08    3900        LDX #NOLOOP   IF THE FIRST BYTE OF THE TBM IS DONE,
9BE1- A9 0A    3910 NEWBT2 LDA #UNUSED     THEN REPEAT THE ABOVE FOR 2ND BYTE
9BE3- 20 64 F8 3920        JSR SETCOL
9BE6- A4 1D    3930        LDY HORPOS
9BE8- 18       3940        CLC                       "
9BE9- 66 08    3950        ROR TBM
9BEB- B0 05    3960        BCS FREE2                 "
9BED- A9 08    3970        LDA #USED
9BEF- 20 64 F8 3980        JSR SETCOL                "
9BF2- A5 1C    3990 FREE2  LDA VERPOS
9BF4- 18       4000        CLC                       "
9BF5- E5 1A    4010        SBC MAPNO
9BF7- 20 00 F8 4020        JSR PLOT                  "
9BFA- C6 1C    4030        DEC VERPOS
9BFC- CA       4040        DEX                       "
```

(Continued)

MICRObits

(Continued from page 41)

Real Time Clock for AIM 65

Provides hour, minute, second, day of week, day, month, year. Twelve- or 24-hour format. Pin compatible with AIM expansion connector (also SYM, KIM). Four switch-selectable interrupts; Nicad battery backup; industrial quality board 4.5 × 6. All ICs socketed; single 5V supply; 22-page manual. All software included. Bare board $29. Complete A&T $93, includes batteries. Add $4 shipping and handling. California residents add 6%.

Data Design Group
7100 Mimosa Drive
Carlsbad, CA 92008
(714) 265-6940

OSI Machine Code Tracer

Use *DEBUG* to single step your computer and see all instructions execute. Adds tracing commands to OSI's Extended Monitor. Traces in ROM also. Adds breakpoint, high and low trace limits. All Extended Monitor commands retained. $12.95 for all 65D 5¼" systems.

DMP Systems
319 Hampton Blvd.
Rochester, NY 14612

Joystick Interface

For PET, AIM 65, SYM, KIM or other 6502-based computer. Uses five VIA ports to give eight-bit conversion of up to eight resistance devices. Requires 64 bytes of memory (software included). Assembled, tested — $29.95; bare board — $12.95.

Sydney S. Koegler
Micro-K Computer Products
2339 Carriage Ave.
Richland, WA 99352

OSI Superboard II, C1P

Tac-Man: Similar to *Pac-Man*, with ghosts, dots, power pills, cherries, and a great maze, plus fast, smooth action. For 8K only — $9.95. *Star-Fire*: Similar to *Defender*. 4K — $7.95; 8K — $9.95. Send $1.00 for a complete catalog. Cassette only.

Swany's OSI Software
2652 37th West
Seattle, WA 98199
(206) 282-7376

Lessons in Algebra

An easy and fun way to learn the basic elements of high school algebra. Apple computer diskette $29.95. 30-day money-back guarantee if not satisfied.

George Earl
1302 South General McMullen Dr.
San Antonio, TX 78237

(Continued on page 117)

Conclusion

This program provides valuable insight as to exactly where on the disk programs are located. This, in turn, allows you to speed up random access type programs significantly, and to determine when a garbage collection operation should be performed on often written-to disks.

One additional note: DISKMAP assumes that the VTOC accurately reflects the storage status of the disk. This is true 99.9% of the time. There are cases, however, where the VTOC will have gotten clobbered due to one of the following reasons:

- Assembly-language programs which directly manipulate the disk files without utilizing DOS.
- Interruption of AC power during a disk access (static discharge can cause the same thing).
- Manual manipulation of individual disk sectors using one of the sector read/write utility programs readily available on the market.

In all of these cases, DISKMAP will accurately reflect only what DOS (*via* the VTOC) thinks is used or unused.

Listing 1 *(Continued)*

```
9BFD- D0 E2     4050        BNE NEWBT2
9BFF- 60        4060        RTS                     AND RETURN TO DO NEXT
                                                    TRACK (IF ANY)
                4070 *-------------------------------------
                4080 *
                4090 *      ROUTINE TO PRINT ANY ASCII STRING WHOSE
                4100 *       ADDRESS IS FOUND AT (MESADD+1,MESADD)
                4110 *         STRING MUST END WITH C.R. ($0D)
                4120 *         AND CONTAIN LESS THAN 255 CHARACTERS
                4130 *
                4140 *-------------------------------------
9C00- A0 00     4150 PRINTM LDY #ZERO       ZERO TEXT CHARACTER INDEX
9C02- B1 06     4160 LOOP2  LDA (MESADD),Y DO AN INDEXED INDIRECT
                                           LOAD VIA MESADD
9C04- C9 0D     4170        CMP #CR        CHECK FOR CARRIAGE
                                           RETURN CHARACTER
9C06- D0 01     4180        BNE GOON       IF NOT THEN JUMP
9C08- 60        4190        RTS            OTHERWISE FINISHED, SO RETURN
9C09- 20 F0 FD 4200 GOON    JSR COUT1      OUTPUT THE CHARACTER
9C0C- C8        4210        INY            BUMP INDEX TO NEXT CHARACTER
9C0D- D0 F3     4220        BNE LOOP2      ALWAYS LOOP IF
                                           TEXT <= 255 CHARACTERS
9C0F- 60        4230        RTS            ONLY TAKEN IF
                                           TEXT > 255 CHARACTERS
                4240 *-------------------------------------
9C10- A9 41     4260 MM1    LDA #M1        PUT TEXT STRING ADDRESS
                                           INTO MESADD
9C12- 85 06     4270        STA MESADD
9C14- A9 9A     4280        LDA /M1         AND MESADD+1
9C16- 85 07     4290        STA MESADD+1
9C18- 20 00 9C 4300         JSR PRINTM     PRINT THE TEXT STRING
9C1B- 60        4310 ENDP   RTS            RETURN
```

Clyde Camp may be contaced at 3518 Wildflower Lane, Johnson City, Tennessee 37601.

FEELING A LITTLE OVERWHELMED

WITH APPLESOFT?

B.E.S.T. Our original Applesoft Optimizer (AOPT) has now been enhanced to bring to the Applesoft programmer the most comprehensive set of software tools available today—B.E.S.T. The BASIC ENHANCED SOFTWARE TOOLS is a machine language program to aid in writing, analyzing, and debugging Applesoft Basic programs. B.E.S.T. will certainly increase programmer efficiency for improved productivity. B.E.S.T. offers: 1) Variable Cross-Reference, 2) Line number Cross-Reference, 3) A Complete Cross-Reference, 4) Optimize variable names with recognition of both short (two character/ Applesoft standard) and long labels, 5) REM Optimizer—with an option to "protect" specified REM statements, 6) Line Optimizer to join short instruction lines of specified length, 7) Merging two programs, 8) Automat'c line numbering, and 9) A powerful, yet easy to use Renumber function. Implementation of B.E.S.T. can reduce memory requirements by up to 50% and increase execution speed of Applesoft programs by up to 100%. B.E.S.T. is a low cost alternative to an Applesoft compiler.

B.E.S.T. requires a 48K Apple II/II+, Applesoft in ROM or Language card, and DOS 3.3...... **$40.00**

EDIT-SOFT. A powerful, yet affordable, line editor for Applesoft. Using EDIT-SOFT, you can drastically cut your programming time. EDIT-SOFT not only contains the standard line editor features like inserting or deleting characters, moving to a specific character, entry of lower case letters, going to the beginning or end of a line, and displaying control characters, but it also has the advanced features that will prove indispensable:

- AUTO LINE NUMBERING
- SPLICING two lines together
- AUTOMATIC CHARACTER COUNTER for quoted strings makes screen formatting a snap
- EDITING LINES as they are being TYPED
- A STATUS LINE keeps you constantly aware of which options are currently being used
- Up to TWENTY MACROS available at any time
- MACRO EDITING AT ANY TIME
- HELP PAGES available when needed

Compare the features of EDIT-SOFT to other line editors, then compare the price. No other line editor has so many features at such a reasonable price!

EDIT-SOFT requires 48K of RAM, Applesoft in ROM (language and RAM expansion cards are fine), and DOS 3.3. ONLY **$30.00**

APLUS. The original Structured Basic enhancement for Applesoft Basic. APLUS provides the tools that will increase your efficiency, simplify program rewriting, and make your program flow more understandable. No longer do you have to remember the line numbers of subroutines and procedures; instead, just "DO GRAPHIT" and APLUS does the rest. APLUS adds the following structured programming commands to Applesoft basic: WHEN..ELSE..FIN, UNTIL, WHILE, CASE, SELECT, and OTHERWISE. Multi-line IF..FIN statements are also supported. APLUS includes functions to output indented listings to clarify logic flow and converting source code to standard Applesoft commands.

Requires Applesoft, 32K of RAM, and DOS 3.3...Only **$25.00.**

BUILD USING. Do you have trouble printing charts, reports, or formatting numbers on the screen? If you do, BUILD USING can solve your problems. BUILD USING is a powerful Applesoft utility which provides a "print-using" type routine for numbers and strings. By creating simple "formats", you tell BUILD USING how to format the output. The output from BUILD USING are strings which may be printed, written to disk, saved for later usage, or even reformatted. With BUILD USING, you can choose how many digits should be displayed to the left and right of the decimal point, and even fill the leading positions with the character of your choice. For example, you can print the nurnber '157.23' and '157.2' or '000157.230', or '****$157. AND 23/100 DOLLARS', or hundreds of other ways (including exponential formats). Working with strings is just as easy. Also included are three levels of error trapping, so you can correct numbers that cannot fit into your specified format.

Utilities like BUILD USING are usually difficult to use because they must be located in one memory location (usually between DOS and the DOS file buffers), they cannot be used with your favorite editor or other special routines. BUILD USING does not have this limitation, as it can be easily located in many different memory locations: 1) the "standard" between DOS and DOS file buffers, 2) at HIMEM, 3) APPENDED to your Applesoft program, or 4) anywhere else in memory. Appending BUILD USING to your program is as simple as EXECing a TEXT file. BUILD USING uses the "CALL" command thereby leaving the ampersand vector free for your own use.

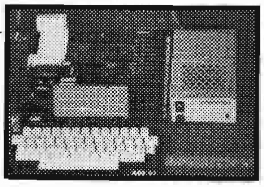
BUILD USING requires Applesoft in ROM (Language cards are fine), DOS 3.3 and a minimum of 32K...Only **$30.00**

Sensible Software
6619 Perham Drive Dept. MO
West Bloomfield, Michigan 48033
(313) 399-8877

Please specify program desired.
Visa and Mastercard Welcome. Add $1.25 postage and handling per diskette.

• Applesoft is a registered trademark of APPLE Computer Company •

AIM + POWER from COMPUTECH



All prices **postpaid** (Continental U.S.- otherwise $2 credit)

Check the **outstanding documentation** supplied with AIM65!

Top quality power supply designed to Rockwell's specs for fully populated AIM65 — includes overvoltage protection, transient suppression, metal case and power cable:

PSSBC-A (5V 2A Reg; 24V .5A Avg, 2.5A Peak, Unreg) **$64.95**
Same but an extra AMP at 5 volts to drive your extra boards:
PSSBC-3 (5V 3A Reg; 24V .5A Avg, 2.5A Peak, Unreg) **$74.95**

The **professional's choice** in microcomputers:

AIM65/1K RAM	**$429.95**	BASIC (2 ROMS)	**$59.95**
AIM65/4K RAM	**$464.95**	ASSEMBLER (1 ROM)	**$32.95**
		FORTH (2 ROMS)	**$59.95.**

SAVE EVEN MORE ON COMBINATIONS

AIM65/1K+PSSBC-A ... **$479.95** AIM65/4K+PSSBC-3 .. **$524.95**
We gladly quote on all AIM65/40 and RM65 items as well.

P.O. Box 20054 • Riverside, CA 92516
California residents add 6% sales tax

OSI **Disk Users**

Double your disk storage capacity
Without adding disk drives

Now you can more than double your usable floppy disk storage capacity—for a fraction of the cost of additional disk drives. Modular Systems' DiskDoubler™ is a double-density adapter that doubles the storage capacity of each disk track. The DiskDoubler plugs directly into an OSI disk interface board. No changes to hardware or software are required.

The DiskDoubler increases total disk space under OS-65U to 550K; under OS-65D to 473K for 8-inch floppies, to 163K for mini-floppies. With the DiskDoubler, each drive does the work of two. You can have more and larger programs, related files, and disk utilities on the same disk—for easier operation without constant disk changes.

Your OSI system is an investment in computing power. Get the full value from the disk hardware and software that you already own. Just write to us, and we'll send you the full story on the DiskDoubler, along with the rest of our growing family of products for OSI disk systems.

™DiskDoubler is a trademark of Modular Systems.

Modular Systems

Post Office Box 16C
Oradell, NJ 07649.0016
Telephone 201 262.0093

progressive computing

EXTENDED Z-FORTH IN ROM by Tom Zimmer	
5 to 10 times faster than Basic. Once you use it, you'll never go back to BASIC!	$100.00
A FULL 8K OPERATING SYSTEM source listing add	$ 20.00
OSI FIG-FORTH True fig FORTH model for 0S65D with fig editor named files, string package & much more	$ 45.00
STARTING FORTH by L. Brodie The best reference for Forth available to date. A must for the beginner, and refreshing to the experienced Forth Programmer, paperback only.	$15.95
TINY PASCAL Operates in fig-FORTH, an exceptional value when purchased with forth.	
Requires 32K min. TINY PASCAL & documentation	$ 45.00
FORTH & TINY PASCAL	$ 65.00
16 K RAM BOARD with 2K of EPROM The first and only memory board for 16 K of Ram plus 2 K of 2716 EPROM. Available as a bare board for the C1P only.	$39.95
PROGRAMMABLE CHARACTER GENERATOR Use OSI's graphics or make a complete set of your own! Easy to use, Bare Board Only.	$ 39.95
PROGRAMMABLE SOUND BOARD Complete sound system featuring the AY-3-8910 sound chip. Bare boards only.	$ 29.95
32/64 CHARACTER VIDEO MODIFICATION Oldest and most popular video mod. True 32 chr. C1P, or 32/64 chr. C4P video display.	$ 39.95
Optional Multiplexer board allows all three OSI Screens. Bare Board only.	$ 14.95
ROMS!!! Augment Video Mod with our Roms. Full screen editing, print at, selectable scroll, disc support and many more features.	
Basic 4 & Monitor	$ 44.95
Basic 3	$ 15.95
All 3 for	$ 59.95
65D DISASSEMBLY MANUAL. by Software Consultants First class throughout. A must for any 65D user.	$ 25.95

NUMEROUS BASIC PROGRAMS, UTILITY PROGRAMS AND GAMES ALONG WITH HARDWARE PROJECTS. ALL PRICES ARE U S FUNDS.
Send for our $1.50 catalogue with free program (hardcopy) and Auto Load Routine.
Canadian Residents: Price list in Canadian available on request.

OSI

Progressive Computing
Rick Lotoczky
3486 Countryside Circle
Pontiac Twp., MI 48057
(313) 373-0468

Joe Endre
3336 Avondale Court
Winsor, Ontario
Canada N9E 1X6
(519) 969-2500
(after 4:00)

OSI

Build An Apple Cart

by Tom Fisher and Michael Straka

Even if you live in an apartment and have only a few hand tools, it's possible for you to build a decent home for your Apple for less than $50.

Do you still have your computer perched on an old packing crate? Or maybe it's on the dining table and has to be moved before each meal? Well, here's an inexpensive way to give your Apple a home of its own.

Construction

The scale drawing (figure 1) and the parts list (table 1) provide sufficient information for the average handyperson to construct the AppleCart. The cart consists of several shelves, a back panel, two short and two tall side panels butted together, reinforcement, and casters. Simple design makes the AppleCart easy to construct. The desk (bottom) shelf is made from two shelves simply because the commercial vinyl-clad shelving isn't available in a 24'' width. The monitor (middle) shelf is shifted rearward to keep the screen away from your nose; the book (top) shelf is shifted forward to be more accessible to a seated operator. Because we expected to have to transport the cart in a small car occasionally, we used wood screws to provide for easy disassembly. The screws were positioned 2'' apart in the shelves and backpanel, and ¾'' vertically and 6'' horizontally elsewhere.

Additional Pointers

You can save money by cutting your own shelves and covering them with adhesive shelf paper. Buy good casters, but shop around. We found that prices for comparable casters vary by a factor of three or more!

Until the lumber industry learns to use a ruler, the dimensions shown in the drawing must be taken as approximations. The width of the particle board we purchased varied from less than 11½'' to almost 12''. Even the finished shelving varied by ¼'' from shelf to shelf. The design's most troublesome factor is the length of the shelf reinforcement; each piece must be cut to match the corresponding shelf.

Modifications

By now apartment dwellers are saying: ''That's fine if you have a 173 hp radial saw, an industrial-grade drill press, and deaf neighbors. But we can't build that in an apartment!'' While it's true that we waltzed down to the Physics Workshop to build our carts, the design can be modified so that it requires only hand tools and lots of glue. To minimize sawing, purchase the 60'' lengths of particle board (for the tall side panels) which are available from some lumber yards. If you can't get the short side panels cut at the yard, you'll only have to make two 12'' cuts with a handsaw.

***Table 1:* Parts List**

Three 60'' × 12'' particle board panels
Three 36'' × 12'' particle board panels
One 36'' × 16'' vinyl-clad shelf
One 36'' × 10'' vinyl-clad shelf
Two 36'' × 8'' vinyl-clad shelves
Seventy-six 10'' × 1½''slotted flathead wood screws
Four 2'' diameter rubber casters
Sixteen 10'' × 1½'' slotted roundhead wood screws

...or for apartment dwellers and others who don't have a power saw and an electric drill...

Cancel one of the 60'' × 12'' panels, the casters, and all the wood screws and add:

Five 36'' × 1'' diameter dowel rods
Wood glue

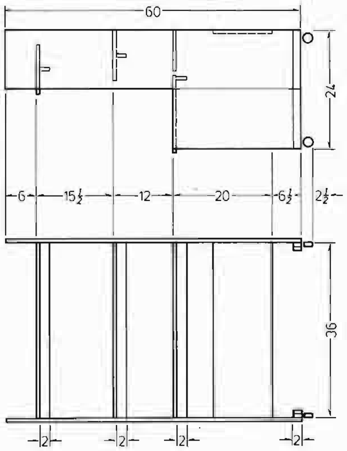


***Figure 1:* Scale drawing of the AppleCart. All dimensions are inches. The side panels, back panel, and shelf reinforcements are particle board; the four shelves are white vinyl-clad commercial shelving.**

Substitute 1" diameter dowel rods for the shelf reinforcement. It may take a while using the "file-and-fit" method, but at least you don't have to buy a radial saw. You'll have to forego casters if you use dowels to reinforce the base of the cart; just remember that the casters add 2½" to the height.

Using glue makes it unnecessary to drill, but requires that you construct a jig to hold the pieces while the glue dries. (Contact cement is not recommended because of the awkwardness of handling such large pieces of lumber.) So find a clear corner of the apartment and put some books of equal thickness (an encyclopedia set?) on the floor to create a surface that is higher than the baseboard. That corner, plus a few chairs and some heavy books, can serve as an overnight jig. Use wax paper to protect everything you don't want glued! Since the shelving (listed in the parts list) is covered with vinyl tape, this should be removed from areas that are to be glued, and those areas should be cleaned.

Improvements

More than 50 people have used the AppleCarts we built. Here are some ways they would improve them.

Many people asked: "When are you going to paint them?" We're not. We like the way they look, and furthermore, we're lazy. If you're going to paint yours, you might as well save some money by using particle board for the shelves, too. But you'll probably have to do lots of sanding.

Our secretary says the desk shelf is too high for convenient typing; you may want to lower the shelf about 2". She also doesn't like the monitor up so high. We like it there because we can conveniently remove the Apple's cover to eyeball the chips. So, if you use your Apple mostly for text editing, and you never mess with the Apple's innards, you may want to omit the monitor shelf and place the monitor on top of the Apple in the classical fashion.

If you install an undershelf light, position it so that it won't glare on the monitor screen. Also remember that, over a long period of time, a fluorescent light could erase your EPROMs!

An undershelf drawer could be used to store pencils and candy bars; you can fashion one from a silverware tray.

One essential attachment is a plugbar or powerstrip mounted on the rear panel, or perhaps at the very rear of the desk shelf.

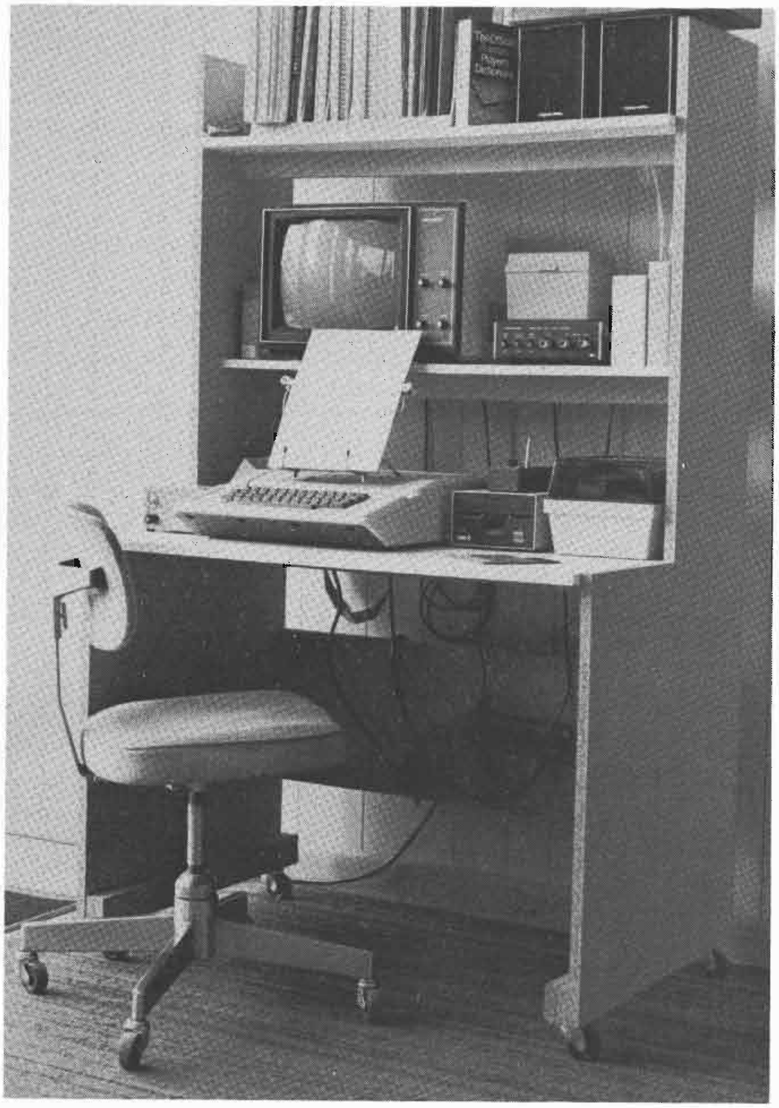
Acknowledgement

Our thanks to Juniata College for providing the facilities for this work, and to Dawn Herzberg, Science Secretary, for her outspoken evaluations.

Happy building!

Tom Fisher is an Associate Professor of Chemistry at Juniata College, Huntingdon, PA. Mike Straka, an alumnus of Juniata, is now a graduate student in the Chemistry Department at Miami University, Oxford, OH. Both are interested in applications of personal computers to chemical research. Contact them at Juniata College Chemistry Department, Huntingdon, PA 16652.

Figure 2: **The completed AppleCart. Photo by Ruth E. Reed**



S&H Software presents Apple II™ users with two chances to increase speed and productivity up to 500%.

Universal Boot Initializer 4.0 will work up to 500% faster than standard Apple DOS 3.3...$69.95

The new UBI 4.0 now includes The DOS Enhancer, a DOS-transparent routine that allows execution of Apple DOS 3.3 files (Integer, Applesoft and Binary) up to 500% faster than standard Apple DOS—depending on file length. In addition, a new "FREE" command in DOS now allows determination of free space on a disk in "any slot, any drive"—from the command mode or the program mode.

UBI 4.0-created disks increase efficiency by breaking the "language barrier" between Apple II hardware and software and breaking the "time barrier" by loading the RAM card with FPBASIC/INTBASIC (or leading assemblers) in 1.7 seconds. This unique combination of features greatly increases productivity: copyable disks, one-stage booting with DOS 3.3 or DOS 3.2.1 PROMS, fast-loading the RAM card with the "missing BASIC" (or your favorite utility), fast-(B)RUNning or fast-(B)LOADing your programs and complete compatibility with all DOS 3.3 programs.

The new UBI 4.0 package includes the utility disk, training disk, support disk, demo disk and complete documentation. System requirements: Apple II or Apple II Plus, ROM or RAM card, DOS 3.3 or 3.2.1 and one or more disk drives.

Amper-Sort/Merge (A-S/M) works up to 500% faster than even VisiCorp's VisiFile™ program...$52.95

The fastest "file clerk" you've ever met. Of all the sort utilities developed to manage Apple II data files, none does the job nearly so fast as Amper-Sort/Merge.

Here's a quick profile of "A-S/M": With 25K of working memory, one of five unsorted files can be sort/merged into a single file of up to 125K per disk. If a file to be sorted is more than 25K in length, the utility temporarily lays it aside to be sorted and merged when more memory space is available.

Because sorting routines take up to 50% of the computer running time in many business applications, you'll reap continuing benefits having this "invisible speed demon" on your Apple II team. We estimate that it will save twenty to thirty minutes a day of your "human" clerk's time—time that would otherwise be spent waiting for "sort/merge" operations.

The A-S/M "speed demon" package includes the utility disk, the training disk and 24 page instruction manual. System requirements: 48K Apple II, ROM or RAM card, DOS 3.3 and one or more disk drives or 48K Apple II Plus, DOS 3.3 and one or more disk drives.

To Order: *Send Check To* S&H Software, Box 5, Manvel, ND 58256 Credit Cards: *Phone* Cybertronics International Clearinghouse at 212 532-3089.

Box 5 Manvel ND 58256
(701) 696-2574

S&H Software

A feast of computing ideas.

NO. 41
$2.50
OCTOBER 1981
MICRO
THE 6502/6809 JOURNAL
12:80
Special PET/VIC Feature
A Look at the TRS-80 Color Computer
Step Up to Programmable Motion
A to D Converters for Your Computer

You'll love every byte.

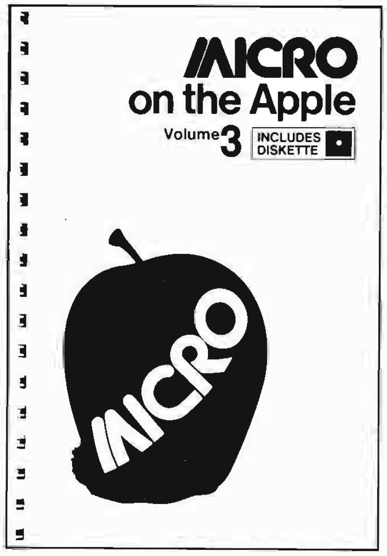
If you work with a 6502 or 6809 based system, you're probably hungry for the facts and ideas that will help you understand the inner workings of your computer. You want to go beyond canned software–use your computer for more than games–learn the advanced programming techniques that enable you to get the most out of your 6502/6809 system.

MICRO, The 6502/6809 Journal, gives you page after page, month after month, of solid information to sink your teeth into. **MICRO** is the premier how-to magazine for serious users of the Apple, PET/CBM, OSI, Atari, AIM, SYM, KIM, and all 6809 based systems including the TRS-80 Color Computer. It's a resource journal internationally respected by professionals in business, industry, and education.

SUBSCRIPTION RATES (U.S. dollars)
Yearly subscription (ISSN 027-9002) saves 20% off the single-issue price.
U.S., $24 (SPECIAL OFFER: Save 30% off single-issue price: 2 years, $42)
Other countries, $27 (via surface mail. Foreign air rates available on request.)

Get more out of your Apple . . .

with the MICRO ON THE APPLE series



VOLUME 3 just released!

More than 40 new programs on diskette to help you get more from your Apple:

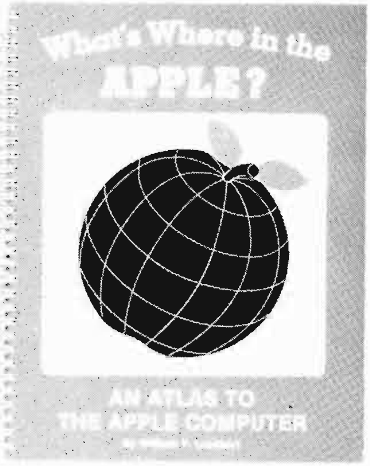
- Machine Language Aids
- I/O Enhancements
- Applesoft Aids
- Graphics and Games
- Reference Information

19 choice articles
43 tested programs on diskette
(16 sector DOS 3.3 format)

Volume 1 & 2 also available at $24.95.

Together **MICRO on the Apple 1, 2, & 3** provide more than 110 programs on diskette for less than $1.00 each. No need to type in hundreds of lines of code.

with the most important book ever published for the Apple



The most comprehensive description of Apple II firmware and hardware ever published—all in one place.

What's Where in the Apple?

- Guides you—with a numerical Atlas and an alphabetical Gazetteer—to over **2,000** memory locations of **PEEKs, POKEs** and **CALLs.**
- Gives names and locations of various **Monitor, DOS, Integer BASIC,** and **Applesoft** routines—and tells you what they're used for.
- Helps BASIC users to speed up their programs.
- Enables assembly language programmers to simplify coding and interfacing.

All Apple users will find this book helpful in understanding their machine, and essential for mastering it!

★ Look for all these **MICRO INK** publications at your local computer store, or

★ Call our **toll-free** number: **1-800-345-8112**
(In Pennsylvania, 1-800-662-2444) and charge it to your VISA or MasterCard, or

★ Use the order form below. Send your check (payable to MICRO) and the form to:
MICRO, Dept. OA, P.O. Box 6502, Chelmsford, MA 01824.

QTY	ITEM	PRICE EACH	COST
____	**MICRO on the Apple 1, 2, & 3** . .	@ $24.95	____
____	**What's Where in the Apple?** . . .	@ $14.95	____
____	**MICRO** (U.S.) . . . . 1 yr @ $24;	2 yrs @ $42	____
____	**MICRO** (Foreign)	1 yr @ $27	____
		Subtotal	____
	Massachusetts residents add 5% sales tax		____
	Add $2 **per book** for shipping MICRO Journal excluded Allow 4-6 weeks for delivery		____
		TOTAL	____

□ Check enclosed □ Charge my credit card below

Name ______________________

Company ______________________

Street ______________________

City ______________ State ______ ZIP ______

Signature ______________ □ VISA □ MasterCard

Card Number ______________ Expiration Date ______

MICRO™

Apple Slices

By Tim Osborn

Apple Slices is a new series designed for the do-it-yourselfer who wants to know more about the Apple II computer. Through a tutorial approach, Apple Slices will present concise programming examples and techniques for solving common programming problems.

If you are a beginner, this series will bring you up to speed on Apple II programming. If you are an intermediate to expert, Apple Slices will offer you helpful insights and alternative methods of programming.

No matter what level, if you're a do-it-yourselfer, you'll want to keep this series handy as a reference to Apple II programming techniques.

When it comes to speed of execution, nothing beats machine language. But this advantage must be weighed against the difficulty of rewriting Applesoft's built-in functions. So for most applications, Applesoft's ease of use compensates for its slower run-time performance. There are, however, occasions when Applesoft's speed is not acceptable, and we may wish to include machine-language subroutines to speed up the execution of critical functions. One of the problems that arises in doing this is in the passing of information to and from the assembly-language subroutines. This article will demonstrate one of the methods to accomplish this task.

Getting to the Subroutine

Before we talk about passing variables, I would like to review a popular method of calling machine-language subroutines, the ampersand (&) vector. The ampersand vector is one of the handiest, but least documented, features of Applesoft. When the interpreter finds an ampersand character at the beginning of an Applesoft statement, it executes a JSR $3F5. If the user has loaded a JMP instruction at $3F5 to a machine-language subroutine, the connection is effectively made. One smooth way to install this JMP is demonstrated by the routine named "SETVEC" in listing 1. Using this method, we need only BRUN the object file from BASIC, as in line 20 of listing 2. This method has the added feature of establishing HIMEM for you and thus protecting your subroutine from being destroyed by Applesoft.

Passing Parameters

Now that we can make the connection with the subroutine, let's look at parameter passing. A nice feature of the ampersand method is that the first non-space character following the ampersand is in the accumulator upon entry to your subroutine. Also, the Applesoft text pointer (TXTPTR) is pointing to this character in your BASIC program. If the ampersand is followed by an Applesoft command keyword, then the token for that command will be in the accumulator upon entry to your subroutine.

To take advantage of these features, we must first understand the following routines that are essential to the performance of the interpreter.

CHRGET ($B1)

This Applesoft routine accesses characters in your BASIC program. It uses TXTPTR ($B8-$B9) to point into the BASIC program. Upon entry, the TXTPTR is incremented and the value it then points to is loaded into the accumulator. Another entry point, which is called CHRGOT ($B7), does essentially the same — but does not increment TXTPTR. Upon exit from these subroutines, the zero status flag is set if we are at the end of a BASIC command, and the carry flag is clear if the value loaded was an ASCII number.

FIND ($E053)

This routine uses VARNAM ($81-$82), which should contain the two-byte variable name upon entry, and exits with the DSCPTR ($9B-$9C) pointing to the variable's descriptor. For the encoding of the variable names, Applesoft uses the following scheme:

	Byte-1	Byte-2
REAL	Positive	Positive
INTEGER	Negative	Negative
STRING	Positive	Negative

The second byte is null ($00) or negative ASCII null ($80) if it is a one-byte name. If you have the same version as I do of the Applesoft BASIC Programming Reference Manual, then page 137 contains an error. The manual shows string variable names as being encoded as negative ASCII, positive ASCII: this should be positive ASCII, negative ASCII as noted above. You may want to scratch in the correction as I did. Also, you may wish to check with page 137 to gain a better understanding of Applesoft's descriptor formats, (which are, by the way, the same for all versions of Microsoft BASIC).

An added feature of FIND is that it will create a descriptor for you if none is found.

DATA ($D995)

This routine sets the Text-pointer (TXTPTR) to the end of the current statement. It should be called before returning to BASIC; otherwise a syntax error may occur.

The Example

Listings 1 and 2 are an example of how we can put this all together to pass parameters back and forth between BASIC and machine language. In the example, we want to print either the left-most or right-most character of a string or assign the value of another string to X$. This example has purposely been simplified in order to illustrate the principle clearly. The BASIC program (listing 2) hands the machine-language program a string and a command. The machine-language program will actually perform the functions.

Once the machine-language program is installed *via* a BRUN, Applesoft will, upon encountering an ampersand, effectively JSR to the routine named "ENTRY" in listing 1. The first thing we need to do in our machine-language routine is to decide which function we have been called to perform. ENTRY does this by looking at the value in the accumulator and branching to the proper internal routine. I have chosen the Applesoft commands LEFT$ and RIGHT$ to print the left-most and

Example 1

```
]RUN
THIS IS A TESTER
T
R
PPPPPPPP%%&&&&&&&&
P
&
]
```

right-most byte of the passed string respectively. The equals sign ("=") was chosen to assign passed strings to X$. The syntax for the commands follows:

& RIGHT$ (< xx > $) : Print right-most byte of string specified by xx.

& LEFT$ (< xx> $) : Print left-most byte of string specified by xx.

& = (<xx > $) : Move string specified by xx to X$.

Once we know what function we have been called to perform, the next thing to do is to find out which string we have passed. We do this with a JSR to GETVAR, the internal subroutine to load VARNAM and JSR to FIND. I have included some error checking in GETVAR to make sure we have been passed a valid variable name. After we return from FIND, we index into the descriptor to get the length and save it, and then do the same for the address.

In the case of & RIGHT$, we load the Y register with the length and decrement it to establish an index to the right-most byte of the passed string. For & LEFT$, this index is always zero. After establishing the index, we JMP to OUTPUT, which uses the index and address to find the desired character and simply prints it. OUTPUT does a JSR to DATA to advance TXTPTR to the end of the statement. The RTS sends us back to BASIC.

The move (& =) is just a little more involved. After finding the address and length of the passed variable, we set up VARNAM to X$ ($58, $80) and JSR to FIND (which either finds X$ or establishes its descriptor). Either way, upon return we have the address of its descriptor in DSCPTR and the passed variable's address in VARPNT and length in LENGTH. X$'s address is known because it is internal to the subroutine at $80D0. Having all this information, we just need to move the passed variables bytes from VARPNT through VARPNT + LENGTH−1 to

Listing 1

```
0800             ;******************************
0800             ;*                            *
0800             ;*        APPLE SLICES        *
0800             ;*  DEMONSTRATION OF PASSING  *
0800             ;*    VARIABLES TO AND FROM   *
0800             ;*      MACHINE LANGUAGE      *
0800             ;*          T. S. O.          *
0800             ;*                            *
0800             ;******************************
0800             ;
0800             ;
0800             ;EQUATES
0800             ;
0073             HIMEM    EPZ $73              ;HIMEM POINTER
009B             DSCPTR   EPZ $9B              ;POINTER TO CURRENT VARIABLE D
ESCRIPTOR
0081             VARNAM   EPZ $81              ;CONTAINS LAST USED VARIABLE N
AME
00D0             WRKPTR   EPZ $D0              ;WORK POINTER FOR THIS PROGRAM

0083             VARPNT   EPZ $83              ; POINTER TO VARIABLE STORAGE
0800             ;
0800             ;APPLESOFT TOKENS
00E9             RIGHT$   EPZ $E9
00E8             LEFT$    EPZ $E8
00D0             EQL      EPZ $D0              ;EQUALS SIGN
0800             ;
0800             ;APPLESOFT BUILT IN ROUTINES
0800             ;
00B1             CHRGET   EPZ $B1              ;ROUTINE TO GET A CHARACTER
D995             DATA     EQU $D995            ;ROUTINE TO ADVANCE TXTPTR TO
END OF COMMAND
E053             FIND     EQU $E053            ;ROUTINE TO FIND DESCRIPTOR
DEC9             SYNERR   EQU $DEC9            ;ROUTINE TO DISPLAY ERROR MESS
AGE
0800             ;
0800             ;
03F5             AMPERV   EQU $3F5             ;AMPERSAND VECTOR ADDRESS
FDED             COUT     EQU $FDED            ;CHARACTER OUTPUT ROUTINE
0800             ;
0800             ;
8000                      ORG $8000
8000                      OBJ $800             ;FOR LISA
8000             ;SETVEC SETS UP THE AMPERSAND
8000             ;VECTOR ADDRESS TO ENTRY
8000             ;AND SETS HIMEM TO ENTRY TO PROTECT
8000             ;THIS ROUTINE FROM OVERWRITES.
8000             ;
8000 A9 4C       SETVEC   LDA #$4C             ;JUMP ABSOLUTE INSTRUCTION
8002 8D F5 03             STA AMPERV
8005 A9 14                LDA #ENTRY           ;LSB OF ENTRY ADDRESS
8007 8D F6 03             STA AMPERV+1
800A 85 73                STA HIMEM            ;SET HIMEM TO ENTRY
800C A9 80                LDA /ENTRY           ;MSB OF ENTRY ADDRESS
800E 8D F7 03             STA AMPERV+2
8011 85 74                STA HIMEM+1
8013 60                   RTS
8014             ;
8014             ;
8014 C9 E9       ENTRY    CMP #RIGHT$          ;DO WE WANT RIGHTMOST CHARACTE
R?
8016 D0 03                BNE LEFT$?
8018 4C 6D 80             JMP RIGHT            ;YES
801B C9 E8       LEFT$?   CMP #LEFT$           ;DO WE WANT LEFTMOST CHARACTER
?
801D D0 03                BNE MOVE?
801F 4C 79 80             JMP LEFT             ;YES
8022 C9 D0       MOVE?    CMP #EQL             ;DO WE WANT TO DO MOVE
8024 D0 03                BNE ERROR            ;IF NOT, THEN SYNTAX ERROR
8026 4C 86 80             JMP MOVE
8029 4C C9 DE    ERROR    JMP SYNERR           ;DISPLAY ERROR MESSAGE
802C             ;
802C             ;
802C 20 B1 00    GETVAR   JSR CHRGET           ;GET NEXT CHARACTER
802F F0 F8                BEQ ERROR            ;ERROR IF END OF LINE
8031 C9 28                CMP #'('             ;SHOULD BE LEFT PAREN
8033 D0 F4                BNE ERROR            ;IF NOT THEN SYNTAX ERROR
8035 20 B1 00             JSR CHRGET           ;GET NEXT CHARACTER
8038 F0 EF                BEQ ERROR            ;SHOULD NOT BE END OF LINE
803A 90 ED                BCC ERROR            ;SHOULD NOT BE ASCII NUMBER
803C 85 81                STA VARNAM           ;FIRST CHAR OF NAME
803E 20 B1 00             JSR CHRGET           ;GET NEXT CHARACTER
8041 F0 E6                BEQ ERROR            ;SHOULD NOT BE END OF LINE
8043 C9 24                CMP #'$'             ;DOLLAR SIGN?
8045 D0 02                BNE NAMLNG           ;NO, MUST BE TWO CHARACTER NAM
E
8047 A9 00                LDA #$00             ;YES, THIS IS A ONE CHARACTER
NAME
8049 09 80       NAMLNG   ORA #$80             ;NEGATIVE ASCII
804B 85 82                STA VARNAM+1
804D 20 53 E0             JSR FIND             ;FIND DESCRIPTOR
8050 A0 02                LDY #2
8052 B1 9B                LDA (DSCPTR),Y       ;GET AND SAVE THE
8054 8D CF 80             STA LENGTH           ;LENGTH OF PASSED STRING
8057 C8                   INY
8058 B1 9B                LDA (DSCPTR),Y       ;GET AND SAVE THE
805A 85 83                STA VARPNT           ;ADDRESS OF PASSED STRING
805C C8                   INY
805D B1 9B                LDA (DSCPTR),Y
```

Listing 1 *(Continued)*

```
805F 85 84          91              STA VARPNT+1
8061 60             92              RTS
8062                93   ;
8062                94   ;
8062 B1 83          95   OUTPUT     LDA (VARPNT),Y        ;GET CHARACTER TO PRINT
8064 09 80          96              ORA #$80              ;APPLE NORMAL IS NEG-ASCII (OT
HERWISE WILL PRINT INVERSE)
8066 20 ED FD       97              JSR COUT              ;AND PRINT IT
8069 20 95 D9       98   NOTFND     JSR DATA              ;POINT TXTPTR AT NEXT COMMAND

806C 60             99              RTS                   ;RETURN TO BASIC
806D               100   ;
806D               101   ;
806D 20 2C 80      102   RIGHT      JSR GETVAR            ;GET LENGTH AND ADDRESS
8070 AC CF 80      103              LDY LENGTH            ;INDEX TO RIGHTMOST CHARACTER
8073 F0 F4         104              BEQ NOTFND            ;LENGTH=0 MEANS STRING NOT FOU
ND
8075 88            105              DEY
8076 4C 62 80      106              JMP OUTPUT            ;OUTPUT AND RETURN TO BASIC
8079               107   ;
8079               108   ;
8079 20 2C 80      109   LEFT       JSR GETVAR            ;GET LENGTH AND ADDRESS
807C AC CF 80      110              LDY LENGTH            ;LENGTH=0 MEANS
807F F0 E8         111              BEQ NOTFND            ;STRING NOT FOUND
8081 A0 00         112              LDY #$00              ;INDEX TO LEFTMOST CHARACTER
8083 4C 62 80      113              JMP OUTPUT            ;OUTPUT AND RETURN TO BASIC
8086               114   ;
8086               115   ;
8086 20 2C 80      116   MOVE       JSR GETVAR            ;GET LENGTH AND ADDRESS
8089 AD CF 80      117              LDA LENGTH            ;LENGTH=0
808C F0 DB         118              BEQ NOTFND            ;MEANS STRING NOT FOUND
808E A5 83         119              LDA VARPNT            ;SAVE VARPNT FOR LATER USE
8090 85 D0         120              STA WRKPTR
8092 A5 84         121              LDA VARPNT+1
8094 85 D1         122              STA WRKPTR+1
8096 A9 58         123              LDA #'X'
8098 85 81         124              STA VARNAM
809A A9 80         125              LDA #$80              ;NEGATIVE ASCII NULL
809C 85 82         126              STA VARNAM+1
809E 20 53 E0      127              JSR FIND              ;FIND DESCRIPTOR
80A1 A5 D0         128              LDA WRKPTR            ;RESTORE VARPNT
80A3 85 83         129              STA VARPNT
80A5 A5 D1         130              LDA WRKPTR+1
80A7 85 84         131              STA VARPNT+1
80A9 A0 02         132              LDY #2
80AB AD CF 80      133              LDA LENGTH            ;LENGTH OF PASSED STRING
80AE 91 9B         134              STA (DSCPTR),Y        ;STORE LENGTH IN X'S DESCRIPTO
R
80B0 A9 D0         135              LDA #X$               ;LSB OF X'S ADDRESS
80B2 C8            136              INY
80B3 91 9B         137              STA (DSCPTR),Y        ;STORE ADDRESS IN X'S DESCRIPT
OR
80B5 85 D0         138              STA WRKPTR            ;SET UP WORKPOINTER
80B7 A9 80         139              LDA /X$
80B9 C8            140              INY
80BA 91 9B         141              STA (DSCPTR),Y
80BC 85 D1         142              STA WRKPTR+1
80BE AE CF 80      143              LDX LENGTH            ;SET UP CHARACTER COUNTER
80C1 A0 00         144              LDY #$0
80C3 B1 83         145   MVLOOP     LDA (VARPNT),Y        ;GET PASSED STRING
80C5 91 D0         146              STA (WRKPTR),Y        ;AND MOVE TO X
80C7 C8            147              INY
80C8 CA            148              DEX                   ;DECREMENT CHARACTER COUNTER
80C9 D0 F8         149              BNE MVLOOP
80CB 20 95 D9      150              JSR DATA              ;MOVE TXTPTR TO NEXT COMMAND
80CE 60            151              RTS
80CF               152   ;
80CF               153   ;INTERNAL STORAGE AREA
80CF               154   ;
80CF               155   LENGTH     DFS $1
80D0               156   X$         DFS $100
81D0               157   ;
81D0               158   ;
81D0               159              END
```

Listing 2

```
10 CD$ =  CHR$ (4)
20  PRINT CD$"BRUN ARTICLE.CODE,A$8000"
30 A$ = "THIS IS A TESTER"
40  &  = (A$)
50  PRINT X$
60  &  LEFT$ (A$)
65  PRINT
70  &  RIGHT$ (A$)
75  PRINT
80 BB$ = "PPPPPPPP%%&&&&&&&&"
90  &  = (BB$)
100  PRINT X$
110  &  LEFT$ (X$)
115  PRINT
120  &  RIGHT$ (X$)
```

$80D0 through $80D0 + LENGTH – 1, stuff $80D0 into the address portion of X$'s descriptor, and LENGTH into the length portion. After that is accomplished, we JSR to DATA and RTS back to BASIC. Once back in BASIC, we can access X$ just like any other variable.

Follow through listings 1 and 2 and look at the output in example 1 to clarify these points. This basic system of passing parameters can be expanded to include REAL and INTEGER simple variables. If you have any questions, problems, or suggestions for future topics, please feel free to drop me a note.

References

1. Apple Computer Inc., "Applesoft Variable Maps," pg. 137, *Applesoft Programming Reference Manual.*

2. Crossley, "Applesoft Internal Entry Points," *The Apple Orchard,* March/April 1980.

3. Cornelis Bongers, "Applesoft's CHARGET Routine," *Call —A.P.P.L.E.,* March 1982.

MICRO

Low-Resolution Graphics for Apple Pascal

by Richard C. Vile, Jr.

This article and accompanying programs will provide a method for accessing Apple's low-resolution capabilities from Apple Pascal.

Requires

Apple II or Apple II Plus
with Apple Pascal

An Apple II feature absent in Apple Pascal is low-resolution graphics. Although many users may not miss lo-res graphics, I certainly did. I decided to see what could be done. The results of my efforts are presented in this article.

Desirable Low-Resolution Capabilities

If low-resolution capabilities are to be provided in Pascal, we must add new procedures and functions. A reasonable list resembles the features found in BASIC:

```
PROCEDURE plot(row,col:INTEGER);
PROCEDURE hline(col1,col2,row:
  INTEGER);
PROCEDURE vline(row1,row2,col:
  INTEGER);
PROCEDURE setcolor(c:INTEGER);
PROCEDURE grclear;
PROCEDURE grcltop;
PROCEDURE setlow;
PROCEDURE settext;
FUNCTION scrn(row,col:INTEGER):
  INTEGER;
```

Most of these capabilities are already present in Apple's Monitor ROMs. Therefore, it would seem that we could simply use the Pascal 6502 macro assembler to create a code file containing the appropriate parts of the monitor. Then we could link that file to whatever high-level Pascal code we create to use the low-resolution features. This idea is feasible in principle, but it needs a little remodeling to put it to work.

The assembly code in listing 1 provides the capabilities listed above in the form of Pascal EXTERNAL procedures and functions. You need to assemble that listing using the assembler in the language system. This will produce a code file, which we will call LOWRES.CODE here, but you can call it whatever you like. Listing 2 presents a typical Pascal program that uses the lo-res features. Notice that the procedures and functions are declared at the beginning of the program as EXTERNAL. You will need to duplicate these declarations in any of your own programs that use LOWRES.CODE.

Suppose that the program of listing 2 were compiled and a code file named VIDEO.CODE were produced. It would be necessary to link the code files LOWRES.CODE and VIDEO.CODE together to produce a useable Pascal program. To do this, you must invoke the Pascal linker program (after putting both code files on the disk containing the linker). The dialogue which will ensue should look as follows:

```
LINKER II.1 [A4]
HOST FILE? VIDEO
OPENING VIDEO.CODE
LIB FILE? LOWRES
OPENING LOWRES.CODE
LIB FILE?
MAP FILE?
READING VIDEO
READING IROUTINE
OUTPUT FILE? DEMO
LINKING VIDEO #1
COPYING PROC IROUTINE
COPYING PROC PLOT
COPYING PROC HLINE
COPYING PROC VLINE
COPYING FUNC SCRN
COPYING PROC SETCOLOR
COPYING PROC GRCLEAR
COPYING PROC GRCLTOP
COPYING PROC SETLOW
COPYING PROC SETTEXT
```

The PROCs and FUNCs that are indicated as being copied are all those which you declare as EXTERNAL in your Pascal host program. If you don't use a particular PROC or FUNC, then don't declare it. Otherwise it will clutter up your program.

The Nature of EXTERNAL Procedures in Apple Pascal

To call assembly code from Apple Pascal programs, the code must resemble either a procedure or a function to the caller (or host as it is referred to in the manuals). This means that the assembly code must have a .PROC or .FUNC declaration surrounding it. Otherwise, the language system assembler detects a syntax error.

If an external routine is called from a host and parameters are passed to it, they will be transmitted on the 6502 stack. This means that pre-existing code, such as the lo-res routines from the monitor that expect parameters in registers, may not be called directly by a host program. That is, if we simply put a .PROC declaration in front of the code and call it from Pascal, it is not going to operate correctly because it won't get its parameters as expected.

To remedy this situation, write interface routines, which the Pascal host program can call. Their only job is to take the parameters passed on the stack, rearrange them into the appropriate registers, and call the pre-existing code. Figure 1 illustrates this idea.

To clarify the schematic explanation of figure 1, we consider the EXTERNAL procedure PLOT. It takes two arguments; namely, row and col, both of type INTEGER. The Pascal compiler generates code that causes the arguments to be passed on the stack, both occupying two bytes. Figure 2 illustrates the configuration of the 6502 stack when the routine PLOT is entered. Notice that the return address to the Pascal calling routine occupies the top two bytes. The assembly code must save this address. Later it will be restored to the stack top so that the RTS instruction ending the PLOT procedure gets back to the caller. The macros POP and PUSH have been included for this purpose.

After POP is invoked to store the return address, the two parameters are exposed. As pointed out above, they both occupy a full word (two bytes). However, in both cases only the low byte of the word argument is significant. This explains the need for the extra PLA instruction to process each parameter. Figure 3 shows what happens as the parameters are rearranged. After putting row and col into the appropriate 6502 registers, PLOT simply does a JSR to the internal plot routine, which we have called IPLOT. IPLOT is identical to the monitor PLOT routine, which expects its inputs in the A and Y registers.

Summary

We have presented assembly code that allows a Pascal user to access the Apple II low-resolution graphics capabilities. The implementation illustrates the use of the 6502 assembler in the language system. It also shows the need for interface routines when trying to use assembly code that expects parameters to be passed in registers. Finally, it illustrates the use of the linker program in the language system in order to tie the assembly code and the Pascal host code together.

If you want to extend the capabilities presented here, you should attempt to add other procedures and functions. One obvious feature (missing in BASIC) is a procedure, SETFULL, that will select full-screen low-resolution graphics. Other possibilities might involve procedures that produce more complex graphics elements than points or lines. For example, you could add BLOCK(col1,col2,row1,row2:INTEGER); which draws a solid block with the boundaries indicated.

Contact Richard Vile at 3467 Yellowstone Drive, Ann Arbor, Michigan 48105.

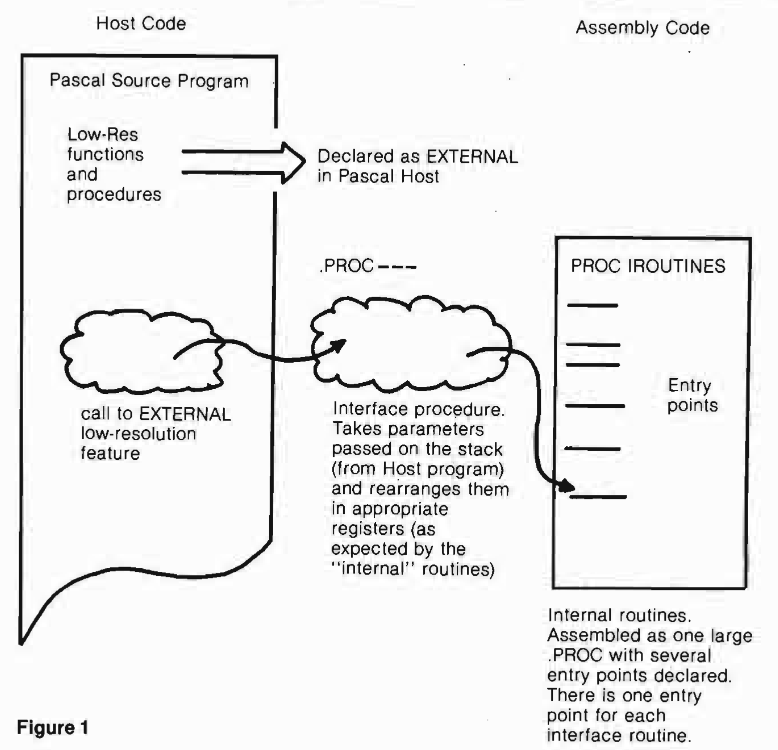

Host Code
Assembly Code
Pascal Source Program
Low-Res functions and procedures
Declared as EXTERNAL in Pascal Host
.PROC ---
PROC IROUTINES
Entry points
call to EXTERNAL low-resolution feature
Interface procedure. Takes parameters passed on the stack (from Host program) and rearranges them in appropriate registers (as expected by the "internal" routines)
Internal routines. Assembled as one large .PROC with several entry points declared. There is one entry point for each interface routine.

Figure 1

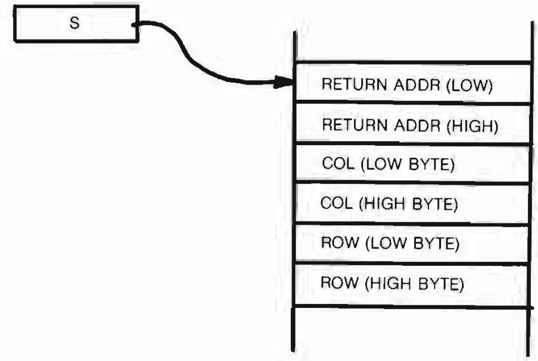


Figure 2

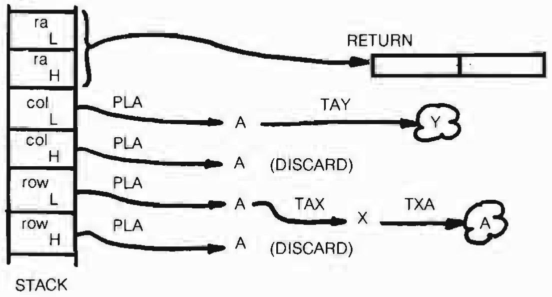


Figure 3

Listing 1

```
;-----------------------------------------------------------
;  low resolution routines "stolen" from the rom monitor
;-----------------------------------------------------------


;-----------------------------------------------------------
;  set up page zero temporaries matching the original
;  apple ][ system monitor definitions of the graphics
;  locations.
;-----------------------------------------------------------


GBASL   .EQU     26
GBASH   .EQU     27
H2      .EQU     2C
V2      .EQU     2D
MASK    .EQU     2E
COLOR   .EQU     30
GRAPHIC .EQU  0C050
TEXTMOD .EQU  0C051
FULLSCR .EQU  0C052
MIXED   .EQU  0C053
LORES   .EQU  0C056

;-----------------------------------------------------------
;  define the macros push and pop which are used in all
;  the pascal external routines to save and restore
;  the pascal return address.
;-----------------------------------------------------------


        .MACRO PUSH

        LDA %1+1
        PHA
        LDA %1
        PHA

        .ENDM

        .MACRO POP

        PLA
        STA %1
        PLA
        STA %1+1

        .ENDM

;-----------------------------------------------------------
;  first duplicate the code for low resolution graphics
;  routines from the rom monitor.  these routines are
;  the "internal" routines to be called from the pascal
;  external interface routines which set up the
;  parameters passed from above.
;-----------------------------------------------------------


        .PROC IROUTINES
        .DEF  IPLOT,IHLIN,IVLIN
        .DEF  IGRCLEAR,ICLEAR,ISCRN
        .DEF  GBASCALC,ISETGR,ITEXT
        .DEF  IWAIT,ISETCOL

IPLOT   LSR A
        PHP
        JSR GBASCALC
        PLP
        LDA #0F
        BCC RTMASK
        ADC #0E0

RTMASK  STA MASK
PLOT1   LDA @GBASL,Y
        EOR COLOR
        AND MASK
        EOR @GBASL,Y
        STA @GBASL,Y
        RTS

IHLIN   JSR IPLOT
HLINE1  CPY H2
        BCS RTS1
        INY
        JSR PLOT1
        BCC HLINE1
VLINEZ  ADC #01
IVLIN   PHA
        JSR IPLOT
        PLA
        CMP V2
        BCC VLINEZ
RTS1    RTS

IGRCLEAR LDY #2F
        BNE CLRSC2

ICLEAR  LDY #27
CLRSC2  STY V2
        LDY #27

CLRSC3  LDA #0

        STA COLOR
        JSR IVLIN
        DEY
        BPL CLRSC3
        RTS
```

Listing 1 *(Continued)*

```
GBASCALC PHA
        LSR A
        AND #03
        ORA #04
        STA GBASH
        PLA
        AND #18
        BCC GBCALC
        ADC #7F
GBCALC  STA GBASL
        ASL A
        ASL A
        ORA GBASL
        STA GBASL
        RTS

ISCRN   LSR A
        PHP
        JSR GBASCALC
        LDA @GBASL,Y
        PLP
SCRN2   BCC RTMSKZ
        LSR A
        LSR A
        LSR A
        LSR A
RTMSKZ  AND #0F
        RTS

ISETCOL AND #0F
        STA COLOR
        ASL A
        ASL A
        ASL A
        ASL A
        ORA COLOR
        STA COLOR
        RTS

ISETGR  LDA GRAPHIC
        LDA LORES
        LDA MIXED
        JSR ICLEAR
        RTS

ITEXT   LDA TEXTMOD
        LDA FULLSCR

        LDA #0A0
        JSR ICLEAR
        RTS

IWAIT   SEC
WAIT2   PHA
WAIT3   SBC #01
        BNE WAIT3
        PLA
        SBC #01
        BNE WAIT2
        RTS

;-----------------------------------------------------------
;  now finally the code for the external routines.
;  each routine is responsible for picking up the
;  parameters from the stack and calling the parallel
;  internal routine.
;-----------------------------------------------------------


        ;-----------------
        ;   p l o t
        ;-----------------


        .PROC PLOT,2      ;PROCEDURE PLOT(ROW,COL:INTEGER);
        .REF IPLOT

RETURN  .EQU    0

        POP RETURN

        PLA
        TAY               ;COL ARGUMENT
        PLA               ;DISCARD MSB OF ARGUMENT
        PLA
        TAX
        PLA               ;DISCARD MSB
        TXA               ;RESTORE ROW ARGUMENT TO A-REG
        JSR IPLOT         ;CALL INTERNAL ROUTINE

        PUSH RETURN
        RTS               ;RETURN TO PASCAL CALLER


        ;----------------
        ; H  L  I  N  E
        ;----------------

        .PROC HLINE,3     ;PROCEDURE HLINE(COL1,COL2,ROW:INTEGER);
        .REF  IHLIN
```

(Continued)

Listing 1 *(Continued)*

```
RETURN  .EQU    0

        POP RETURN

        PLA
        TAX             ;SAVE ROW IN X-REG
        PLA             ;DISCARD HIGH BYTE OF ARG.
        PLA             ;GET ENDING COLUMN
        STA H2          ;SAVE IN PAGE ZERO TEMP
        PLA             ;TOSS H.B.
        PLA             ;GET STARTING COLUMN
        TAY             ;PUT IN Y-REG
        PLA             ;TOSS H.B.
        TXA             ;PUT ROW BACK IN ACC

        JSR IHLIN       ;CALL INTERNAL ROUTINE

        PUSH RETURN

        RTS


        ;-----------------
        ;  V  L  I  N  E
        ;-----------------

        .PROC VLINE,3   ;PROCEDURE VLINE(ROW1,ROW2,COL:INTEGER);
        .REF  IVLIN

RETURN  .EQU    0

        POP RETURN

        PLA             ;GET COLUMN
        TAY             ;ITS EXPECTED IN Y-REG
        PLA             ;DICARD H.B. OF ARG
        PLA             ;GET ENDING ROW
        STA V2          ;PUT IN PAGE ZERO TEMP
        PLA             ;TOSS H.B.
        PLA             ;GET STARTING ROW
        TAX
        PLA             ;TOSS H.B.
        TXA             ;ROW INTO ACC

        JSR IVLIN       ;CALL INTERNAL ROUTINE

        PUSH RETURN
        RTS
        ;-----------------
        ;  S  C  R  N
        ;-----------------

        .FUNC SCRN,2    ;FUNCTION SCREEN(ROW,COL:INTEGER):INTEGER;
        .REF  ISCRN

RETURN  .EQU    0

        POP RETURN

        PLA
        PLA
        PLA
        PLA             ;DISCARD 4 BYTES FOR FUNCTIONS

        PLA
        TAY             ;COLUMN
        PLA
        PLA             ;ROW
        TAX
        PLA
        TXA
        JSR ISCRN

        TAX             ;SAVE RESULT
        LDA #00
        PHA             ;PUSH MSB = 0
        TXA
        PHA             ;RESULT FROM ISCRN

        PUSH RETURN
        RTS


        ;-------------------------
        ;  S  E  T  C  O  L  O  R
        ;-------------------------


        .PROC SETCOLOR,1
        .REF  ISETCOL

RETURN  .EQU 0

        POP RETURN

        PLA
        TAX
        PLA             ;DISCARD MSB
        TXA

        JSR ISETCOL

        PUSH RETURN
        RTS             ;RETURN TO PASCAL CALLER
```

Listing 1 *(Continued)*

```
        ;-----------------------
        ;  G  R  C  L  E  A  R
        ;-----------------------


        .PROC GRCLEAR   ;PROCEDURE GRCLEAR;
        .REF  IGRCLEAR

RETURN  .EQU    0

        POP RETURN

        JSR IGRCLEAR    ;CALL INTERNAL ROUTINE
                        ;NO ARGUMENTS TO SET UP

        PUSH RETURN
        RTS


        ;--------------------------
        ;  G  R  C  L  R  T  O  P
        ;--------------------------

        .PROC GRCLTOP   ;PROCEDURE GRCLTOP
        .REF ICLEAR

RETURN  .EOU    0

        POP RETURN

        JSR ICLEAR      ;CALL INTERNAL ROUTINE

        PUSH RETURN
        RTS


        ;------------------------
        ;  S  E  T  L  O  W
        ;------------------------


        .PROC SETLOW    ;PROCEDURE SETLOW;
        .REF ISETGR

RETURN  .EQU    0

        POP RETURN

        JSR ISETGR
        PUSH RETURN
        RTS


        ;----------------------
        ;  S  E  T  T  E  X  T
        ;----------------------


        .PROC SETTEXT   ;PROCEDURE SETTEXT
        .REF  ITEXT

RETURN  .EQU    0

        POP RETURN
        JSR ITEXT
        PUSH RETURN
        RTS

        .END
```

Listing 2

```
(***************************************)
(*                                     *)
(*      v    i    d .  e    o          *)
(*                                     *)
(*  program to test the pascal low     *)
(* resolution graphics interface to    *)
(* assembly routines.                  *)
(*                                     *)
(***************************************)

PROGRAM video;

  USES applestuff;

CONST

  lores          =      -16298;
  graphics       =      -16304;
  mixtext        =      -16301;
  alltext        =      -16303;
  fullscreen     =      -16302;
  color          =          48;
  escape         =          27;
```

(Continued)

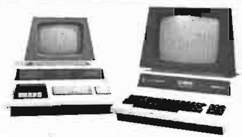
Listing 2 *(Continued)*

```
 VAR
  colors:         0..15;
  location:      INTEGER;

  r1,
  r2,
  c1,
  c2:            INTEGER;
  width,
  which,
  i,
  j:             INTEGER;
  ch:            CHAR;

  down:          BOOLEAN;

PROCEDURE plot(row,col:INTEGER); EXTERNAL;
PROCEDURE hline(col1,col2,row:INTEGER); EXTERNAL;
PROCEDURE vline(row1,row2,col:INTEGER); EXTERNAL;
PROCEDURE setcolor(c:INTEGER); EXTERNAL;
PROCEDURE grclear; EXTERNAL;
PROCEDURE grcltop; EXTERNAL;
PROCEDURE setlow;  EXTERNAL;
PROCEDURE settext; EXTERNAL;

(**************************************)
(*          r    n    d               *)
(**************************************)

FUNCTION rnd(a,b:INTEGER):INTEGER;
BEGIN

  rnd := a + random MOD (b - a + 1);

END (* FUNCTION rnd *);
(**************************************)
(*      t   i   c   t   a   c         *)
(**************************************)

 PROCEDURE tictac(v:INTEGER);
 BEGIN
    hline(0,39,v);
    hline(0,39,39-v);
    vline(0,39,v);
    vline(0,39,39-v);

  END (* PROCEDURE tictac *);
(**************************************)
(*     q   u   a   d   t   a   c      *)
(**************************************)
PROCEDURE quadtac(v:INTEGER);
BEGIN
    hline(0,39,v);
    hline(0,39,39-v);
    vline(0,39,v);
    vline(0,39,39-v);
    hline(0,39,20+v);
    hline(0,39,19-v);
    vline(0,39,20+v);
    vline(0,39,19-v);
  END;

  BEGIN
    write('input width===>');
    readln(width);

    setlow;
    randomize;

    ch := ' ';
    i  := 0;
    down := true;

  setcolor(8);

  WHILE ch <> chr(escape) DO
  BEGIN
```

Listing 2 *(Continued)*

```
    IF rnd(0,width)=0
    THEN
    BEGIN
      which := rnd(0,15);
      setcolor(which);
    END;

    IF down
    THEN
    BEGIN
      quadtac(i);
      i := i + 1;
      IF i = 20
      THEN
        down := false
      (* endif *);
    END
    ELSE
    BEGIN
      i := i - 1;
      quadtac(i);
      IF i = 0
      THEN
        down := true
      (* endif *);

    END (* IF down *);
    IF keypress
    THEN
      read(ch)
    (* endif *);

  END (* WHILE ch <> chr(escape) *);

  settext;
END.
```

RPL



64K RAM for AIM-65® and SYM-1® from BYTE MICROSYSTEMS

MICRO™

PET Vet

By Loren Wright

POWER — A Flexible ROM Utility Package

In the May PET Vet I discussed the many virtues of the PET's system for editing BASIC programs. However, there are a few deficiencies, which have been left for the user to correct or put up with. An early cure for some of these ills was the Programmer's Toolkit ROM from Palo Alto Integrated Circuits. For good reasons, it was instantly very popular and it remains so. It provides auto-numbering, renumbering, delete, search, dump, and trace functions. Lately, larger and more powerful utility packages have been introduced, which provide these and other capabilities. POWER from Professional Software combines some very unusual and powerful features with an excellent set of editing, testing, and debugging commands.

Editing Enhancement

The editing commands in POWER include AUTOnumber, RENumber, and DELete. The renumber command allows you to renumber any part of your program or the whole thing. POWER adds repeating cursor keys (to those machines that don't already have them) and a handy list scrolling feature.

Testing and Debugging Aids

Another group of commands help in testing and debugging. There are powerful search and search-and-replace commands. You can have either command act on only the first occurrence of the search string (with an easy repeat), or you can have the command operate globally. In addition, you can enable special characters called meta-characters. These, when included in your search string, indicate that you don't care about a particular character or series of characters.

POWER has a TRaCe command, which you can select to operate on several levels. The most complete version displays each BASIC line number, its contents, and the values of variables as they are assigned. By holding down a single key you can quickly trace through parts of your program, and then single step by pressing the key only once for each line.

The DUMp command displays the values of all non-array variables. WHY indicates the cause of an error by highlighting the part of the BASIC line that caused a run-time error.

The features described so far are essentially the same (with improvements) as those included in the Programmer's Toolkit. The remaining features are really what distinguish POWER from the Toolkit and the many other utility packages now available.

Instant Keywords, Phrases, and Subroutines

At some point you probably learned that instead of typing out long BASIC keywords like VERIFY, RESTORE, RETURN, and COLLECT, you could get away with typing only the first one or two letters followed by the next one shifted. With POWER, if you select the instant keyword feature, nearly every key, when shifted, causes a BASIC keyword to be spelled out on the screen. For instance, shift-B is GOSUB, shift-L is LIST, and shift-R is RETURN.

You can, however, redefine any of these keys to produce "instant phrases." These are defined in special REM statements that you enter in your program (and delete when you're through with them). To assign a phrase meaning to the shifted W you would enter a line in your program such as:

```
1 REM"W=GOSUB 5000: X= -1
```

This would cause "GOSUB 5000: X= -1" to be displayed on the screen every time you type a shifted W. Of course you can reassign as many keys as you want, and the others will retain their original keyword meaning.

The final "instant" feature is instant subroutines. These are defined in a similar way to instant phrases. The subroutine, which must exist in your current program, is executed when the appropriated shifted key is typed. Unlike instant phrases, instant subroutines can include GET and INPUT commands, and of course they can include more than one line. There are lots of possibilities for instant subroutines, such as generating an array dump.

You can customize your keyboard to include an appropriate balance of instant keywords, phrases, and subroutines. If you need to use the shifted keys as they were originally intended, it is a simple matter to disable the instant features.

The XEC Command and Others

The XEC command transfers control to a PET sequential file. The lines of the file are executed as if they were typed in from the keyboard. The most obvious application of this command is to merge programs or subroutines that have been stored as sequential files.

There is an OFF command to completely disable POWER and restore the PET's pointers and CHRGET routine to their normal states. The SEL command is used to enable and disable instant keywords, phrases, and subroutines; meta-characters in the search and search-and-replace commands; and the various options for the TRC command. An MLM command performs a call to the resident monitor, as opposed to the break entry caused by SYS4 or SYS 1024. A break disturbs the stack, while a call leaves it untouched. Finally, there is a FIX command, which resets POWER's features to their default conditions and restores any BASIC pointers that may have been disturbed.

Machine Language and Documentation

Many of POWER's features work just fine outside BASIC. For instance, you can set up an instant phrase such as "S,01,TEST,033A,035F" and a single keystroke will enter that after the monitor prompt. No mention is made of this in the manual, so this kind of POWER application is definitely at your own risk!

The documentation (by Jim Butterfield) is generally excellent. It explains the operation of most of the commands well, although it falls a little short in

PET VET *(Continued)*

explaining the XEC command. An example or two of XEC command operation would be a real help. Quite a bit of effort is made to point out quirks and to remind you of things you might overlook.

Where the manual really shines is in documenting the workings of POWER beyond the user's level. All RAM locations used are listed with their functions. Several internal machine-language routines, which can be used by the programmer, are documented. Also, the process of adding commands to POWER is explained, as is altering the entire command table.

What Doesn't It Do?

Other things you might expect of a utility package are: PRINT USING, a sound command, hex/decimal and decimal/hex conversion, screen dump, listing a sequential file, a merge that uses program rather than sequential files, and spooling of files from the disk to a printer. PRINT USING and sound commands require that the PET's command interpreter be intercepted during program operation. This slows down the operation of BASIC, and requires program is used. If you do require either the chip to be present whenever the of these, you can add them yourself as machine-language routines. The other commands can be added to POWER pretty easily as instant subroutines or whole new commands. In fact, they already have been. Next month we present POWER-Aid, a collection of routines designed to complement POWER.

Recommendations

Because of its flexibility, POWER stands above any other single utility package. You have control of how just about everything works. If you need a special function, and you can't do it with an instant phrase or subroutine, then you can add a new command.

At $89.95, POWER is an excellent value. However, if this seems too steep to you, perhaps the Programmer's Toolkit ($40) has enough of what you need. BASIC-Aid (available through most users' groups) is free, and it may offer enough to meet your needs if you can afford the loss of 10K of RAM. SYS-RES from Cansoft Data Inc. has many interesting features, and it will be reviewed in MICRO in the near future. It, too, is RAM-dependent.

POWER is available from Professional Software (51 Fremont Street, Needham, MA 02194) and from dealers. It was written by Brad Templeton to occupy 4K of ROM at $9000. There are three different versions: 4030 for upgrade 40-column, 4040 for 4.0 40-column and 8040 for 80-column..

Commodore to Introduce New Generation of Computers

At a recent show in Germany, Commodore exhibited prototypes of its new line of computers to be introduced on the market in the fall. The PET II is a 128K color computer that hooks up to a TV or monitor. The CBM II is a 256K computer that will include a black and white 80-column monitor and two 5¼" disk drives. An 'X' option will be offered which includes the 16-bit capability of an 8088 processor. In addition, a Z-80 option will be available to provide access to CP/M and other Z-80-dependent software.

At the heart of these new computers is the 6509, a new processor from MOS Technology. Its instruction set is identical to that of the 6502, but several other enhancements have been added. The addressing capability is expanded from the 6502's current 64K to 1 megabyte.

In last month's column the price for HESCAT was inadvertently listed as $23.95. It should be $39.95.

MICRO

ARK COMPUTING

LOWERS THE BOOM ON HIGH PRICES

SPECIALS

16K Ram Board	$79.95
80 Column Board (Wiz-80)	$195.00
Parallel Printer Interface (W/Cable) (compatible with Pascal, Basic, CP/M)	$59.95
Z-80 Softcard	$259.95
Joyport	$59.95
ARK Special (includes Wiz-80, Lazer Keyboard Plus, Lazer Lower Case Plus)	$295.00
On-Line Systems: General Manager	$99.95
Diskettes w/hubring (10)	$19.95
Hi-Res Secrets	$84.95
Expediter II	$59.95
The Dictionary	$69.95
Microsoft 16K Ram Card	$99.95
Time Zone	$69.95
Lazer Keyboard Plus	$69.95
Lazer Lower Case Plus	$44.95
Lazer Lower Case Plus II	$19.95
Anix 1.0	$34.95
Lazer Pascal	$29.95
Anix-Pac (Anix, Pascal, Sources)	$99.95
Using 6502 Assembly Language	$14.95
Datamost Joystick	$39.95
Datamost Expandaport	$49.95

GAMES

Softporn Adventure (no graphics)	$29.95/**$19.95**	Upper Reaches of Apshai	$19.95/**$15.95**
#0 Mission Asteroids	$19.95/**$14.95**	The Keys to Acheron	$19.95/**$15.95**
#1 Mystery House	$24.95/**$16.95**	Datestones of Ryn	$19.95/**$15.95**
#2 Wizard & Princess	$32.95/**$19.95**	Morloc's Tower	$19.95/**$15.95**
#3 Cranston Manor	$34.95/**$24.95**	Ricochet	$19.95/**$15.95**
#4 Ulyesses & the Golden Fleece	$34.95/**$24.95**	Starfleet Orion	$24.95/**19.95**
		Invasion Orion	$24.95/**$19.95**
#5 Time Zone	$99.95/**$69.95**	The Dragon's Eye	$24.95/**$19.95**
Hi-res Cribbage	$24.95/**$19.95**	Tues. Morning Quarterback	$29.95/**$21.95**
Hi-res Soccer	$29.95/**$19.95**	Jabbertalky	$29.95/**21.95**
Hi-res football	$39.95/**$29.95**	Sorcerer of Siva	$29.95/**$21.95**
Sabotage	$24.95/**$19.95**	Rescue at Rigel	$29.95/**$21.95**
Jawbreaker	$29.95/**$19.95**	Crush, Crumble, and Chomp	$29.95/**$21.95**
Threshold	$39.95/**$24.95**	Temple of Apshai	$39.95/**$29.95**
Missle Defense	$29.95/**$19.95**	Hellfire Warrior	$39.95/**$29.95**
Crossfire	$29.95/**$19.95**	Star Warrior	$39.95/**$29.95**
Pegasus	$29.95/**$19.95**	Hi-res golf	$29.95/**$21.95**
Warp Destroyer	$29.95/**$21.95**	Race for Midnight	$29.95/**$21.95**
Star Crusier	$24.95/**$19.95**	Midnight Music	$34.95/**$24.95**
Adventure	$29.95/**$21.95**	Star Blazer	$31.95/**$24.95**
Both Barrels	$24.95/**$19.95**	Apple Panic	$29.95/**$19.95**
Cyber Strike	$39.95/**$29.95**	Space Quark	$29.95/**$19.95**
Phantoms Five	$29.95/**$21.95**	Red Alert	$29.95/**$19.95**
Space Eggs	$29.95/**$19.95**	Track Attack	$29.95/**$19.95**
Autobahn	$29.95/**$21.95**	Genetic Drift	$29.95/**$19.95**
Pulsar II	$29.95/**$21.95**	Arcade Machine	$44.95/**$29.95**
Orbitron	$29.95/**$21.95**	Raster Blaster	$29.95/**$21.95**
Gamma Goblins	$29.95/**$19.95**	Trilogy of Games	$29.95/**$21.95**
Gorgon	$39.95/**$29.95**	Space Album	$39.95/**$29.95**
Sneakers	$29.95/**$19.95**	Fender Bender	$24.95/**$19.95**
EPOCH	$34.95/**$24.95**	3-D Graphics	$39.95/**$29.95**
Cops & Robbers	$34.95/**$24.95**	Akalabeth	$34.95/**$24.95**
Outpost	$29.95**$19.95**	Appleoids	$29.95/**$21.95**
Dark Forest	$29.95/**$19.95**	Ultima	$39.95/**$29.95**
Beer Run	$29.95/**$19.95**	LA Land Monopoly	$29.95/**$21.95**
Borg	$29.95/**$19.95**	Hyperspace Wars	$29.95/**$21.95**
Joy Port w/foosball	$74.95/**$59.95**	3-D Skiing	$24.95/**$19.95**
Hadron	$34.95/**$24.95**	Torpedo Terror	$24.95/**$19.95**
Twerps	$29.95/**$19.95**	Computer Bingo	$24.95/**$19.95**
Computer Foosball	$29.95/**$19.95**	Kaves Karkhan	$49.95/**$34.95**
Wizardry	$49.95/**$39.95**	Dragon Fire	$49.95/**$34.95**
Galactic Attack	$29.95/**$21.95**	Rings of Saturn	$39.95/**$29.95**
Minator	$34.95/**$24.95**	Alkemstone	$39.95/**$29.95**
Oylmpic Decathlon	$29.95/**$21.95**	Snack Attack	$29.95/**$19.95**
Three Mile Island	$39.95/**$29.95**	Casino	$39.95/**$24.95**
ABM	$24.95/**$19.95**	Thief	$29.95/**$19.95**
Robot Wars	$39.95/**$29.95**	County Fair	$29.95/**$19.95**
Global War	$24.95/**$19.95**	Swashbuckler	$34.95/**$24.95**
Castle Wolfenstein	$29.95/**$21.95**	Firebird	$29.95/**$19.95**
Falcons	$29.95/**$21.95**	Russki Duck	$34.95/**$25.95**
Suicide	$29.95/**$21.95**	Horizon V	$34.95/**$25.95**
Grand Prix	$29.95/**$21.95**	Sargon II	$34.95/**$25.95**
The Best of Muse	$39.95/**$29.95**	Reversal	$34.95/**$25.95**
Flight Simulator	$33.50/**$26.95**	Zork	
Dungeon Campaign	$17.50/**$14.95**	Zork II	$39.95/**$29.95**
Odyssey	$29.95/**$21.95**	Pool 1.5	$34.95/**$24.95**
Escape From Arcturus	$24.95/**$19.95**	Shuffleboard	$29.95/**$21.95**
Palace in Thunderland	$24.95/**$19.95**	Trick Shot	$39.95/**$29.95**
MAD Venture	$24.95/**$19.95**	Crossword Magic	$49.95/**$39.95**
Roach Motel	$34.95/**$26.95**	Master Type (hi-res)	$39.95/**$29.95**
English SAT #1	$29.95/**$21.95**	Dogfight	$29.95/**$21.95**
U.S. Constitution	$29.95/**$21.95**	Crown of Arthain	$34.95/**$26.95**

HARDWARE

12" hi-res green display	$285.00/**$159.95**
12" lo-res color display	$430.00/**$349.95**
12" hi-res RGB color display	$1095.00/**$895.95**
Apple Cat II	$389.00/**$329.00**
Apple Clock	$280.00/**$195.95**
Supertalker SD2000	$199.00/**$149.95**
Romplus	$155.00/**$124.95**
Romwriter	$175.00/**$139.95**
Music System	$395.00/**$295.95**
A/D+D/A	$350.00/**$249.95**
Expansion Chassis (125v)	$750.00/**$495.00**
CPS Multifunction Card	$239.00/**$149.95**
RAM Plus+ (32 RAM board w/16K installed)	$189.00/**$139.95**
COPYROM	$55.00/**$44.95**
Keyboard Filter ROM	$55.00/**$44.95**
13 Key Keypad (new or old)	$125.00/**$94.95**
Bar code reader	$195.00/**$145.95**
Softkey (15-key user-defin.)	$150.00/**$119.95**
Asynchronous Serial	$179.95/**$129.95**
Datamost Joystick	$59.95/**$39.95**
Datamost Expandaport	$69.95/**$49.95**
Micromodem	$379.00/**$269.00**
Smartmodem (RS-232)	$279.00/**$199.00**
Lower Case +(Rv 6 & earlier)	$64.95/**$44.95**
Lower Case + II (Rv. 7 & Ltr)	$24.95/**$14.95**
Lower Case+ III (Rv. 7 & Ltr)	$49.95/**$34.95**
Keyboard +Plus	$99.95/**$69.95**
Character Set+ Plus	$24.95/**$19.95**
Sup 'R' Term 80-col. board	$395.00/**$295.00**
Sup 'R' Switcher(pwr suppl)	$295.00/**$195.00**
A35 35-track disk drive w/o controller	$479.00/**$379.00**
A35 35-track disk drive w/ controller	$579.00/**$449.00**
A40 40-track disk drive w/o controller	$449.00/**$359.00**
A40 40-track disk drive w/ controller	$549.00/**$449.00**
A70 70-track disk drive w/o controller	$599.00/**$499.00**
A70 70-track disk drive w/ controller	$699.00/**$599.00**

BUSINESS SOFTWARE

Mail Label & Filing System	$74.95/**$49.95**
Payroll	$395.00/**$295.00**
General Ledger(w/payables)	$495.00/**$395.00**
Home Money Minder	$34.95/**$24.95**
The Mail Room	$34.95/**$24.95**
CPA I General Ledger	$249.95/**$169.95**
CPA II Accounts Receivable	$249.95/**$169.95**
CPA III Accounts Payable	$249.95/**$169.95**
CPA IV Payroll	$249.95/**$169.95**
CPA V Property Manage	$249.95/**$169.95**
The Home Accountant	$74.95/**$54.95**
Word Star	$375.00/**$195.00**
Supersort	$200.00/**124.95**
Mail Merger	$125.00/**$79.95**
Data Star	$300.00/**$195.00**
Spellstar	$200.00/**$124.95**
Calc star	$200.00/**$124.95**
Visicalc 3.3	$250.00/**$179.95**
PFS (Personal Filing System)	$125.00/**$89.95**
PFS Report	$95.00/**$69.95**
The Correspondent	$59.95/**$49.95**
Mailing List Database	$49.95/**$39.95**
Higher Text II	$39.95/**$29.95**
Directory Manager v.2	$29.95/**$21.95**
Real Estate Analysis Pgm	**30% OFF**
PIE WRITER	**(call for details)**
Easy Mover 40-column	$49.95/**$39.95**
Easy Mailer 40-column	$69.96/**$49.95**
Easy Writer 40-column	$99.95/**$74.95**
Pro Easy Mailer	$175.00/**$124.95**
Pro Easy Writer	$249.95/**179.95**
Super Scribe II	$129.95/**$79.95**

UTILITIES

Hi-Res Secrets (D. Fudge)	**$124.95/$84.95**
Super Shape Drg & Animate	$34.95/**$24.95**
The Creative Tool Box	$44.95/**$34.95**
Applesoft Compiler	$175.00/**129.95**
Datadex	$150.00/**$99.95**
Using 6502 Assembly Lang	$19.95/**$14.95**
ANIX (UNIX-like Oper. sys.)	$49.95/**$39.95**
Lazer Pascal	$39.95/**$29.95**
DOSOURCE 3.3	$39.95/**$24.95**
Painter Power	$39.95/**$29.95**
The Voice	$39.95/**$29.95**
E-Z Draw	$49.95/**$39.95**
Pascal Graphics Editor	$99.95/**$69.95**
ACE	$39.95/**$29.95**
Apple-DOC	$49.95/**$39.95**
List Master	$39.95/**$29.95**
ASCII Express	$79.95/**$59.95**
Z-Term (req's Z-80 Card)	$99.95/**$69.95**
Z-Term Pro	$149.95/**$99.95**
Speed Star	$134.95/**$99.95**
On-Line	$89.95/**$69.95**
DB Master	$229.00/**$159.95**

UTILITIES

LISA v2.5	$79.95/**$59.95**
LISA Educational Pack	$119.95/**$79.95**
Speed/ASM	$39.95/**$29.95**
Expediter II	$99.95/**$59.95**
Disk Organizer II	$29.95/**$21.95**
Applesoft Plus	$24.95/**$19.95**
Applesoft Optimizer	$24.95/**$19.95**
Disk Recovery	$29.95/**$21.95**
Multi-disk Catalog	$24.95/**$19.95**
Back it up	$59.95/**$39.95**
Image Printer	$29.95/**$29.95**
Pascal Lower Case	$24.95/**$19.95**
Dos Plus	$24.95/**$19.95**
The Original Quick Loader	$24.95/**$19.95**
Typing Tutor	$24.95/**$19.95**
Fortran	$195.00/**$139.95**
A.L.D.S.	$125.00/**$89.95**
Basic Compiler	$395.00/**$295.00**
Cobol	$750.00/**$495.00**
TASC	$175.00/**$129.95**
Data Plot	$59.95/**$44.95**
DB Utility Pack	$99.00/**$69.95**

We accept: MASTERCARD, VISA (Include card # and expiration date), CASHIER or CERTIFIED CHECKS, MONEY ORDERS, or PERSONAL CHECKS (Please allow 10 days to clear).
Please add 3% for shipping & handling (minimum $2.00). Foreign orders please add 10% for shipping & handling.
We accept COD's (Please Include $2.00 COD charge). California residents add 6% sales tax. All equipment is subject to price change and availability without notice. All equipment is new & complete with manufacturer's warranty

Apple is a registered trademark of APPLE COMPUTER INC.

YOUR SALVATION IN THE SEA OF INFLATION

ARK COMPUTING

P.O. BOX 2025
CORONA, CALIFORNIA 91720
(714) 735-2250

EXCEL-9

The Ultimate 6809 Board for Apple

from Esdec Corporation, a subsidiary of ESD LAB Co. Ltd.

- EXCEL-9 FLEX, a famous DOS, Assembler and Editor included.
- Also able to use Apple DOS.
- 8KB versatile monitor contains 35 commands including 6809.
- Can handle all Apple slot I/O routine from EXCEL-9.
- On-board programmable timer for both 6809 and 6502 systems allows printer spooling, multitask, etc.
- 50 page well documented manual.
- 64K RAM area expandable for multi-MPU operation.
- Able to switch MPU from 6809 to 6502 and vice versa in both machine code routine and BASIC.
- TSC 6809 BASIC, EXTENDED BASIC, PRECOMPILER, SOFT/MERG, etc., are coming soon.

Ask your nearest dealer or

Norell Data Syst. Corp.
3400 Wilshire Blvd.
P.O. Box 70127
Los Angeles, CA 90010
(213) 257-2026
United States Distributor

Dealer Inquiries are Invited.

Introductory Price: **$450.00** for Board & FLEX diskette
(Sales tax not included)

- FLEX is a trade mark of Technical Systems Consultants, Inc.

GRAPHTRIX™ 1.3

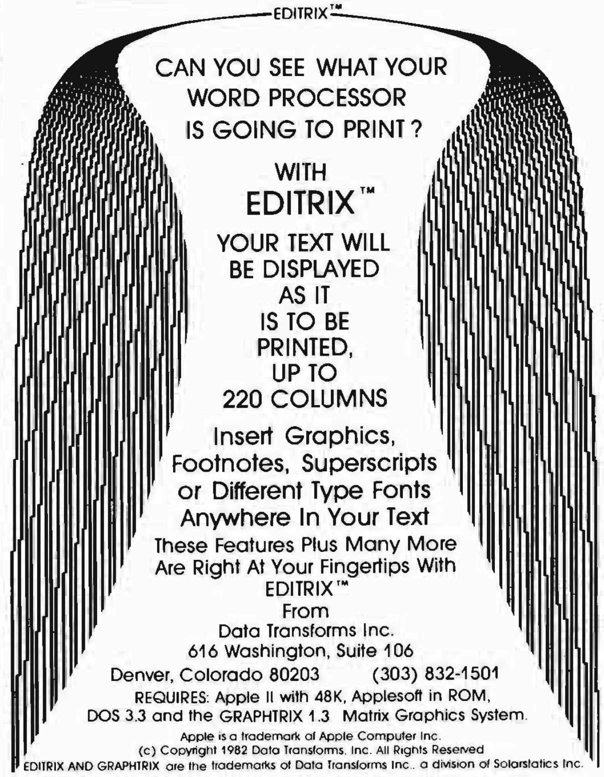
NEED
HARD COPY OF YOUR APPLE II
HI-RES GRAPHIC?
WITH
GRAPHTRIX™ 1.3
YOU CAN
INSERT YOUR
GRAPHIC
ANYWHERE
IN YOUR
TEXT.
USE ANY OF
19 PRINTERS
AND
10 INTERFACE CARDS.

From
Data Transforms Inc.
616 Washington, Suite 106
Denver, Colorado 80203
(303) 832-1501

Features: Graphic Magnification, Normal/Inverse, Page Centering, High and Low Crop Marks, Title String, Superscript, Footnotes, Chapters, Fully Menu Driven

REQUIRES: Apple II with 48K, Applesoft in ROM, One Disk Drive with DOS 3.3.

Apple is a Trademark of Apple Computer Inc.
© Copyright 1982 Data Transforms Inc All Rights Reserved.
GRAPHTRIX is the Trademark of Data Transforms Inc., a division of Solarstatics Inc



EDITRIX™

CAN YOU SEE WHAT YOUR
WORD PROCESSOR
IS GOING TO PRINT?
WITH
EDITRIX™
YOUR TEXT WILL
BE DISPLAYED
AS IT
IS TO BE
PRINTED,
UP TO
220 COLUMNS

Insert Graphics, Footnotes, Superscripts or Different Type Fonts Anywhere In Your Text

These Features Plus Many More Are Right At Your Fingertips With EDITRIX™

From
Data Transforms Inc.
616 Washington, Suite 106
Denver, Colorado 80203 (303) 832-1501

REQUIRES: Apple II with 48K, Applesoft in ROM, DOS 3.3 and the GRAPHTRIX 1.3 Matrix Graphics System.

Apple is a trademark of Apple Computer Inc.
(c) Copyright 1982 Data Transforms, Inc. All Rights Reserved
EDITRIX AND GRAPHTRIX are the trademarks of Data Transforms Inc., a division of Solarstatics Inc.

Reviews in Brief

Product Name:	**Line Printer VIII**
Equip. req'd:	Centronics or RS-232 standard computer or terminal
Price:	$799.99
Manufacturer:	Tandy Radio Shack Fort Worth, TX 76102

Description: Dot matrix printer with high density 9-wire print head. Logic seeking, unidirectional head movement. Print speed is 40 to 100 CPS in print modes. 480 dot per second in graphic mode. Modes include data processing, word processing, and bit graphics. There are six character fonts, including 5, 8.3, 10, 16.7 cpi and 2 proportional fonts. Features include partial and reversible line feeds, thus allowing super- and subscripts. A 9 × n mode allows the space between characters to be altered. By introducing the proper commands between words or letters, right justification of text is supported. A simple command causes text to be underlined until another command is received. All fonts can be accessed at any time within the line. Paper feeds include friction, pin, or single sheet. Another feature supported is a backspace command. Backspaces, or any character, may be repeated with a single command. Switch selection of European or Katakana is allowed. Controls include power, On-line/Off-line and restart/reset switch.

Pluses: The standard Centronics parallel interface, as well as the built-in 600 or 1200 baud RS-232 interface allows connection to many different models and brands of computers, or terminals. Fonts and modes may be changed at any time, even in mid-line by software control. Font capacity is increased by combining ordinary, condensed, and elongated modes. The range of characters per line varies from 40 to 132. A visual indicator warns of paper feed problems, and the front panel restart switch allows easy recovery from those kinds of interruptions. When the problem is repaired, pressing the restart switch will allow the printer to continue from where it left off with no loss of data. The two proportional fonts produce excellent quality characters, with serifs.

Minuses: Our sample has a problem with the ribbon hanging up on the perforations between sheets of fanfold paper. Occasionally, when this happens, the ribbon comes off the print head, and must be rethreaded. Increasing tension on the paper leaving the printer will usually alleviate the problem. A permanent solution was found in our case by gluing a small washer to the top of the print head. The washer hangs over the top edge, keeping the ribbon in place. Other samples of the machine were not available to check to see if this is a common occurrence or a flaw in this particular machine. Though not a problem with the printer itself, at the present time, there is little software available that supports its advanced features.

Documentation: Though thorough, it is not an instruction manual as much as a reference manual. Printer set up, operation, and programming information are included, but be prepared for heavy reading.

Skill level required: Hook-up and use of the printer is easy, and all commonly supported features are available for commercial programs. Writing your own software to operate the printer in its proportional modes, however, would take a high degree of programming skill.

Reviewer: John Steiner

Product Name:	**TEXTPRO**
Equip. req'd:	TRS-80C Color Computer with matching printer; empty ROMpack for EPROM version.
Price:	$39.95 - cassette tape $59.95 - EPROM
Manufacturer:	CER-COMP 5566 Ricochet Ave. Las Vegas, NV

Description: *TEXTPRO* is a combination text editor and text processor. The editor portion allows nearly a full range of editing capability, including tape file handling and the ability to concatenate tape files. Lines can be deleted, renumbered, and added using line numbers; numbers can be removed to save space or added to files recorded from other editors. String search, move, replace, and copy modes are available. The edit process has non-destructive cursor control, insert and delete, and forward and reverse scrolling capability.

Once the text has been entered and edited, it can be LISTed with or without line numbers, or the printing can be turned over to the text processor. Text processing consists of over 30 commands that imbed printer control function commands within the text.

Pluses: The package has a great capability for extremely low cost; either version requires 16K of memory, but the ROM version is a practical necessity if one intends to do much writing on a 16K machine, since the program is about 6K long. Even though the total cost for the ROM is high, it is worth the extra convenience.

Minuses: Lack of non-destructive cursor and insert mode during text entry, lack of horizontal scrolling to solve the 32-character Color Computer display problem, and inability to handle hyphenation in either mode.

Documentation: There is a lot of documentation, considering the product's price, but some of it is incomplete or confusing, without examples of how to use certain commands. A sample edit session is furnished, but it is far too brief and does not illustrate the more difficult commands. In spite of these difficulties, this package is a bargain, even at the price for the ROM.

Skill level required: One must be an experienced writer who really needs a text processor before he can appreciate this package and its capabilities.

Reviewer: Ralph Tenny

Product Name:	**Universal Boot Initializer**
Equip. req'd:	48K Apple II or Apple II Plus with 16K memory card or ROM card
Price:	$49.95 plus $3.00 postage and handling ($15.00 for back-up disk)
Manufacturer:	S&H Software Box 5 Manvel, ND 58256
Author:	Art Schumer
Copy Protection:	Yes
Language:	6502 Machine Language

Description: A utility for modifying both 13- and 16-sectored disks to permit booting with either 13- or 16-sectored disk controller PROMs. Includes a quick loading feature to rapidly install the "other" BASIC in a memory card. Permits greeting programs of binary and text files as well as the more usual BASIC types.

Pluses: For software developers and club librarians faced with serving Apple owners of both 13- and 16-sectored controller PROMs, this utility will be quite valuable. For memory card owners, the quickloading of the "other" BASIC is a time saving feature. The manual is carefully written, following a "training" program which is included with the package. After working with both for a few minutes, proper use of the program is virtually assured. Added features include: the ability to use binary and text files as greeting programs as an alternative to the usual BASIC program; the use of Directory Title Formatting for more carefully described CATALOG listings and the facility to report an error message in the event that the required BASIC (Integer or Applesoft) is not present.

Minuses: The user must have both BASICs available *via* 16K memory card or ROM card. To permit 16-sectored users to boot DOS 3.2.1 disks, a copy of UPDATE 16, written by Steve Wozniak is included. Use of UPDATE 16 prohibits the standard COPY program from reading the modified track on the 13-sectored disk. Such 13-sectored disks must be copied with other procedures. Individuals purchasing this utility for the quick loading of the other BASIC must be aware that disks initialized after such a boot will look for the same file type greetings program as was on the boot disk. In the reviewer's opinion, the cost of the package is notably high.

Skill level required: The carefully written manual and training program can be used by almost any Apple owner.

Reviewer: David R. Morganstein

Product Name:	**CMEMORY**
Equip. req'd:	TRS-80C Color Computer
Price:	$24.95
Manufacturer:	MICRO-LABS, Inc. 902 Pinecrest Dr. Richardson, TX 75080

Description: A molded plug-in module for the Color Computer with a removable PC board; it has sockets for four 2716 EPROMs or 2716-pinout CMOS or CMOS read-write memory devices. The part is well-manufactured and fits properly in the port.

Pluses: Allows convenient packaging of any mix of 2716 and 6116-type memory devices, thus facilitating special-purpose program plug-ins for instant change of programming. In addition, it is possible to piggy-back four additional 6116 read-write devices on the four installed in the module, thus giving almost 16K of additional memory for the computer. Very reasonable price.

Minuses: None noted.

Documentation: Adequate explanation is given to use and/or modify this product for any of the intended purposes.

Skill level required: None, unless the owner supplies program for the cartridge or uses the piggy-back method to expand to 16K. In the case of user programs, it is necessary to understand the techniques used to adapt programs to run in ROM at specific addresses. In the case of cartridge modification, minimal soldering and assembly skills are required.

Reviewer: Ralph Tenny

Reviews in Brief *(continued)*

Product Name:	**Key Perfect**
Equip. req'd:	Apple II or Apple II Plus with 48K RAM, 1 Disk II
Price:	$29.95
Manufacturer:	micro-sparc, inc. P.O. Box 639 Lincoln, MA 01773

Description: Utility which computes "check codes" associated with Apple II program files. This allows the user who keys in a program from a published listing to compare published values of check codes with those obtained from the keyed-in version. The aim is to facilitate the location of keying errors.

Pluses: Seems to work.

Minuses: This is really the kind of program that should be published and given away, rather than sold. That is, I feel it is overpriced! The program is not particularly user-forgiving. For example, to switch to or from *Key Perfect* you must reboot each time. The program does not allow a CATALOG command to be issued while it is running — so don't forget your program name! The program should prompt for what DOS is on the check disk; instead it deduces this information thus slowing things down. The program incessantly prompts you to make sure you haven't made a mistake in specifying information. While this is okay for novices, it should be optional for experts.

Documentation: Adequate. It would be interesting and of educational value to include a discussion of the algorithms used to compute the check codes.

Skill level required: Ability to follow directions.

Reviewer: Richard C. Vile, Jr.

Product Name:	**Amper-Sort/Merge**
Equip. req'd:	48K Apple with Applesoft and DOS 3.3 data files
Price:	$49.95 plus $3.00 postage and handling ($15.00 for back-up disk)
Manufacturer:	S&H Software Box 5 Manvel, ND 58256
Author:	Alan Hill
Copy Protection:	Yes
Language:	6502 Machine Language

Description: A utility package for sorting of sequential or random access text files. Can sort and merge up to five user-supplied file names at machine-language speeds. Can sort on as many as five user-specified keys, each key selected for ascending or descending order.

Pluses: Program is user-friendly with ample prompts. Uses work files for multiple merging operation under program control, thus allowing the user to sort large text files. Twenty-four-page manual is well written and clear and includes some history of the development of the Amper-sort package. The author has published many articles on sorting and related topics, and has provided in those articles source code for the original version. The package will sort Visifile data bases, given the use of a short BASIC program provided in the manual. The user specifies the slot and drive of the input files, of a disk with space for work files required to perform the sort, and of the disk to contain the sorted data. The sorting parameters (file names, fields for sorting, slot and drive numbers) can be saved in a file for later use. The program uses fast garbage collection routines to increase speed and special file reading routines for faster disk access of arrays.

Minuses: The package has few faults. The reviewer found a screen prompt regarding slot and drives specified to be in error. When a slot and drive for the work file was specified to be other than the default value, the prompt incorrectly referred to the default with a statement like "insert workfile in slot 6, drive 1" when drive 2 was the selected one. The program correctly read from drive 2 and sorted properly, however. This fairly minor problem should be easy to correct. Unlike the articles published by Alan Hill, the program is on a protected disk; it cannot be easily incorporated in a user's program.

Skill level required: The program must be handled as a utility by a fairly knowledgeable user who understands the file names and formats of the data to be sorted. The user needs no programming knowledge, however.

Reviewer: David R. Morganstein

MICRO

HDE inc.

BOX 120
ALLAMUCHY, N.J. 07820
201-362-6574

HUDSON DIGITAL ELECTRONICS INC.

THE TASK* MASTERS

HDE supports the *TIM, AIM, SYM and KIM (TASK) with a growing line of computer programs and peripheral components. All HDE component boards are state-of-the-art 4½" x 6½", with on board regulation of all required voltages, fully compatible with the KIM-4 bus.

OMNIDISK 65/8 and 65/5
Single and dual drive 8" and 5¼" disk systems. Complete, ready to plug in, bootstrap and run. Include HDE's proprietary operating system, FODS (File Oriented Disk System).

DM816-M8A
An 8K static RAM board tested for a minimum of 100 hours and warranted for a full 6 months.

DM816-UB1
A prototyping card with on-board 5V regulator and address selection. You add the application.

DM816-P8
A 4/8K EPROM card for 2708 or 2716 circuits. On board regulation of all required voltages. Supplied without EPROMS.

DM816-CC15
A 15 position motherboard mounted in a 19" RETMA standard card cage, with power supply. KIM, AIM and SYM versions.

DISK PROGRAM LIBRARY
Offers exchange of user contributed routines and programs for HDE Disk Systems. Contact Progressive Computer Software, Inc. for details.

HDE DISK BASIC
A full range disk BASIC for KIM based systems. Includes PRINT USING, IF . . . THEN . . . ELSE. Sequential and random file access and much more. $175.00

HDE ADVANCED INTERACTIVE DISASSEMBLER (AID)
Two pass disassembler assigns labels and constructs source files for any object program. Saves multiple files to disk. TIM, AIM, SYM, KIM versions. $95.00

HDE ASSEMBLER
Advanced, two pass assembler with standard mnemonics. KIM, TIM, SYM and KIM cassette versions. $75.00 ($80.00 cassette)

HDE TEXT OUTPUT PROCESSING SYSTEM (TOPS)
A comprehensive text processor with over 30 commands to format and output letters, documents, manuscripts. KIM, TIM and KIM cassette versions. $135.00 ($142.50 cassette)

HDE DYNAMIC DEBUGGING TOOL (DDT)
Built in assembler/disassembler with program controlled single step and dynamic breakpoint entry/deletion. TIM, AIM, SYM, KIM AND KIM cassette versions. $65.00 ($68.50 cassette)

HDE COMPREHENSIVE MEMORY TEST (CMT)
Eight separate diagnostic routines for both static and dynamic memory. TIM, AIM, SYM, KIM and KIM cassette versions. $65.00 ($68.50 cassette)

AVAILABLE DIRECT OR FROM THESE FINE DEALERS:

Progressive Computer Software
405 Corbin Road
York, PA 17403
(717) 845-4954

Johnson computers
Box 523
Medina, Ohio 44256
(216) 725-4560

Falk-Baker Associates
382 Franklin Avenue
Nutley, NJ 07110
(201) 661-2430

Perry Peripherals
P.O. Box 924
Miller Place, NY 11764
(516) 744-6462

Lux Associates
20 Sunland Drive
Chico, CA 95926
(916) 343-5033

Laboratory Microcomputer Consultants
P.O. Box 84
East Amherst, NY 14051
(716) 689-7344

Computer-Assisted Translation of Programs from 6502 to 6809

by Edgar Pass

The article discusses techniques of translating 6502 programs to run on a 6809-based machine. Tables, 6809 routines, and discussion of special problems are included.

Initial Comparison

From a review of the Motorola 6800 and 6809, and MOS 6502, the instruction sets of the 6809 and 6502 are both seen to be derivatives of the (older) 6800 instruction set. However, the extensions and changes made in the 6809 and 6502 instruction sets have been in quite different directions. Table 1 presents the programming models for each of the processors, to indicate the flavor of some of the changes and extensions.

Register Comparison

The similarities and differences in the register structures of the processors are apparent in table 1. Of the three processors, the 6809 has the most versatile register structure with its two 8-bit accumulators, 8-bit direct page register, two 16-bit index registers, and two 16-bit stack pointers. The 6502 has a less versatile register structure than either of the other two processors, its only highlight being a second 8-bit index register. The relative speed of the processors or relative compactness of the code are not issues here.

When matching up the register structures from the 6502 to the 6809, most registers map to the similarly named register. The exception is the 6502 A register, which corresponds more closely to the 6809 B register than the A register because of the manner in which the 6809 TFR and EXG instructions function.

The condition code registers of the three processors all differ in format and content, with the 6800 and 6809 being the most similar and the 6502 the most

Table 1: **Programming Models for the 6800, 6809, and 6502**

Register	Bits	Description
		6800
A	8	Accumulator
B	8	Accumulator
CC	8	Condition Code Register (11HINZVC)
PC	16	Program Counter
S	16	Stack Pointer
X	16	Index Register
		6809
A	8	Accumulator
B	8	Accumulator
CC	8	Condition Code Register (EFHINZVC)
D	16	A and B Registers (Concatenated)
DP	8	Direct Page Register
PC	16	Program Counter
S	16	Stack Pointer
U	16	User Stack Pointer
X	16	Index Register
Y	16	Index Register
		6502
A	8	Accumulator
CC	8	Condition Code Register (NV0BDIZC)
PC	16	Program Counter
S	8	Stack Pointer (First 8 bits = 01)
X	8	Index Register
Y	8	Index Register

where Condition Code Register bits are defined as follows:

B	BRK command (6502)
C	carry/borrow
D	decimal mode (6502)
E	entire state on stack (6809)
F	fast interrupt (6809)
H	half carry (6800/6809)
I	interrupt mask
N	negative
V	overflow
Z	zero

NEW LOW PRICE
DIRECT FROM MANUFACTURER
$120.00

16K RAM EXPANSION BOARD FOR THE APPLE II* $120.00

The Andromeda 16K RAM Expansion Board allows your Apple to use RAM memory in place of the BASIC Language ROMs giving you up to 64K of programmable memory. Separate Applesoft* or Integer BASIC ROM cards are no longer needed. The 16K RAM Expansion Board works with the Microsoft Z-80 card, Visicalc, DOS 3-3, Pascal, Fortran, Pilot, and other software. A switch on the card selects either the RAM language or the mainboard ROMs when you reset your Apple.

The Andromeda 16K RAM Expansion Board has a proven record for reliability with thousands of satisfied customers.

Now with One Year Warranty.

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

VISA
master charge
THE INTERBANK CARD

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

*DEALER INQUIRIES WELCOME.

unlike. All three condition code registers contain carry/borrow, interrupt mask, negative, overflow, and zero bits, although the interpretation and setting of bits may vary considerably among the three.

The 6502 "V" flag is modified by far fewer instructions than the "V" flags on the 6800 and 6809 processors. The 6502 "B" flag allows an interrupt processing routine to determine the difference between an external interrupt and an internal interrupt generated by a BRK command. The 6502 "D" flag determines whether the ADC and SBC commands will operate in decimal or binary mode. There are no directly corresponding flags for "B" and "D" on the 6800 or 6809 processors. The (nearly) equivalent functions are performed in quite different ways.

The addressing modes supported by each of the processors are generally similar, although there are a few significant differences. Table 2 presents the addressing modes of interest in each of the processors of interest.

One significant difference between the 6502 and the other two processors lies in the storage format of a 16-bit address. Whereas the Motorola processors store 16-bit addresses as high-order 8-bits, then low-order 8-bits in successive locations, the 6502 stores 16-bit addresses as low order 8-bits, then high-order 8-bits in successive locations. This difference appears in the format of instructions containing 16-bit addresses and offsets, return addresses in the stack, 16-bit indirect addresses, interrupt vectors, jump tables, etc.

There are several differences in the use of the S registers on the 6502, 6800, and 6809. The most obvious is that the 6800 and 6809 use a 16-bit S register, whereas the 6502 uses an 8-bit S register and prefixes these 8-bits with an 8-bit constant 01 to form a 16-bit address. Thus the 6502 stack is restricted to addresses $0100-$01FF. The 6800 and 6502 decrement the stack pointer after placing a new item into it, whereas the 6809 decrements it before. Thus the 6800 and 6502 stack pointers always point to one address below the current stack limit, whereas the 6809 stack pointer always points to the last item placed onto the stack (if any). The TSX and TXS instructions on the 6800 (but not on the 6502) take this into account by adding one to the X register after transferring the contents of the the S register to it and by subtracting one from the S register after transferring the X register to it.

This difference can cause a problem when you translate programs from the 6800 to the 6809. However, because of the highly restricted nature of the 6502 S register, it should cause little difficulty in translating programs from the 6502 to the 6809. The main problem stems from the 6800 trick of using the stack pointer as a second index register. However, the 6502 Y register functions as a second index register in many addressing modes, and the 6502 S register is restricted to page 01 in memory addresses, eliminating it as an effective third index register on the 6502.

Table 3 summarizes many of the differences and similarities already discussed concerning the 6502, 6800, and 6809, in terms of the 6502 instruction set. This set has 56 members, as opposed to 97 members for the 6800 and 58 members for the 6809. However, counting address mode and register variations, the 6502 can execute approximately 100 instructions, the 6800 can execute approximately 200 instructions, and the 6809 can execute approximately 750 instructions. Complete instruction sets for each of the 6502, 6800, and 6809 processors may be

***Table 2:* Addressing Modes**

Mode	Description
Inherent (Accumulator, Implied)	Changes registers or processor states without explicit regard for memory addressing
Direct (Zero-Page)	Prefixes 8-bit address in instruction with 8-bit 00 (DP on 6809) to provide 16-bit effective address
Extended (Absolute)	Uses 16-bit address in instruction directly as effective address
Immediate	Uses 8-bit or 16-bit value in instruction directly, and not as a memory address
Relative	Adds 8-bit offset in instruction to address of next sequential instruction to provide effective address of next instruction to be executed
Indexed (6800)	Adds 8-bit offset in instruction to value in X register to provide 16-bit effective address
Indexed (6809)	Uses one or more post-byte values in instruction to indicate an entire range of register and direct, indirect, or non-indirect addressing schemes
Zero Page Indexed (6502)	Adds 8-bit offset in instruction to value in X or Y register to compute 8-bit value; prefixed this value with 8-bit 00 to provide 16-bit effective address
Absolute Indexed (6502)	Adds 16-bit offset in instruction to value in X or Y register to provide a 16-bit effective address
Indirect (6502)	Uses the 16-bit address in instruction to provide a 16-bit effective address; uses the contents of the locations at that address and at the next address to provide a 16-bit memory address
Indexed Indirect (6502)	Adds the 8-bit offset in instruction to value in X or Y register to provide an 8-bit value, which is prefixed by an 8-bit 00 to form a 16-bit effective address; the locations at that address and at the next address to provide a 16-bit effective address
Indirect Indexed (6502)	Prefixes 8-bit address in instruction with 8-bit 00 to provide a 16-bit effective address; uses the contents of the locations at that address and at the next address to provide a 16-bit effective address

found at the end of this article. An asterisk in table 3 indicates that the instruction has the indicated address mode. An entry under Condition-Code-Reg Form indicates the conversion of the Condition-Code format. An entry under Stack indicates stack manipulation, and an entry under X/Y indicates X or Y register modification. The entries under 6809 Condition-Code-Reg indicate the results provided by the translation suggested later in this article.

Emulation Discussion

The additional registers and instructions on the 6809 make possible an almost exact emulation of the 6502. The 6809 code will not generally have the same length as the 6502 code, nor will it require the same amount of time to execute. Because the translation is being done before assembler time, no run-time instruction modification is assumed.

Certain features of the two processors are similar but not identical. If the incremental cost of the exact emulation of a 6502 instruction or feature exceeds its incremental utility in a specific program or subroutine, it would be highly desirable to be able to trade off the exact emulation for a speed and space reduction in the 6809 code. For instance, the format and contents of the 6502 and 6809 condition code registers are different. Assuming that the "B" and "D" flags of the 6502 are handled separately, many 6502 programs would run correctly with no or minor changes (after translation) on the 6809, even with the 6809 format of condition code register.

The following differences in the processors' instruction sets cause time and space problems in the emulation process:

- reversed order of absolute address high and low bytes
- stack restriction to $01XX address range
- "B", "D", and "V" flag handling in many instructions
- format of condition code register
- page-zero wraparound in several addressing modes
- 8-bit X and Y register limitations

Other major tradeoffs will be discussed in relation to the individual instructions.

Table 3: **Summary Table**

6502 Opcode	Absolute/ Zero-Page	Condition-Code-Reg 6502			Stack	Zero Wrap	Indirect Wrap	X/Y
		NVØBDIZC	6809 EFHINZVC	Form				
ADC	*	NV....ZC	..H.NZVC			*	*	
AND	*	N.....Z.	....NZ..			*	*	
ASL	*	N.....ZC	....NZ.C			*		
BCC								
BCS								
BEQ								
BIT	*	NV....Z.	....NZV.					
BMI								
BNE								
BPL								
BRK		...1.1..	...1....		-3			
BVC								
BVS								
CLC		.......Ø	.......Ø					
CLD		....Ø...	RESET D					
CLI		.....Ø..	...Ø....					
CLV		.Ø......	......Ø.					
CMP	*	N.....ZC	....NZ.C			*	*	
CPX	*	N.....ZC	....NZ.C					
CPY	*	N.....ZC	....NZ.C					
DEC	*	N.....Z.	....NZ..			*		
DEX		N.....Z.	....NZ..					X
DEY		N.....Z.	....NZ..					Y
Opcode	Absolute/ Zero-Page	Condition-Code-Reg 6502 NVØBDIZC	6809 EFHINZVC	Form	Stack	Zero Wrap	Indirect Wrap	X/Y
EOR	*	N.....Z.	....NZ..			*	*	
INC	*	N.....Z.	....NZ..			*		
INX		N.....Z.	....NZ..					X
INY		N.....Z.	....NZ..					Y
JMP	*							
JSR	*				-2			
LDA	*	N.....Z.	....NZ..			*	*	
LDX	*	N.....Z.	....NZ..			*		X
LDY	*	N.....Z.	....NZ..			*		Y
LSR	*	Ø.....ZC	....ØZ.C			*		
NOP								
ORA	*	N.....Z.	....NZ..			*	*	
PHA					-1			
PHP				TO	-1			
PLA		N.....Z.	....NZ..		+1			
PLP		NVØBDIZC	EFHINZVC	FROM	+1			
ROL	*	N.....ZC	....NZVC			*		
ROR	*	N.....ZC	....NZ.C			*		
RTI		NVØBDIZC	EFHINZVC		+3			
RTS					+2			
SBC	*	NV....ZC	....NZVC			*	*	
SEC		1	1					
SED		....1...	SET D					
SEI		.....1..	...1....					
STA	*					*	*	
STX	*					*		X
STY	*					*		Y
TAX		N.....Z.	....NZ..					X
TAY		N.....Z.	....NZ..					Y
TSX		N.....Z.	....NZ..		Ø			X
TXA		N.....Z.	....NZ..					X
TXS		....	....		X+1			X
TYA		N.....Z.	....NZ..					Y

Reversed Address Bytes

To reverse the order of high and low address bytes on the 6809 from the 6502, several approaches are possible. The most direct method, which still maintains an exact emulation, is to assume that all extended address bytes, except within instructions, are reversed. You must include 6809 code of the following form to actively flip the address before use:

```
TFR CC,DP      Save CC Register
LDU address    Load Address
EXG U,D        Move Address
EXG A,B        Reverse Bytes
EXG D,U        Put Address in U
               Register
TFR DP,CC      Restore CC Register
```

Executing this code is time-consuming and wasteful if it is not needed. The definition of the 6502 .WORD (or · equivalent) assembler pseudo-op code will require defining in such a manner as to reverse the bytes of its address operands. The TFR instructions used above are included to avoid disturbing the condition code register; most such sections of code will require protection of the condition code register.

In many cases, the programmer may decide to use the 6809 rather than 6502 form of extended addressing, and modify the translated program as necessary to accomplish this. Then the reversal of address bytes as described above will not be required and the 6502 .WORD (or equivalent) assembler pseudo-op code will be translated to the 6809 FDB. The programmer will be required to correspondingly modify references to the bytes in the program representing reversed extended addresses. However, this tradeoff preserves more of the flavor of the 6809 and less of the 6502 and is hence more efficient.

The 6502 Stack Page Restriction

The 6502 stack restriction to the $01XX address range causes translation problems as far-reaching as the reversed address bytes situation. Every operation involving items placed onto the stack or pulled from the stack or the setting of the S register must be done through special inline code. The translator may not directly insert any operation, such as a subroutine call, which uses the stack. The 6502 S register always points to the next available location, whereas the 6809 S register always points to the last item pushed onto the stack. Whether 6502 stack emulation is used or not, the translated program must initialize the S register. Interrupt processing may not be supported with the emulated stack. The 6502 instructions which directly place information on the stack are as follows: BRK, JSR, PHA, PHP; those which directly gather information from the stack are: PLA, PLP, RTI, RTA; and those which directly use or modify the stack pointer are: TSX, TXS.

The inserted 6809 code to emulate the placing of an item onto a 6502 stack is of the following form:

```
STB ,S        Store B Register in
              Stack
TFR CC,DP     Save CC Register
TFR D,U       Save D Register
TFR S,D
DECB          Bump S Register Down
TFR D,S       Set S Register
TFR U,D       Restore D Register
TFR DP,CC     Restore CC Register
```

and that of removing of an item from a 6502 stack is of the following form:

```
TFR CC,DP     Save CC Register
TFR D,U       Save D Register
TFR S,D
INCB          Bump S Register Up
TFR D,S       Set S Register
TFR U,D       Restore D Register
LDB ,S        Get B Register from
              Stack
TFR DP,CC     Restore CC Register
```

Instructions (such as BRK, JSR, RTI, and RTS) that require multiple stack operations will require multiple copies of these stack push and pull operations for exact emulation. Even with the pull and push routines, exact 6502 stack emulation must be done with interrupts turned off. The 6502 TXS and TSX instructions will require review if either the S or the X register is assumed to be 16 bits long, as in the 6809 processor. Unless such exact stack emulation is required in a given situation (which it seldom is), most 6502 programs will run after translation using 6809 stack handling with little or no change, and with a great increase in efficiency and functionality for stack-related operations.

The B, D, and V Flags

The content differences in the condition code registers of the 6502 and 6809 are apparent primarily in the cases of interrupt processing and the ADC, BIT, BRK, CLD, CLV, PHP, PLP, RTI, SBC, and SED instructions.

The 6502 BRK instruction has no exact 6809 counterpart with respect to the "B" flag in the condition code register. However, if 6502 stack emulation and condition code register format are not required, the 6502 BRK instruction may be translated to the 6809 SWI instruction, which has a different vector address in high memory from the IRQ interrupt.

The 6809 has no direct counterpoint to the use of the 6502 "D" flag; however, it is modified only by the CLD, PLP, RTI, and SED instructions and is used only by the ADC and SBC instructions. Thus the 6502 "D" flag is easily emulated using a separate byte. The only difficulties are with the 6809 SBC instruction, which does not interface with the DAA instruction, and with properly separating and combining multiple "D" flag bytes during interrupt processing.

(Continued on page 84)

ATARI

ATARI 800

16K . . . $669
32K . . . $749
48K . . . $779

410 Recorder	$76.00
810 Disc Drive	$449.00
822 Printer	$289.00
825 Printer	$629.00
830 Modem	$159.00
820 Printer	$269.00
850 Interface	$169.00
New DOS 2 System	$29.00
CX30 Paddle	$18.00
CX40 Joy Stick	$18.00

CX853 16K RAM	$77.95
Microtek 16K RAM	$64.95
Microtek 32K RAM	$99.95
Ramdisk (128K)	$429.95
INTEC 32K Ram	$99.95
INTEC 48K Ram	$199.95

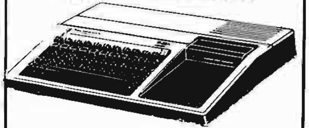
One year extended warranty	$70.00
481 Entertainer	$69.00
482 Educator	$130.00
483 Programmer	$49.00
484 Communicator	$344.00

ATARI 400

16K . . . $299
32K . . . $419
48K . . . $519

Visicalc	$179.00
Letterperfect	$109.00
Ricochet	$14.50
Crush Crumble & Chomp	$24.00
Star Warrior	$29.00
Rescue at Rigel	$24.00
Datestones	$16.00
Invasion Orion	$18.50
Mission Asteroid	$22.00
MouskATTACK	$31.00
The Next Step	$34.00
Softporn	$27.00
Wizzard & Princess	$29.00
K-BYTE Krazy Shoot Out (ROM)	$39.00
Protector (Disk 32K)	$24.00
Jaw Breaker (on line disk)	$27.00
Ghost Hunter (cassette)	$24.00
Ghost Hunter (disk)	$30.00
PAC MAN	$35.00
Centipede	$35.00
Caverns of Mars	$32.00
Synapse	
File Manager 800	$69.95
Dodge Racer	$19.00
Chicken	$24.00
Slime	$24.00
Nautilus	$24.00
Disk Manager	$24.00
Fort Apocalypse	$24.00
Assembler	$39.00

Texas Instruments



TI-99/4A $299

R.F. Modulator	$29.00
Telephone Coupler	$169.00
PHP1200 Peripheral Box	$199.00
PHP1220 RS232 card	$143.00
PHP1240 Disk Controller	$199.00
PHP1250 Disk Drive	$319.00
PHP1260 32K RAM	$229.00
Wired Remote Controllers(Pair)	$31.00
PHP Printer Solid State	$319.00
Home Financial Decisions	$26.00
Personal Record Keeping	$43.00
Mailing List	$60.00
Checkbook Manager	$18.00
Tombstone City 21st Century	$34.00
Munch Man	$34.00
TI Invaders	$34.00
Car Wars	$34.00

Computer Covers

ATTRACTIVE COVERS FOR YOU COMPUTER AND DISK DRIVE.

Atari 400	$6.99
Atari 800	$6.99
Atari 810	$6.99

All Atari Covers are Beige.

Commodore VIC-20	$6.99
Commodore 8032	$14.99
Commodore 8050/4040	$10.99

All Commodore Covers are Royal Blue.

Monitors

Amdek	
300G	$169.00
Color I	$339.00
Color II	$699.00
Color III	$429.00
NEC	
12" B&W	$169.00
12" Color	$339.00
TI 10" Color	$349.00
Zenith ZVM 121 (Green)	$119.00
BMC 12" Green	$85.00

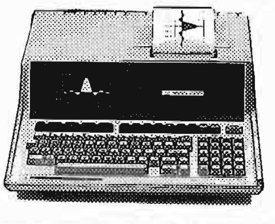
Televideo

910	$579.00
912C	$699.00
920C	$749.00
925C	$749.00
950	$939.00
802	$Call
802H	$Call
606	$Call
816	$Call

Modems

Hayes	
Smart	$239.00
Chronograph	$199.00
Micromodem II	$279.00
Micromodem 100	$309.00
Novation Auto	$239.00
D Cat	$169.00
Cat	$159.00

HEWLETT PACKARD



HP•85 $1899

HP • 86	call
HP•87	$1769.00
HP•125	$1999.00
HP•85 18K Memory Module	$169.00
5¼" Dual Master Disc Drive	$1769.00
Hard Disk w/Floppy	$4349.00
Hard Disk	$3440.00
80 Column Printer	$799.00
"Sweet Lips" Plotter	$1149.00
87 CP/M	$399.00
87 128K Memory	$610.00
87 Visicalc	$205.00
125 & 85 Visicalc	$179.00

HP41CV Calculator $239

41C	$189.00
11C	$104.00
12C	$114.00
34C	$114.00
38C	$114.00
HP•41 Printer	$340.00
HPIL CALCULATOR PERIPHERALS	
IL Modual	$104.00
Digital Cassette	$449.00
Printer/Plotter	$419.00
Card Reader	$164.00
Optical Wand	$99.00

Apple

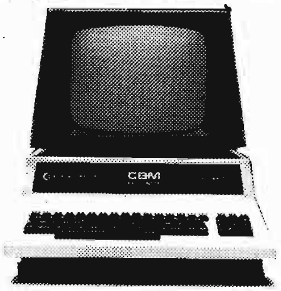
Call for availability and prices on all Apple computers and peripherals.

Printers

Smith Corona TP1	$699.00
Centronics 739-1	$519.00
Centronics 739-3	$619.00
Diablo 630 Special	$1799.00
Epson	
MX80 w/Graftrax	$449.00
MX80FT	Call
MX100	Call
NEC	
8023	$549.00
7730	Call
7720	Call
7710	Call
Okidata	
82A	$499.00
83A	$749.00
84	$1129.00
Citoh Starwriter	
F10-40 CPS	$1469.00
F10-55 CPS	Call
Prowriter	$479.00
Talley	
8024-L	$1829.00
IDS	
Prism	$Call
MPC Apple Parallel Board & Cable	$69.00
2 Meter RS232-RS232	$29.95

Cables Available For Most Interfacing Purposes

commodore



8032 $1039

4032	$969.00
4016	$769.00
8096 Upgrade Kit	$369.00
Super Pet	$1599.00
2031	$529.00
8250 (Double Sided D. Drive)	$1699.00
5 Megabyte Hard Disk	$2399.00
8050	$1299.00
4040	$969.00
8300 (Letter Quality)	$1799.00
8023	$769.00
4022	$599.00
Pet to IEEE Cable	$37.00
IEEE to IEEE Cable	$46.00
Tractor Feed for 8300	$240.00

SOFTWARE

Commodore	Magis
BPI	Professional Software
Visicorp	Creative Software

VIC 20 $239

Call for price and availability of VIC-64

Commodore Catassette	$69.00
Disk Drive	$499.00
VIC Graphic Printer	$339.00
3K Memory Expander	$32.00
8K Memory Expander	$53.00
16K VIC Expansion	$99.00
RS232C Terminal Interface	$43.00
VIC IEEE-488 Interface	$86.00
VIC 20 Super Expander	$53.00
Programmers Reference Guide	$15.00
Introduction to Computing	$19.00
Introduction to BASIC Programming	$19.00
Household Finance	$27.00
VIC Games	$19.00
VIC Home Inventory	$13.00
VIC Rec/Ed II	$13.00
Terminal	$13.00
Un Word	$13.00
Grafix Menagerie	$11.00
VIC PICS	$15.00
Ticker Tape	$13.00
Banner Headliner	$13.00
RS 232	$39.00
Super Slot	$23.00
VIC Avengers	$23.00
Super Alien	$23.00
Super Lander	$23.00
Draw Poker	$23.00
Midnite Drive	$23.00

computer mail order

east
800-233-8950
477 East Third Street
Williamsport, PA 17701
(717) 327-9575
Patricio Habla Espanol

CALL TOLL FREE

HOW TO ORDER:

In-stock items shipped same day you call. No risk, no deposit on C.O.D. orders. Pre-paid orders receive free shipping within the continental United States with no waiting period for certified checks or money orders. All prices shown are cash prices add 3% for Mastercard and Visa. NV and PA residents add sales tax. All items subject to availability and price change.

west
800-648-3311
P.O. Box 6689
Stateline, Nevada 89449
(702) 588-5654
Franco Habla Espanol

Table 4: Translation Analysis

6502 Opcode	6809 Code	Comments
ADC Operand	ADC Operand	Add with Carry
	TFR CC,DP	Save CC Register
	TFR CC,A	
	ANDA #$02	
	STA SEVFLG	Set V Flag Byte
	TST SEDFLG	Check D Flag
	BEQ • +7	
	TFR DP,CC	Restore CC Register
	DAA	Convert to Decimal
	BRA • +4	
	TFR DP,CC	Restore CC Register
AND Operand	AND Operand	AND Accumulator
ASL Operand	ASL Operand	Arithmetic Shift Left
BCC Operand	BCC Operand	Check C Flag
BCS Operand	BCS Operand	Check C Flag
BEQ Operand	BEQ Operand	Check Z Flag
BIT Operand	ANDA Operand	Bit Test
	• N and V Flags Not Set	
BMI Operand	BMI Operand	Check N Flag
BNE Operand	BNE Operand	Check Z Flag
BPL Operand	BPL Operand	Check N Flag
BRK	SWI	(Requires Vector)
	• Interrupt Handler May Convert CC Format	
BVC Operand	TFR CC,DP	Save CC Register
	TST SEVFLG	Check V Flag Byte
	BNE • +6	Change 6 to 7 for LBRA
	TFR DP,CC	Restore CC Register
	BRA Operand	Branch if V Clear
	TFR DP, CC	Restore CC Register
BVS Operand	TFR CC,DP	Save CC Register
	TST SEVFLG	Check V Flag Byte
	BEQ • +6	Change 6 to 7 for LBRA
	TFR DP,CC	Restore CC Register
	BRA Operand	Branch if V Set
	TFR DP,CC	Restore CC Register
CLC	ANDCC #$FE	Clear C Flag
CLD	TFR CC,DP	Save CC Register
	CLR SEDFLG	Clear D Flag Byte
	TFR DP,CC	Restore CC Register
CLI	ANDCC #$EF	Clear I Flag
CLV	TFR CC,DP	Save CC Register
	CLR SEVFLG	Clear V Flag Byte
	TFR DP,CC	Restore CC Register
CMP Operand	CMPB Operand	Compare Accumulator
CPX Operand	EXG D,X	Prepare for Compare
	CMPB Operand	Compare X Register
	EXG X,D	
CPY Operand	EXG D,Y	Prepare for Compare
	CMPB Operand	Compare Y Register
	EXG Y,D	
DEC	DECB	Bump Accumulator Down
DEX	EXG X,D	Prepare for DEX
	LDA #$00	Clear MS 8 Bits, Not C Flag
	DECB	Bump X Down
	EXG D,X	Correct D and X
DEY	EXG Y,D	Prepare for DEY
	LDA #$00	Clear MS 8 Bits, Not C Flag
DECB	Bump Y Down	
	EXG D,Y	Correct D and Y
EOR Operand	EORB Operand	EOR Accumulator
INC	INCB	Bump Accumulator
INX	EXG X,D	Prepare for INX
	LDA #$00	Clear MS 8 Bits, Not C Flag
	INCB	Bump X Up
	EXG D,X	Correct D and X
INY	EXG Y,D	Prepare for INY
	LDA #$00	Clear MS 8 Bits, Not C Flag
INCB	Bump Y Up	
	EXG D,Y	Correct D and Y
JMP Operand	JMP Operand	Jump
JSR Operand	JSR Operand	Subroutine Call
LDA Operand	LDA Operand	Load Accumulator
LDX Operand	EXG X,D	Prepare for LDX
	LDA #$00	Clear MS 8 Bits, Not C Flag
	LDB Operand	Load Value
	EXG D,X	Correct D and X
LDY Operand	EXG Y,D	Prepare for LDY
	LDA #$00	Clear MS 8 Bits, Not C Flag
	LDB Operand	Load Value
	EXG D,Y	Correct D and Y
LSR Operand	LSR Operand	Logical Shift Right
NOP	NOP	No Operation
ORA Operand	ORB Operand	Or Accumulator
PHA	PSHS B	Push Accumulator
PHP	• Execute Cond Code Translation from 6809	
	PSHS A	Push 6502 CC Register
PLA	PULS B	Pull Accumulator
	TSTB	Set CC Register
PLP	PULS A	Pull 6502 CC Register
	• Execute Cond Code Translation to 6809	
ROL Operand	ROL Operand	Roll Left
ROR Operand	ROR Operand	Roll Right
RTI	RTI	Return from Interrupt
	• Interrupt Handler May Convert CC Format	
RTS	RTS	Exit Subroutine
SBC Operand	SBC Operand	Subtract with Borrow
	TFR CC,DP	Save CC Register
	TFR CC,A	
	ANDA #$02	
	STA SEVFLG	Set V Flag Byte
	• Warning: Decimal Flag Not Honored	
	TFR DP,CC	Restore CC Register
SEC	ORCC #$01	Set C Flag
SED	TFR CC,A	Save CC Register
	STA SEDFLG	Set D Flag Byte
	TFR A,CC	Restore CC Register
SEI	ORCC #$10	Set I Flag
STA Operand	TFR CC,DP	Save CC Register
	STB Operand	Store Accumulator
	TFR DP,CC	Restore CC Register
STX Operand	EXG X,D	Prepare for Store
	TFR CC,DP	Save CC Register
	STB Operand	Store X Register
	TFR DP,CC	Restore CC Register
	EXG D,X	Restore D and X
STY Operand	EXG Y,D	Prepare for Store
	TFR CC,DP	Save CC Register
	STB Operand	Store X Register
	TFR DP,CC	Restore CC Register
	EXG D,Y	Restore D and Y
TAX	LDA #$00	Clear MS 8 Bits, Not C Flag
	TSTB	Set CC Register
	TFR D,X	Set X to Accumulator
TAY	LDA #$00	Clear MS 8 Bits, Not C Flag
	TSTB	Set Condition Code
	TFR D,Y	Set Y to Accumulator
TSX	TFR D,U	Save D Register
	TFR S,D	Get S Register
	LDA #$00	Clear MS 8 Bits, Not C Flag
	DECB	Correct Value
	TFR D,X	Set X Register
	TFR U,D	Restore D Register
TXA	TFR X,D	Move X to Accumulator
	TSTB	Set CC Register
TXS	TFR D,U	Save D Register
	TFR X,D	Get X Register
	TFR CC,DP	Save CC Register
	INCB	Correct Value
	TFR DP,CC	Restore CC Register
	TFR D,S	Set S Register
	TFR U,D	Restore V Register
TYA	TFR Y,D	Move Y to Accumulator
	TSTB	Set CC Register

(Continued from page 81)

The 6809 has more instructions that modify the "V" flag than does the 6502, in which only the ADC, BIT, CLV, PLP, RTI, and SBC instructions modify the "V" flag. The 6502 "V" flag is thus easily emulated in the same manner as the "D" flag, with the same potential problems during interrupt processing.

Condition Code Register Format

Since the 6809 condition code register has format "EFHINZVC", and the 6502 condition code register has format "NV0BDIZC", two routines must be defined for the 6502 emulation, one to reformat condition codes in each direction. The routines are very similar; the following reformats the 6809 condition code register into 6502 format:

```
TFR CC,DP    Save CC Register
TFR D,U      Save D Register
TFR CC,A
CLRB         Zero 6502 Register
BITA #$10    I Flag
BEQ *+4
ORAB #$04
BITA #$08    N Flag
BEQ *+4
ORAB #$80
BITA #$04    Z Flag
BEQ *+4
ORAB #$20
TST SEVFLG   V Flag
BEQ *+4
ORAB #$40
BITA #$01    C Flag
BEQ *+4
ORAB #$01
TST SEDFLG   D Flag
BEQ *+4
ORAB #$80
TFR DP,CC    Restore CC Register
TFR B,DP
TFR U,D      Restore D Register
TFR DP,A     6502 CC in A Register
```

Again, since most programs never (or seldom) require the particular format of the 6502 condition code register, a programmer may decide to use the 6809-format condition code register and manually change the translated program, as required.

Page Zero Wraparound

Page zero wraparound is another attribute of the 6502 which is not present on the 6809 and must be handled by the translator through additional code if exact emulation is required. This problem occurs in the 6502 zero-page-indexed and indexed-indirect address modes. In the zero-page-indexed mode, the 8-bit offset in the 6502 instruction is added to the 8-bit value in the X or Y register to provide an 8-bit value, which is prefixed with 8-bit 00 to provide a 16-bit effective address. The 6809 code inserted by the translator would be in the following form:

```
TFR CC,DP          Save CC Register
LEAU ((address) AND
 $FF),X            Compute Address
EXG U,D
CLRA               Truncate to 8 Bits
EXG D,U            Address in U Register
TFR DP,CC          Restore CC Register
OPC ,U             Perform Original
                   Operation
```

The alternative to emulation would be to treat zero-page-indexed address mode as if it were absolute-indexed address mode. In this case the programmer would be responsible for ensuring that the correct effective address is calculated in each case. In the indexed-indirect mode, the 8-bit offset in the instruction is added to the 8-bit value in the X or Y register to form an 8-bit result, which is prefixed by an 8-bit 00 to form a 16-bit effective address. The contents of the locations at that address and at the next address are used to provide a 16-bit effective address. The 6809 code inserted by the translator would be similar to that provided earlier, with the exception of the last line, which would use indirect addressing and would be in the following form:

```
OPC [,U]           Perform Original
                   Operation
```

assuming that no indirect addresses are placed at $00FF and $0000. An alternative to emulation would be to directly use the 6809 indirect address facility, manually correcting any cases in which the contents of the X or Y register plus the offset exceeds $00FE.

The 8-Bit Limitation of X and Y

The 6502 8-bit X and Y register limitations affect the following 6502 instructions: DEX, DEY, INX, INY, LDX, LDY, STX, STY, TAX, TAY, TSX, TXA, TXS, TYA. In virtually every case, the 8-bit value being processed must be moved through the D register in order to properly extend or truncate the value. For instance, the translator-generated 6809 code for INX would be:

```
EXG X,D      Move X Register for
             Truncation
LDA #$00     Clear MS 8 Bits, Not C
             Flag
INCB         Bump Last 8 Bits of X
EXG D,X      Restore New X Register
```

The magnitude of the problems associated with the conversion of the translated program to fully use the 16-bit X and Y registers of the 6809 would depend on the program being translated. However, they may be severe, and the emulation overhead will usually be small.

Translation Analysis

Table 4 presents a simplified representation of the required translator actions in the conversion of each 6502 instruction to 6809 instructions. The following assumptions are made implicitly in this table:

- address mode processing is handled separately but always presents a 16-bit effective address
- absolute addresses are stored in 6809 format (high, then low bytes)
- stack register is handled using 6809 16-bit format and is not restricted to $01XX range
- format conversion of the condition code register is not handled:
  no "B" flag handling is required
  "D" and "V" flags are handled as separate flag bytes
- X and Y registers are restricted to 8 bits
- situations such as "too-long" branches must be handled by the programmer after translation

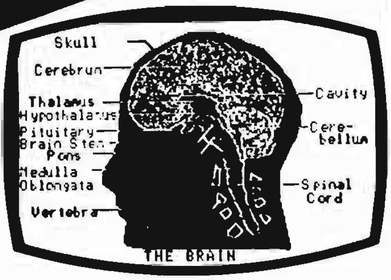
Conversion Analysis

Most computer programs, even on microcomputers, do not run standalone but run under control of an operating system or use external I/O, math, or service subroutines. Thus, even if the translation from 6502 to 6809 is exactly correct on an instruction-by-instruction basis, many 6502 programs would not run after translation without modification. The

Versa Computing

PRESENTS

VersaWriter DRAWING TABLET



COMPLETE HARDWARE / SOFTWARE GRAPHICS SYSTEM - $299

- Hi-Res & Med-Res Draw
- Paint Brush-5 Sizes
- Point to Point / Line Draw
- Air Brush
- Color Fill-In
- Change Color Hue & Intensity
- Reverse Picture
- Scaling
- Split / Full Screen
- Save / Load / Erase
- Text Writer
- Fix X or Y Axis

Requires: Atari ®300, 32K RAM, Basic Language Cartridge, Disk Drive

GRAPHICS COMPOSER

PADDLE / JOYSTICK GRAPHICS SOFTWARE - $39.95

- Draw on Hi-Res Screens 7 or 8
- Save Pictures on Disk or Cassette
- Create Player / Missile Shapes Automatically
- Geometric Figures Program
- Add Text to Screen

Requires: Atari®800, 32K RAM, Basic Language Cartridge, Disk or Cassette

GLOBE MASTER

COMPLETE HI-RES GEOGRAPHY GAME - $29.95

- 8 Hi-Res Color Maps
- U.S.-Europe-World-Asia-Africa-Australia
- Countries-Cities-Capitals-Oceans-Rivers-Mountains, Etc.
- Several Skill Levels

Requires: Atari®800, 32K RAM, Basic Language Cartridge, Disk

ATARI ®is a registered trademark of Atari Inc.

MIND BOGGLERS

THREE STRATEGY GAMES

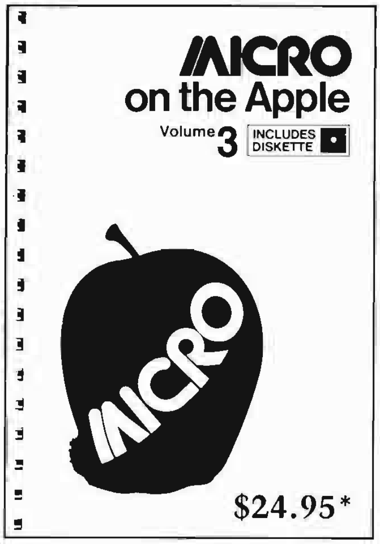
- Capture
- Mystery Box
- Simon Says

CASSETTE ***$15.95*** *DISK* ***$19.95***

Requires: Atari®400, 16K RAM, Cassette
Atari 800, 24K RAM, Cassette or Disk

Versa Computing, Inc.

3541 Old Conejo Road, Suite 104
Newbury Park, CA. 91320 **(805)498-1956**



portions of programs requiring change in a practical environment will generally be in the following areas:

- monitor, operating system, and subroutine library entry points
- I/O addresses and hardware
- memory-mapped video facilities
- miscellaneous tradeoffs made in translation.

Entry points may cause difficulties in terms of addresses, parameters, and functions. The address problems are usually the simplest to solve, since these generally involve merely changing addresses in EQU statements. The parameter-passing problem encompasses addresses and values passed to and from subroutines, monitor entry points, and operating system routines, and may be far more complex. The number of variations in table and control block format and usage, control value interpretation, data structure representation, method of returning results, etc., is astronomical.

The best plan of attack on these problems varies with the nature of the effort. In the case of a well-defined subroutine library or set of operating system routines being referenced, it may be possible and advantageous to code a set of 6809 routines to interface to a similar functional library or routines. Then this interface may be used in any program with few other changes in logic required.

I/O address and hardware differences may cause problems in conversion. Simply changing the EQU statements will probably not affect the complete conversion because of the differences in handling of the various I/O devices, such as VIO's, VIA's, PIA's, ACIA's, etc. These differences may be handled by coding interface subroutines, by modifing the code to handle the new I/O device in native mode, by using similar functional routines already available in the 6809 operating system, etc. In the worst case, the 6502 hardware facility may not even be available on the 6809, requiring extensive modifications.

Memory-mapped video facilities are available on many of the appliance computers as standard features but are not generally directly available on 6809 systems, with the notable exception of the Radio Shack Color Computer. If a 6502 program makes extensive use of memory-mapped video hardware, but the facility is not available on the 6809 or is available but is handled differently, several methods of translating the running 6502 program to become a running 6809 program are possible. The obvious means of performing the conversion, though sometimes the most difficult, would be to rewrite the 6502 code after translation to drive the video board or terminal used on the 6809 directly. Another method would be to write a terminal emulation routine which would make the same output appear on an output device on a 6809 as on a video monitor on a 6502. The method used in a given case will depend upon the situation.

The other primary reason for manual intervention in the conversion process involves the tradeoffs made in the translation. The changes required by this may benefit from some of the same organized attacks as suggested for the I/O and hardware problems. Other changes may be desirable to take advantage of the additional instructions and addressing modes of the 6809 *versus* the 6502.

Summary

The preceding discussion has presented a method to convert 6502 source programs to 6809 source programs. This conversion is performed in two phases.

The first phase is a low-level (instruction-by-instruction) translation process which could be performed manually or by using a computer program. The instruction emulation level may be varied to cause the translated program to have certain attributes closer to the 6502 or to the 6809 architectures, as desired.

The second phase is higher-level, and must generally be performed manually (although possibly with the assistance of an editing or special-purpose computer program) since it usually involves creativity and cleverness on a level not yet found in the most advanced computer programs. This process involves the resolution of the remaining differences between the translated 6502 program and the 6809 environment in which the 6809 program will run, and the final debugging and checkout.

Tables summarizing the instruction sets of the 6502, 6800, and 6809 processors follow.

Edgar Pass may be contacted at Computer Systems Consultants, Inc., 1454 Latta Lane, Conyers, GA 30207.

Table A-1: 6800,01,02,03,08 Op-Codes and Mnemonics

Operation	Mnemonic	Immediate	Direct	Indexed	Extended	Inherent
Add	ADDA	8B	9B	AB	BB	
	ADDB	CB	DB	EB	FB	
Add Double Acc	ADDD*	C3	D3	E3	F3	
Add Accum.	ABA					1B
Add With Carry	ADCA	89	99	A9	B9	
	ADCB	C9	D9	E9	F9	
And	ANDA	84	94	A4	B4	
	ANDB	C4	D4	E4	F4	
Bit Test	BITA	85	95	A5	B5	
	BITB	C5	D5	E5	F5	
Clear	CLR			6F	7F	
	CLRA					4F
	CLRB					5F
Compare	CMPA	81	91	A1	B1	
	CMPB	C1	D1	E1	F1	
Compare Accum.	CBA					11
Complement,1's	COM			63	73	
	COMA					43
	COMB					53
Complement,2's	NEG			6Ø	7Ø	
	NEGA					4Ø
	NEGB					5Ø
Dec Adj Acc.	DAA					19
Decrement	DEC			6A	7A	
	DECA					4A
	DECB					5A
Exclusive OR	EORA	88	98	A8	B8	
	EORB	C8	D8	E8	F8	
Increment	INC			6C	7C	
	INCA					4C
	INCB					5C
Load Accum.	LDAA	86	96	A6	B6	
	LDAB	C6	D6	E6	F6	
Load Doub Acc	LDAD*	CC	DC	EC	FC	
Multiply	MUL*					3D
Inclusive OR	ORAA	8A	9A	AA	BA	
	ORAB	CA	DA	EA	FA	
Push Data	PSHA					36
	PSHB					37
Pull Data	PULA					32
	PULB					33
Rotate Left	ROL			69	79	
	ROLA					49
	ROLB					59

* Not available in 68ØØ,68Ø2,or 68Ø8

Table A-1 *(continued)*

Operation	Mnemonic	Immediate	Direct	Indexed	Extended	Inherent
Shift Left	ASL			68	78	
Arithmetic	ASLA					48
	ASLB					58
Double	ASLD*					Ø5
Shift Right	ASR			67	77	
Arithmetic	ASRA					47
	ASRB					57
Shift Right	LSR			64	74	
Logical	LSRA					44
	LSRB					54
Double	LSRD*					Ø4
Store Accum	STAA		97	A7	B7	
	STAB		D7	E7	F7	
Doub. Accum.	STAD*		DD	ED	FD	
Subtract	SUBA	8Ø	9Ø	AØ	BØ	
	SUBB	CØ	DØ	EØ	FØ	
Double	SUBD*	83	93	A3	B3	
Subtract Acc.	SBA					1Ø
Subtract	SBCA	82	92	A2	B2	
With Carry	SBCB	C2	D2	E2	F2	
Transfer	TAB					16
Accumulators	TBA					17
Test Zero or	TST			6D	7D	
Minus	TSTA					4D
	TSTB					5D

* Not available in 68ØØ,68Ø2,or 68Ø8

Table A-2: Index Register and Stack Manipulation Instructions

Operation	Mnemonic	Immediate	Direct	Indexed	Extended	Implied
Compare IXR	CPX	8C	9C	AC	BC	
Decrement IXR	DEX					Ø9
Decrment SP	DES					34
Increment IXR	INX					Ø8
Increment SP	INS					31
Load IXR	LDX	CE	DE	EE	FE	
Load SP	LDS	8E	9E	AE	BE	
Store IXR	STX		DF	EF	FF	
Store SP	STS		9F	AF	BF	
IXR-->SP	TXS					35
SP-->IXR	TSX					3Ø
Add B to X	ABX*					3A
Push IXR	PSHX*					3C
Pull IXR	PULX*					38
Rotate Right	ROR			66	76	
	RORA					46
	RORB					56

* Not available in 68ØØ,68Ø2, or 68Ø8

Table A-3: 6800,01,02,03,08 Op-Codes and Mnemonics

CONDITION CODE REGISTER MANIPULATION INSTRUCTIONS

Operation	Mnemonic	Implied
Clear Carry	CLC	ØC
Clear Int Msk	CLI	ØE
Clr Overflow	CLV	ØA
Set Carry	SEC	ØD
Set Int Msk	SEI	ØF
Set Overflow	SEV	ØB
Acc A-->CCR	TAP	Ø6
CCR-->Acc A	TPA	Ø7

Table A-4: Jump and Branch Instructions

Operation	Mnemonic	Relative	Indexed	Extended	Implied
Branch Always	BRA	2Ø			
Branch if Carry Clear	BCC	24			
Branch if Carry Set	BCS	25			
Branch if = Zero	BEQ	27			
Branch if >= Zero	BGE	2C			
Branch if > Zero	BGT	2E			
Branch if Higher	BHI	22			
Branch if <= Zero	BLE	2F			
Branch if Lower/Same	BLS	23			
Branch if < Zero	BLT	2D			
Branch if Minus	BMI	2B			
Branch if Not = Zero	BNE	26			
Branch if V Clear	BVC	28			
Branch if V Set	BVS	29			
Branch if Plus	BPL	2A			
Branch to Subroutine	BSR	8D			
Jump	JMP		6E	7E	
Jump to Subroutine	JSR		AD	BD	
No Operation	NOP				Ø1
Return from Interrupt	RTI				3B
Return from Subroutine	RTS				39
Software Interrupt	SWI				3F
Wait for Interrupt	WAI				3E

Table B-1: 6809 Op-Codes and Mnemonics

Operation	Mnemonic	Immediate	Direct	Indexed	Extended	Inherent
Add B to X	ABX					3A
Add w/ carry	ADCA	89	99	A9*	B9	
	ADCB	C9	D9	E9*	F9	
Add	ADDA	8B	9B	AB*	BB	
	ADDB	CB	DB	EB*	FB	
	ADDD	C3	D3	E3*	F3	
And	ANDA	84	94	A4*	B4	
	ANDB	C4	D4	E4*	F4	
	ANDCC	1C				

Table B-1 *(continued)*

Operation	Mnemonic	Immediate	Direct	Indexed	Extended	Inherent
Arithmetic Shift Left	ASLA					48
	ASLB					58
	ASL		Ø8	68*	78	
Arithmetic Shift Right	ASRA					47
	ASRB					57
	ASR		Ø7	67*	77	
Bit Test	BITA	85	95	A5*	B5	
	BITB	C5	D5	E5*	F5	
Clear	CLRA					4F
	CLRB					5F
	CLR		ØF	6F*	7F	
Compare	CMPA	81	91	A1*	B1	
	CMPB	C1	D1	E1*	F1	
	CMPD	1Ø83	1Ø93	1ØA3*	1ØB3	
	CMPS	118C	119C	11AC*	11BC	
	CMPU	1183	1193	11A3*	11B3	
	CMPX	8C	9C	AC*	BC	
	CMPY	1Ø8C	1Ø9C	1ØAC*	1ØBC	
Complement,1's	COMA					43
	COMB					53
	COM		Ø3	63*	73	
Wait for int.	CWAI					3C
Dec. adj Acc.	DAA					19
Decrement	DECA					4A
	DECB					5A
	DEC		ØA	6A*	7A	
Exclusive OR	EORA	88	98	A8*	B8	
	EORB	C8	D8	E8*	F8	
Exchange Reg's	EXG**					1E
Increment	INCA					4C
	INCB					5C
	INC		ØC	6C*	7C	
Load	LDA	86	96	A6*	B6	
	LDB	C6	D6	E6*	F6	
	LDD	CC	DC	EC*	FC	
	LDS	1ØCE	1ØDE	1ØEE*	1ØFE	
	LDU	CE	DE	EE*	FE	
	LDX	8E	9E	AE*	BE	
	LDY	108E	1Ø9E	1ØAE*	1ØBE	
Load Effective Address	LEAS			32*		
	LEAU			33*		
	LEAX			3Ø*		
	LEAY			31*		

* Post byte required (see indexed addressing chart)
** Post byte specifying registers to be used is required.

Table B-1 *(continued)*

Operation	Mnemonic	Immediate	Direct	Indexed	Extended	Inherent
Logical shift Left	LSLA					48
	LSLB					58
	LSL		Ø8	68*	78	
Logical Shift	LSRA					44
	LSRB					54
	LSR		Ø4	64*	74	
Multiply	MUL					3D
Complement,2's	NEGA					4Ø
	NEGB					5Ø
	NEG		ØØ	6Ø*	7Ø	
No Operation	NOP					12
Inclusive OR	ORA	8A	9A	AA*	BA	
	ORB	CA	DA	EA*	FA	
	ORCC	1A				
Push Reg's on Stack	PSHS**					34
	PSHU**					36
Pull Reg's from Stack	PULS**					35
	PULU**					37
Rotate Left	ROLA					49
	ROLB					59
	ROL		Ø9	69*	79	
Rotate Right	RORA					46
	RORB					56
	ROR		Ø6	66*	76	
Subtract with Carry	SBCA	82	92	A2*	B2	
	SBCB	C2	D2	E2*	F2	
Sign Extend	SEX					1D
Store	STA		97	A7*	B7	
	STB		D7	E7*	F7	
	STD		DD	ED*	FD	
	STS		1ØDF	1ØEF*	1ØFF	
	STU		DF	EF*	FF	
	STX		9F	AF*	BF	
	STY		1Ø9F	1ØAF*	1ØBF	
Subtract	SUBA	8Ø	9Ø	AØ*	BØ	
	SUBB	CØ	DØ	EØ*	FØ	
	SUBD	83	93	A3*	B3	
Software Interrupt	SWI					3F
	SWI2					1Ø3F
	SWI3					113F
Sync to Int.	SYNC					13

Table B1 *(continued)*

Operation	Mnemonic	Immediate	Direct	Indexed	Extended	Inherent
Transfer Reg's	TFR**					1F
Test, Zero or Minus	TSTA					4D
	TSTB					5D
	TST		ØD	6D*	7D	

* Post byte required (see indexed addressing chart)
** Post byte specifying registers to be used is required.

Table B-2: Branch and Long Branch Instructions

Operation	Mnemonic	Relative	Direct	Indexed	Extended
Branch if Carry Clear	BCC	24			
	LBCC	1Ø24			
Branch if Carry Set	BCS	25			
	LBCS	1Ø25			
Branch if = Zero	BEQ	27			
	LBEQ	1Ø27			
Branch if >= Zero	BGE	2C			
	LBGE	1Ø2C			
Branch if > Zero	BGT	2E			
	LBGT	1Ø2E			
Branch if Higher	BHI	22			
	LBHI	1Ø22			
Branch if Higher/Same	BHS	24			
	LBHS	1Ø24			
Branch if <= Zero	BLE	2F			
	LBLE	1Ø2F			
Branch if Lower	BLO	25			
	LBLO	1Ø25			
Branch if Lower/Same	BLS	23			
	LBLS	1Ø23			
Branch if < Zero	BLT	2D			
	LBLT	1Ø2D			
Branch if Minus	BMI	2B			
	LBMI	1Ø2B			
Branch if Not = Zero	BNE	26			
	LBNE	1Ø26			
Branch if Plus	BPL	2A			
	LBPL	1Ø2A			
Branch Always	BRA	2Ø			
	LBRA	16			
Branch Never	BRN	21			
	LBRN	1Ø21			
Branch if V Clear	BVC	28			
	LBVC	1Ø28			
Branch if V Set	BVS	29			
	LBVS	1Ø29			
Branch to Subroutine	BSR	8D			
	LBSR	17			
Jump	JMP		ØE	6E*	7E
Jump to Subroutine	JSR		9D	AD*	BD
Return from Interrupt	RTI	3B (Implied)			
Return from Subroutine	RTS	39 (Implied)			

* Post byte required (see indexed addressing chart)

Table C: 6502 Op-Codes and Mnemonics

Operation	Mnemonic	Code	Addressing
Add with Carry	ADC	61	INDIRECT,X
	ADC	65	ZERO PAGE
	ADC	69	IMMEDIATE
	ADC	6D	ABSOLUTE
	ADC	71	INDIRECT,Y
	ADC	75	ZERO PAGE,X
	ADC	79	ABSOLUTE,Y
	ADC	7D	ABSOLUTE,X
And	AND	21	INDIRECT,X
	AND	25	ZERO PAGE
	AND	29	IMMEDIATE
	AND	2D	ABSOLUTE
	AND	31	INDIRECT,Y
	AND	35	ZERO PAGE,X
	AND	39	ABSOLUTE,Y
	AND	3D	ABSOLUTE,X
Arithmetic Shift Left	ASL	Ø6	ZERO PAGE
	ASL	ØA	ACCUMULATOR
	ASL	ØE	ABSOLUTE
	ASL	16	ZERO PAGE,X
	ASL	1E	ABSOLUTE,X
Branch	BCC	9Ø	RELATIVE
	BCS	BØ	RELATIVE
	BEQ	FØ	RELATIVE
	BMI	3Ø	RELATIVE
	BNE	DØ	RELATIVE
	BPL	1Ø	RELATIVE
	BVC	5Ø	RELATIVE
	BVS	7Ø	RELATIVE
Bit Test	BIT	24	ZERO PAGE
	BIT	2C	ABSOLUTE
Break	BRK	ØØ	IMPLIED
Clr Carry	CLC	18	IMPLIED
Clr Dec Mode	CLD	D8	IMPLIED
Clr Int Mask	CLI	58	IMPLIED
Clr Overflow	CLV	B8	IMPLIED
Jump to SR	JSR	2Ø	RELATIVE
Load Accumulator	LDA	A1	INDIRECT,X
	LDA	A5	ZERO PAGE
	LDA	A9	IMMEDIATE
	LDA	AD	ABSOLUTE
	LDA	B1	INDIRECT,Y
	LDA	B5	ZERO PAGE,X
	LDA	B9	ABSOLUTE,Y
	LDA	BD	ABSOLUTE,X

Operation	Mnemonic	Code	Addressing
Compare Accumulator	CMP	C1	INDIRECT,X
	CMP	C5	ZERO PAGE
	CMP	C9	IMMEDIATE
	CMP	CD	ABSOLUTE
	CMP	D1	INDIRECT,Y
	CMP	D5	ZERO PAGE,X
	CMP	D9	ABSOLUTE,Y
	CMP	DD	ABSOLUTE,X
Compare X	CPX	EØ	IMMEDIATE
	CPX	E4	ZERO PAGE
	CPX	EC	ABSOLUTE
Compare Y	CPY	CØ	IMMEDIATE
	CPY	C4	ZERO PAGE
	CPY	CC	ABSOLUTE
Decrement	DEC	C6	ZERO PAGE
	DEC	CE	ABSOLUTE
	DEC	D6	ZERO PAGE,X
	DEC	DE	ABSOLUTE,X
Decrement-X	DEX	CA	IMPLIED
Decrement-Y	DEY	88	IMPLIED
Exclusive Or	EOR	41	INDIRECT,X
	EOR	45	ZERO PAGE
	EOR	49	IMMEDIATE
	EOR	4D	ABSOLUTE
	EOR	51	INDIRECT,Y
	EOR	55	ZERO PAGE,X
	EOR	59	ABSOLUTE,Y
	EOR	5D	ABSOLUTE,X
Increment	INC	E6	ZERO PAGE
	INC	EE	ABSOLUTE
	INC	F6	ZERO PAGE,X
	INC	FE	ABSOLUTE,X
Increment-X	INX	E8	IMPLIED
Increment-Y	INY	C8	IMPLIED
Jump	JMP	4C	ABSOLUTE
	JMP	6C	INDIRECT
Rotate Left	ROL	26	ZERO PAGE
	ROL	2A	ACCUMULATOR
	ROL	2E	ABSOLUTE
	ROL	36	ZERO PAGE,X
	ROL	3E	ABSOLUTE,X
Rotate Right	ROR	66	ZERO PAGE
	ROR	6A	ACCUMULATOR
	ROR	6E	ABSOLUTE
	ROR	76	ZERO PAGE,X
	ROR	7E	ABSOLUTE,X

Table C *(continued)*

Operation	Mnemonic	Code	Addressing
Load X	LDX	A2	IMMEDIATE
	LDX	A6	ZERO PAGE
	LDX	AE	ABSOLUTE
	LDX	B6	ZERO PAGE,Y
	LDX	BE	ABSOLUTE,Y
Load Y	LDY	AØ	IMMEDIATE
	LDY	A4	ZERO PAGE
	LDY	AC	ABSOLUTE
	LDY	B4	ZERO PAGE,X
	LDY	BC	ABSOLUTE,X
Logical Shift Right	LSR	46	ZERO PAGE
	LSR	4A	ACCUMULATOR
	LSR	4E	ABSOLUTE
	LSR	56	ZERO PAGE,X
	LSR	5E	ABSOLUTE
No Oper.	NOP	EA	IMPLIED
Inclusive OR	ORA	Ø1	INDIRECT,X
	ORA	Ø5	ZERO PAGE
	ORA	Ø9	IMMEDIATE
	ORA	ØD	ABSOLUTE
	ORA	11	INDIRECT,Y
	ORA	15	ZERO PAGE,X
	ORA	19	ABSOLUTE,Y
	ORA	1D	ABSOLUTE,X
Push Data	PHA	48	IMPLIED
	PHP	Ø8	IMPLIED
Pull Data	PLA	68	IMPLIED
	PLP	28	IMPLIED
Transfer Registers	TAX	AA	IMPLIED
	TAY	A8	IMPLIED
	TSX	BA	IMPLIED
	TXA	8A	IMPLIED
	TXS	9A	IMPLIED
	TYA	98	IMPLIED

Operation	Mnemonic	Code	Addressing
Ret. f/Int.	RTI	4Ø	IMPLIED
Ret. f/SR	RTS	6Ø	IMPLIED
Subtract with Carry	SBC	E1	INDIRECT,X
	SBC	E5	ZERO PAGE
	SBC	E9	IMMEDIATE
	SBC	ED	ABSOLUTE
	SBC	F1	INDIRECT,Y
	SBC	F5	ZERO PAGE,X
	SBC	F9	ABSOLUTE,Y
	SBC	FD	ABSOLUTE,X
Set Carry	SEC	38	IMPLIED
Set Decimal	SED	F8	IMPLIED
Set Int Msk	SEI	78	IMPLIED
Store Accumulator	STA	81	INDIRECT,X
	STA	85	ZERO PAGE
	STA	8D	ABSOLUTE
	STA	91	INDIRECT,Y
	STA	95	ZERO PAGE,X
	STA	99	ABSOLUTE,Y
	STA	9D	ABSOLUTE,X
Store X	STX	86	ZERO PAGE
	STX	8E	ABSOLUTE
	STX	96	ZERO PAGE,Y
Store Y	STY	84	ZERO PAGE
	STY	8C	ABSOLUTE
	STY	94	ZERO PAGE,X

Note that, on the 65Ø2, Absolute addresses appear in low-order-byte-first sequence.

Aye! Come aboard for the fight of your life

SWASHBUCKLER

Fantastic HI-RES sword fighting

You're in a duel to the death against a blackhearted pirate crew. Look out! Here they come swinging and slashing with their pikes, swords, and knives.

Pirates to your right, pirates to your left . . . poison snakes, spiders, scorpions and killer rats! Hear your sword ring as you parry, thrust, lunge . . . run them through! If you're swordsman enough to win below decks, you go topside, and the fight continues . . . with Blackbeard himself and the motley crew. What a game! What a test of skill against opponents who really attack and fight back! What a chance to become the swordsman of the year! Get your copy now - thrill to the most realistic HI-RES graphics ever!

SWASHBUCKLER. Only **$34.95** for the Apple II*. At computer stores or:

DATAMOST
9748 Cozycroft Ave.
Chatsworth, Ca. 91311
(213) 709-1202

VISA/MASTERCHARGE accepted.
$1.00 shipping/handling charge.
(California residents add 6% tax)

*Apple II a trademark of Apple Computer, Inc.

TURN YOUR APPLE™ PERSONAL COMPUTER INTO A PROFESSIONAL COMPUTER FOR $750.

The majority of all professional computer software programs available today are written for the CP/M® disk operating system. The SYNERGIZER lets you access all of this vast body of sophisticated software with your Apple II while retaining the capability to access your present Apple software.

In addition to the CP/M interface and software diskette, the SYNERGIZER gives you the required 80 column display and 16K RAM memory expansion boards, the *CP/M Handbook* by Rodnay Zaks, and complete manuals. You get *everything* you need for fast, easy installation and operation in one package.* Each element is designed to complement the others, and everything is designed and produced by the *same* company.

The SYNERGIZER. It'll turn your Apple into a professional computer. And it costs only $750. Phone or write us, or ask your dealer for a SYNERGIZER brochure. Now.

SYNERGIZER™

Manufactured by Advanced Logic Systems,
1195 East Arques Avenue, Sunnyvale, CA 94086, (800) 538-8177 (In California (408) 730-0306)

Apple and Apple II are registered trademarks of Apple Computer, Inc.,
CP/M is a registered trademark of Digital Research, Inc.,
The CP/M Handbook is copyrighted by Sybex, Inc.
And the SYNERGIZER was *our* idea.
*All SYNERGIZER Components are also sold separately.

Auto Entry for the C1P

by Allan J. Zadiraka

This utility program provides the C1P with easier keyboard entry of machine-language programs. It also provides a scrolling screen display of the machine-code entries in either a hex dump or an object code listing format.

Auto Entry
requires:
OSI C1P
May be modified for other computers

Recently, the failure of a cassette tape recorder resulted in the loss of a 500-byte program that had been entered using the ROM monitor in my OSI C1P. This event provided the incentive I needed to produce a program to reduce the effort required for keyboard entry of machine-language programs. The first pass at the program was based on only two objectives:

1. To reduce the number of keystrokes needed to enter a program by eliminating the need for a carriage return after entering each byte of data (Auto Entry).
2. To utilize the video display capabilities of the C1P to provide a scrolling screen display of address and data entries.

The initial version of the program proved its worth when it was used to enter a published version of a 4K extended monitor for the C1P. However, this experience showed the need to add several control functions to the program to simplify its use. The listing shows the second version of the program, which allows:

- the ability to return to the monitor or to establish a new start address without having to resort to the BREAK key
- display of the entered data in either a hex dump format of eight bytes per line, or in an object code listing format of one, two, or three bytes per line.
- display of current address on the same line as the data or on a separate line
- backspacing to allow correction of a data entry.

The program occupies 248 bytes of memory and is located in the first page of BASIC workspace.

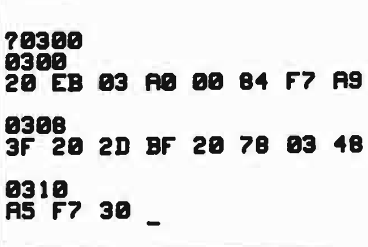
After the program has been loaded using the monitor, the start address of the program ($0300) is entered in the monitor display, and "G" is pressed to begin execution of the program. The program will display a "?", asking where the data is to be stored. After the four hex characters required to specify the address have been entered, the current address is displayed. If the program is working, this should be identical to the starting address entered.

The program then switches to the data entry mode. As each hex character is entered, it is displayed with a space automatically inserted between successive pairs of hex characters. Similar to the C1P monitor, the display shows the data entered and not the contents of the specified address. It will appear that you are writing data into ROM memory or non-existent memory if the current address is not selecting a valid RAM location. After eight bytes of data have been entered, the current address will be re-displayed on a new line and eight more bytes of data will be allowed.

Hitting "Z" toggles the program between the hex dump format shown in figure 1 and the object code format shown in figure 2. The C1P's 24-character line dictates the need for the two data entry display formats. In the hex dump format, eight bytes of data just fit on one line, requiring the address to be shown on a separate line. For the object code format I wanted to show the address on the same line as the data, similar to an object code listing. If a longer line length is possible on a specific machine, the toggle feature may be deleted from the program. Alternately, 16 bytes of data could be shown on a line, rather than eight, if at least a 48-character line is available. A 48-character line is supposedly available with the C1P Series 2 machines.

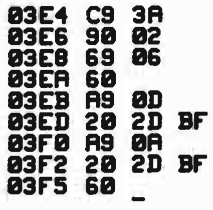
If fewer than eight bytes of data have been entered, pressing the RETURN key will end the current data line and then display the current address. The use of the the RETURN key

***Figure 1:* Screen Display for Hex Dump Format**



```
?0300
0300
20 EB 03 A0 00 84 F7 A9

0308
3F 20 2D BF 20 78 03 48

0310
A5 F7 30 _
```

***Figure 2:* Screen Display for Object Code Format**



```
03E4 C9 3A
03E6 90 02
03E8 69 06
03EA 60
03EB A9 0D
03ED 20 2D BF
03F0 A9 0A
03F2 20 2D BF
03F5 60 _
```

is necessary to end a line if the object code format is desired.

The program will come up in the hex dump format when initially executed. If you want to have the program come up in the object code format, the contents of address $0336 should be changed from $18 (CLC) to $38 (SEC) before the program is saved on tape. "Z" toggles this byte between these two values to change the display mode.

"R" decreases the current address by one and shows the new current address. This feature can be used to backspace to the point where an entry error occurred. The correct data can then be re-entered from that point on. "O" will start the program again to allow a new starting address or origin to be set.

The ESCAPE key causes a return to the C1P's monitor. When the "G" is pressed, the C1P's monitor executes programs by an unconditional jump to the program. Therefore the exit back to the monitor also must be executed by an unconditional jump (JMP). If this program is to be used with an extended monitor that executes programs as subroutines, the return to the monitor should be changed to a return from subroutine (RTS) as noted in the listing. Otherwise you will end up back in the C1P's ROM monitor instead of the extended monitor.

This utility reduces the time it takes to enter programs by fully 50%. Most of the time is saved in the keystroking process: 33% fewer are now needed. And when you lose track of exactly where you are when keying a program in, you now do not have to use the monitor to step through memory to find your place.

The use of an assembler would, in theory, eliminate the need to enter object code from the keyboard. However, if you are limited to hand assembly of programs, or you want to enter the object code from a published program without having to type in all the source code, you'll appreciate the features of this program.

Mr. Zadiraka is an electrical engineer involved in the development of instrumentation and control systems for coal-fired power plants. He has an OSI C1P with 8K of memory and a balky cassette recorder. His address is 4110 State Road, Akron, Ohio 44319.

```
                   ;*******************************;
                   ;                               ;
                   ;   AUTO ENTRY UTILITY PROGRAM  ;
                   ;   Version 2.0                 ;
                   ;   AJ ZADIRAKA  AUGUST 14,1981 ;
                   ;                               ;
                   ;*******************************;
                   ;
                   ; ZERO PAGE TEMPORARY STORAGE
                   ADRLO  =$FE             ;CURRENT ADDRESS
                   ADRHI  =ADRLO+1
                   TEMP   =$F9
                   TEMPCH =$FA
                   FLAG   =$F7             ;SPECIAL COMMAND FUNCTION
                   ;C1P ROM UTILITY ADDRESSES
                   ;
                   CRTPRN =$BF2D           ;PRINT CHAR IN ACCUM ON
                   ;                      ;SCREEN AT CURSOR LOCATION
                   ;
                   LEGAL  =$FE93           ;CONVERT ASCII TO HEX
                   ;                      ;($80 IF NOT HEX)
                   ;
                   INCHAR =$FEE9           ;GET CHAR FROM KEYBOARD
                   ;
                   MONTR  =$FE43           ;C1P MONITOR
                         .=$0300           ;START ADDRESS OF PROGRAM
                   ;
0300: 20 ED 03     AUTOEN JSR CRLF         ;PRINT <CR-LF>
0303: A0 00               LDY #0
0305: 84 F7               STY FLAG         ;CLEAR SPECIAL FUNC FLAG
0307: A9 3F               LDA #'?          ;PROMPT USER WITH '?'
0309: 20 2D BF            JSR CRTPRN
030C: 20 78 03            JSR GETBYT       ;INPUT HI BYTES OF ADDR
                   ;
030F: 48                  PHA               ;CHECK FOR SPECIAL COMMAND
0310: A5 F7               LDA FLAG         ;FUNCTION KEY
0312: 30 5D               BMI DEGLCH       ;IF SO, DO FUNCTION
0314: 68                  PLA
                   ;
0315: 85 FF               STA ADRHI        ;ELSE SAVE BYTE
0317: 20 78 03            JSR GETBYT       ;GET LO BYTE OF ADDR
                   ;
031A: 48                  PHA               ;CHECK FOR SPECIAL COMMAND
031B: A5 F7               LDA FLAG         ;FUNCTION KEY
031D: 30 52               BMI DEGLCH       ;IF SO, DO FUNCTION
031F: 68                  PLA
                   ;
0320: 85 FE               STA ADRLO        ;ELSE SAVE BYTE
0322: 84 F7        NEWLIN STY FLAG         ;CLEAR SPECIAL FUNC FLAG
0324: 20 ED 03            JSR CRLF         ;PRINT<CR-LF>
0327: A5 FF               LDA ADRHI        ;PRINT CURRENT ADDRESS
0329: 20 CF 03            JSR UNPACK
032C: A5 FE               LDA ADRLO
032E: 20 CF 03            JSR UNPACK
0331: A9 20               LDA #$20         ;PRINT SPACE
0333: 20 2D BF            JSR CRTPRN
0336: 18           MODE   CLC               ;MODE CHANCE (CLC OR SEC)
0337: 90 03               BCC NOCRLF
0339: 20 ED 03            JSR CRLF
033C: A2 08        NOCRLF LDX #8
033E: 20 78 03     LOOP   JSR GETBYT       ;INPUT BYTE (TWO ASCII CHAR)
0341: 48                  PHA               ;CHECK FOR SPECIAL COMMAND
0342: A5 F7               LDA FLAG         ;FUNCTION KEY
0344: 30 13               BMI SPCL         ;IF SO, DO FUNCTION
0346: 68                  PLA
0347: 91 FE               STA (ADRLO),Y    ;ELSE SAVE BYTE IN ADDR
0349: E6 FE               INC ADRLO        ;THEN INC ADDRESS
034B: D0 02               BNE SKIP
034D: E6 FF               INC ADRHI
034F: A9 20        SKIP   LDA #$20         ;PRINT SPACE BETWEEN BYTES
0351: 20 2D BF            JSR CRTPRN
0354: CA                  DEX               ;BYTE COUNT
0355: F0 CB               BEQ NEWLIN       ;IF 8TH BYTE THEN NEW LINE
0357: D0 E5               BNE LOOP         ;ELSE GET NEXT BYTE
0359: 68           SPCL   PLA
035A: C9 0D               CMP #$0D         ;CARRIAGE RETURN?
035C: F0 C4               BEQ NEWLIN       ;START A NEW LINE
035E: C9 52               CMP #'R          ;KEY R (REVERSE)?
0360: D0 0B               BNE CK1
0362: A5 FE               LDA ADRLO        ;DECREASE ADDRESS
0364: D0 02               BNE S1
0366: C6 FF               DEC ADRHI
0368: C6 FE        S1     DEC ADRLO
036A: 4C 22 03            JMP NEWLIN
036D: C9 5A        CK1    CMP #'Z          ;KEY Z (MODE CHANGE)?
036F: F0 B1               BEQ NEWLIN       ;START A NEW LINE
```

```
0371: C9 4F     DEGLCH CMP #'O           ;KEY O (ORGIN)?
0373: F0 8B            BEQ AUTOEN        ;SET NEW ADDRESS
0375: 4C 43 FE         JMP MONTR         ;IF ESC KEY (MONITOR) OR
                ;                       ;OR ERROR THEN
                ;                       ;RETURN TO MONITOR (CHANGE
                ;                       ;TO RTS IF ENTERED BY JSR)
                ;
0378: 20 A3 03  GETBYT JSR GETCH         ;INPUT FIRST HEX CHAR OF BYTE
037B: 48               PHA                ;CHECK FOR SPECIAL COMMAND
037C: A5 F7            LDA FLAG          ;FUNCTION KEY
037E: 30 21            BMI EXIT1         ;IF SO EXIT SUBROUTINE
0380: 68               PLA
0381: 0A               ASL A
0382: 0A               ASL A             ;SHIFT TO 4 MSB'S
0383: 0A               ASL A
0384: 0A               ASL A
0385: 85 F9            STA TEMP          ;SAVE IT
0387: A5 FA            LDA TEMPCH        ;RECALL ASCII
0389: 20 2D BF         JSR CRTPRN        ;& PRINT ON SCREEN
038C: 20 A3 03         JSR GETCH         ;INPUT SECOND HEX CHAR
                ;
038F: 48               PHA                ;CHECK FOR SPECIAL COMMAND
0390: A5 F7            LDA FLAG          ;FUNCTION KEY
0392: 30 0D            BMI EXIT1         ;IF SO EXIT SUBROUTINE
0394: 68               PLA
                ;
0395: 05 F9            ORA TEMP          ;COMBINE WITH 1ST HEX
0397: 85 F9            STA TEMP          ;& SAVE BYTE
0399: A5 FA            LDA TEMPCH        ;RECALL ASCII
039B: 20 2D BF         JSR CRTPRN        ;& PRINT IT
039E: A5 F9            LDA TEMP          ;RECALL BYTE
03A0: 60               RTS
03A1: 68        EXIT1  PLA
03A2: 60               RTS
                ;
                ;
03A3: 20 E9 FE  GETCH  JSR INCHAR        ;GET ASCII CHAR FROM KEYBOARD
03A6: C9 0D            CMP #$0D          ;CR KEY?
03A8: F0 1A            BEQ EXIT2
03AA: C9 1B            CMP #$1B          ;ESC KEY
03AC: F0 16            BEQ EXIT2
03AE: C9 52            CMP #'R           ;BACKSPACE
03B0: F0 12            BEQ EXIT2
03B2: C9 4F            CMP #'O           ;ORIGIN
03B4: F0 0E            BEQ EXIT2
03B6: C9 5A            CMP #'Z           ;MODE CHANGE
03B8: D0 0D            BNE L1
03BA: 48               PHA
03BB: A9 20            LDA #%00100000
03BD: 4D 36 03         EOR MODE          ;TOGGLE MODE BETWEEN
03C0: 8D 36 03         STA MODE          ;CLC AND SEC
03C3: 68               PLA
03C4: C6 F7     EXIT2  DEC FLAG          ;SET UP SPECIAL FUNCTION
                ;                       ;KEY FLAG
03C6: 60               RTS
03C7: 85 FA     L1     STA TEMPCH        ;SAVE ASCII
03C9: 20 93 FE         JSR LEGAL         ;CONVERT ASCII TO HEX CHAR
03CC: 30 D5            BMI GETCH         ;IF NOT HEX, TRY AGAIN
03CE: 60               RTS
                ;
                ;
03CF: 48        UNPACK PHA                ;CONVERT A BYTE TO 2 ASCII CHAR
03D0: 4A               LSR A             ;AND PRINT THE ASCII CHARS
03D1: 4A               LSR A
03D2: 4A               LSR A
03D3: 4A               LSR A
03D4: 20 E4 03         JSR CONVRT
03D7: 20 2D BF         JSR CRTPRN
03DA: 68               PLA
03DB: 29 0F            AND #$0F
03DD: 20 E4 03         JSR CONVRT
03E0: 20 2D BF         JSR CRTPRN
03E3: 60               RTS
03E4: 09 30     CONVRT ORA #$30
03E6: C9 3A            CMP #$3A
03E8: 90 02            BCC PASS
03EA: 69 06            ADC #$06
03EC: 60        PASS   RTS
                ;
                ;
03ED: A9 0D     CRLF   LDA #$0D          ;PRINT A CARRIAGE RETURN
03EF: 20 2D BF         JSR CRTPRN
03F2: A9 0A            LDA #$0A          ;PRINT A LINE FEED
03F4: 20 2D BF         JSR CRTPRN
03F7: 60               RTS
```

New Publications

Microcomputer Experimentation with the MOS Technology KIM-1, by Lance A. Leventhal. Prentice-Hall, Inc. (Englewood Cliffs, NJ 07632), 1982, 467 pages, 6¾ × 9¼ inches, paperback.
ISBN: 0-13-580779-4 $16.95

An easy-to-follow guide to MOS Technology KIM-1 experiments, which includes an overview, two major groupings of experiments, and a list of references.

CONTENTS: Laboratory 0 — Basic Operations; Laboratory 1 — Writing and Running Simple Programs; Laboratory 2 — Simple Input; Laboratory 3 — Simple Output; Laboratory 4 — Processing Data Inputs; Laboratory 5 — Processing Data Outputs; Laboratory 6 — Processing Data Arrays; Laboratory 7 — Forming Data Arrays; Laboratory 8 — Designing and Debugging Programs; Laboratory 9 — Arithmetic; Laboratory A — Subroutines and the Stack; Laboratory B — Input/Output Using Handshakes; Laboratory C — Interrupts; Laboratory D — Timing Methods; Laboratory E — Serial Input/Output; Laboratory F — Microcomputer Timing and Control; Appendices; References; Index.

Speaking Pascal: A Computer Language Primer, by Kenneth A. Bowen. Hayden Book Company, Inc. (50 Essex St., Rochelle Park, NJ 07662), 1981, 236 pages, 5¾ × 9 inches, paperback.
ISBN: 0-8104-5164-6 $11.95

A ten-chapter introduction to programming in Pascal, written in a non-mathematical language that requires no technical background or previous programming experience to understand.

CONTENTS: Preface; First Words; Simple Conversations; Controlling the Conversation; Sophisticated Conversations; Last Words; Index.

Computer Choices, by H. Dominic Covvey and Neil Harding McAlister. Addison-Wesley (Reading, MA 01867), 1982, 225 pages, 6¼ × 9¼ inches, paperback.
ISBN 0-201-10113-0 $8.95

Computer Choices is designed to alert potential consumers to the pitfalls of purchasing a computer system. It dispells the myth that all computers are easy to use and gives advice on choosing and implementing the right computer system for home or office.

(Continued on page 117)

SoftSide™

CAN'T AFFORD SOFTWARE? READ THIS:

If the high price of commercial software and the lack of clear information about your microcomputer has got you down, here's the solution you've been waiting for!

SoftSide Magazine

SoftSide is a favorite of computer users and hobbyists alike. They rely on it as a prime source of programs, reviews and articles for the Apple™, ATARI®, and TRS-80® microcomputers. **SoftSide** is the magazine for the microcomputer owner who wants to learn BASIC programming, learn MORE about BASIC programming, or just wants to have FUN! **SoftSide** gives you the BASIC code listings of several programs — adventures, utilities, games, simulations, you name it — for your computer EVERY MONTH.

There's more:

- **Reviews** — of the software and hardware products you want to know about.
- **Articles** — about all aspects of BASIC programming.
- **Tutorials** — on graphics, use of important commands, and more.
- **Programs** — each month **SoftSide** publishes a variety of program for the Apple, ATARI® and TRS-80®.
- **Columns** — which discuss different topics including: computer graphics, picking the right modem for you and marketing your software — just to name a few.
- **Input from our readers** — each month we devote a space in the magazine to let our readers give us some feedback about **SoftSide**.
- **Hints & Enhancements** — programmers and readers provide us with enhancements, to our programs, and programming tips.

Use coupon to order. Mail to: **SoftSide** Publications, 6 South St., Milford, NH 03055

As you can see, you'll receive pages and pages of information and entertainment from SoftSide. Here's the best part:

A subscription to SoftSide is just $24 a year. That's 12 issues for only $2 each! What a value!

□ **YES! Send me the first copy of my SoftSide subscription right away!**

$24/yr for USA and Canada only. For orders to APO/FPO or Mexico — $40/yr. Other foreign orders — $62/yr.

I own a □ Apple □ ATARI® □ TRS-80®

Name ____________________
Address ____________________
City/State ____________________ Zip ________

□ Check is enclosed
□ MasterCard □ VISA

Name of Cardholder ____________________
MC# and Interbank#/VISA# ____________________
Exp. Date ____________________
Signature ____________________

MICRO™

Updates and Microbes

"Zoom and Squeeze" Update

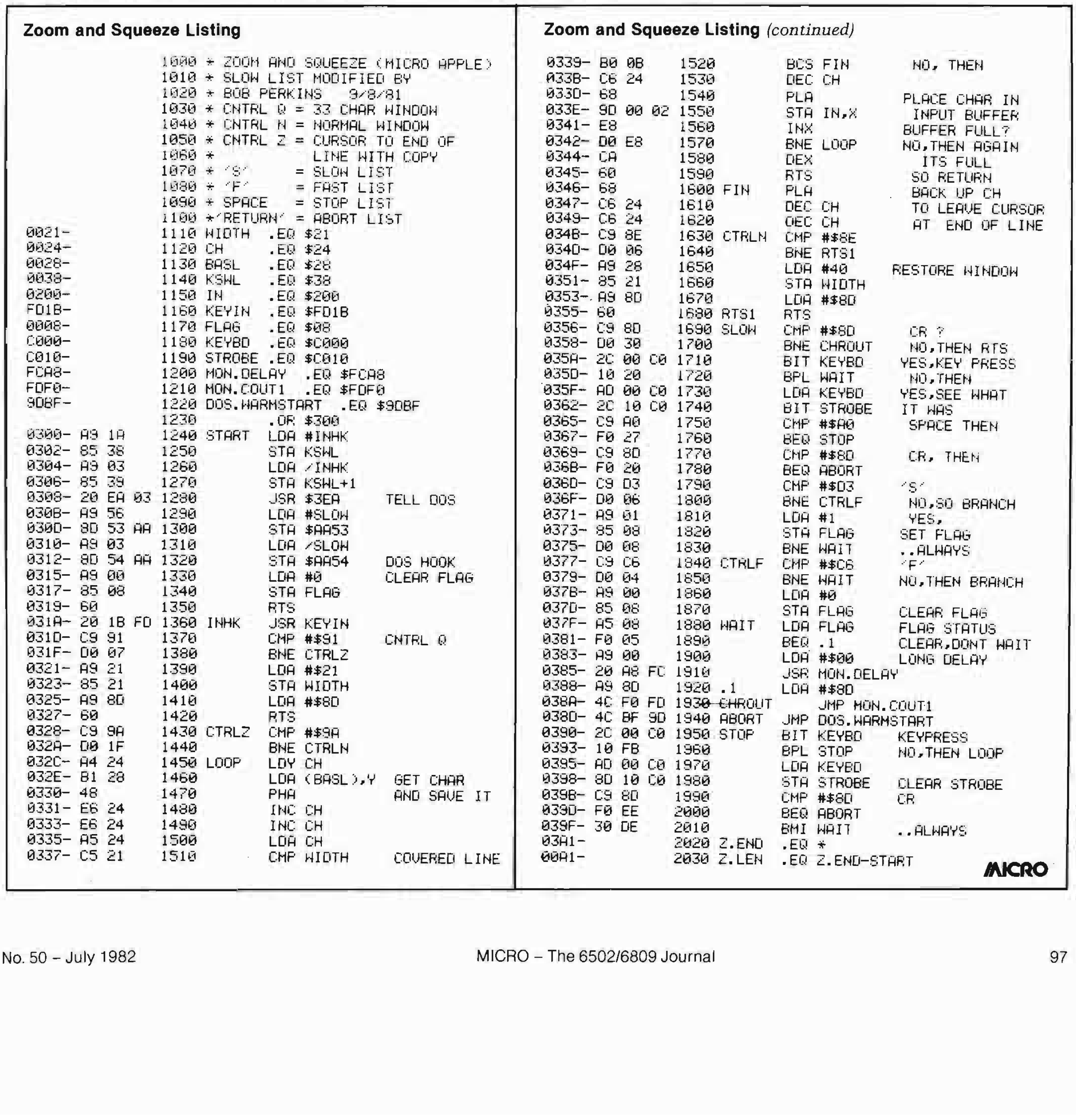
Bob Perkins of Tussy, OK, offered the updates in listing 1 (see below) to Gary B. Little's "Zoom and Squeeze" article. Little's article most recently appeared in MICRO *on the Apple, Volume 1, and originally in* MICRO *26:37.*

Co-Author Omission

Editor's Note: We apologize for not crediting Daniel P. Gerrity with his share of the work on the article "Microcomputer Interfacing: FORTH vs. BASIC," MICRO *49:77. Mr. Gerrity's name was ommitted from the Table of Contents and byline. Also inadvertently omitted was acknowledgment of the authors' research advisors, Professors K.S. Peters and V. Vaida, for whose support the authors express their appreciation.*

May Data Sheet Credits

Editor's Note: The chart on page 2 of the PET/CBM Data Sheet (May '82, page 108) needs further explanation. It applies not only to the PET, but also to CBM and VIC models. The chart was designed by Jim Butterfield and published in The Transactor *(Vol. 3, #4). Our apologies to Jim Butterfield for not giving him the credit he deserves. The* MICRO *staff corrected a couple of minor errors and greatly improved the readability of the graphic characters.*



Zoom and Squeeze Listing

```
                  1000 * ZOOM AND SQUEEZE (MICRO APPLE)
                  1010 * SLOW LIST MODIFIED BY
                  1020 * BOB PERKINS   9/8/81
                  1030 * CNTRL Q = 33 CHAR WINDOW
                  1040 * CNTRL N = NORMAL WINDOW
                  1050 * CNTRL Z = CURSOR TO END OF
                  1060 *           LINE WITH COPY
                  1070 * 'S'     = SLOW LIST
                  1080 * 'F'     = FAST LIST
                  1090 * SPACE   = STOP LIST
                  1100 *'RETURN' = ABORT LIST
0021-             1110 WIDTH  .EQ $21
0024-             1120 CH     .EQ $24
0028-             1130 BASL   .EQ $28
0038-             1140 KSWL   .EQ $38
0200-             1150 IN     .EQ $200
FD1B-             1160 KEYIN  .EQ $FD1B
0008-             1170 FLAG   .EQ $08
C000-             1180 KEYBD  .EQ $C000
C010-             1190 STROBE .EQ $C010
FCA8-             1200 MON.DELAY  .EQ $FCA8
FDF0-             1210 MON.COUT1  .EQ $FDF0
9DBF-             1220 DOS.WARMSTART  .EQ $9DBF
                  1230        .OR $300
0300- A9 1A       1240 START  LDA #INHK
0302- 85 38       1250        STA KSWL
0304- A9 03       1260        LDA /INHK
0306- 85 39       1270        STA KSWL+1
0308- 20 EA 03    1280        JSR $3EA      TELL DOS
030B- A9 56       1290        LDA #SLOW
030D- 8D 53 AA    1300        STA $AA53
0310- A9 03       1310        LDA /SLOW
0312- 8D 54 AA    1320        STA $AA54     DOS HOOK
0315- A9 00       1330        LDA #0        CLEAR FLAG
0317- 85 08       1340        STA FLAG
0319- 60          1350        RTS
031A- 20 1B FD    1360 INHK   JSR KEYIN
031D- C9 91       1370        CMP #$91      CNTRL Q
031F- D0 07       1380        BNE CTRLZ
0321- A9 21       1390        LDA #$21
0323- 85 21       1400        STA WIDTH
0325- A9 8D       1410        LDA #$8D
0327- 60          1420        RTS
0328- C9 9A       1430 CTRLZ  CMP #$9A
032A- D0 1F       1440        BNE CTRLN
032C- A4 24       1450 LOOP   LDY CH
032E- B1 28       1460        LDA (BASL),Y   GET CHAR
0330- 48          1470        PHA            AND SAVE IT
0331- E6 24       1480        INC CH
0333- E6 24       1490        INC CH
0335- A5 24       1500        LDA CH
0337- C5 21       1510        CMP WIDTH      COVERED LINE
```

Zoom and Squeeze Listing *(continued)*

```
0339- B0 0B       1520        BCS FIN        NO, THEN
033B- C6 24       1530        DEC CH
033D- 68          1540        PLA           PLACE CHAR IN
033E- 9D 00 02    1550        STA IN,X       INPUT BUFFER
0341- E8          1560        INX           BUFFER FULL?
0342- D0 E8       1570        BNE LOOP      NO,THEN AGAIN
0344- CA          1580        DEX             ITS FULL
0345- 60          1590        RTS            SO RETURN
0346- 68          1600 FIN    PLA            BACK UP CH
0347- C6 24       1610        DEC CH         TO LEAVE CURSOR
0349- C6 24       1620        DEC CH         AT  END OF LINE
034B- C9 8E       1630 CTRLN  CMP #$8E
034D- D0 06       1640        BNE RTS1
034F- A9 28       1650        LDA #40      RESTORE WINDOW
0351- 85 21       1660        STA WIDTH
0353- A9 8D       1670        LDA #$8D
0355- 60          1680 RTS1   RTS
0356- C9 8D       1690 SLOW   CMP #$8D       CR ?
0358- D0 30       1700        BNE CHROUT     NO,THEN RTS
035A- 2C 00 C0    1710        BIT KEYBD     YES,KEY PRESS
035D- 10 20       1720        BPL WAIT       NO,THEN
035F- AD 00 C0    1730        LDA KEYBD     YES,SEE WHAT
0362- 2C 10 C0    1740        BIT STROBE    IT WAS
0365- C9 A0       1750        CMP #$A0       SPACE THEN
0367- F0 27       1760        BEQ STOP
0369- C9 8D       1770        CMP #$8D       CR, THEN
036B- F0 20       1780        BEQ ABORT
036D- C9 D3       1790        CMP #$D3      'S'
036F- D0 06       1800        BNE CTRLF      NO,SO BRANCH
0371- A9 01       1810        LDA #1         YES,
0373- 85 08       1820        STA FLAG      SET FLAG
0375- D0 08       1830        BNE WAIT      ..ALWAYS
0377- C9 C6       1840 CTRLF  CMP #$C6      'F'
0379- D0 04       1850        BNE WAIT      NO,THEN BRANCH
037B- A9 00       1860        LDA #0
037D- 85 08       1870        STA FLAG      CLEAR FLAG
037F- A5 08       1880 WAIT   LDA FLAG      FLAG STATUS
0381- F0 05       1890        BEQ .1        CLEAR,DONT WAIT
0383- A9 00       1900        LDA #$00      LONG DELAY
0385- 20 A8 FC    1910        JSR MON.DELAY
0388- A9 8D       1920 .1     LDA #$8D
038A- 4C F0 FD    1930 CHROUT     JMP MON.COUT1
038D- 4C BF 9D    1940 ABORT  JMP DOS.WARMSTART
0390- 2C 00 C0    1950 STOP   BIT KEYBD     KEYPRESS
0393- 10 FB       1960        BPL STOP      NO,THEN LOOP
0395- AD 00 C0    1970        LDA KEYBD
0398- 8D 10 C0    1980        STA STROBE    CLEAR STROBE
039B- C9 8D       1990        CMP #$8D      CR
039D- F0 EE       2000        BEQ ABORT
039F- 30 DE       2010        BMI WAIT      ..ALWAYS
03A1-             2020 Z.END  .EQ *
00A1-             2030 Z.LEN  .EQ Z.END-START
```

MICRO

TAPDUP — AIM Tape Copy Utility

by Joel Swank

This article will provide you with an easy way to back up your AIM cassettes by controlling two recorders at the same time.

TAPDUP
requires:
AIM 65
and two tape recorders

```
                    ;    TAPDUP : AIM TAPE DUPLICATOR

                    ;    ZERO PAGE

                    FILGAP =$0             ;QUARTER SECONDS BETWEEN FILES
                    BLKCNT =$1             ;COUNT OF BLOCKS TO COPY

                    ;    AIM RAM

                    BLK    =$115           ;BLOCK COUNT
                    TABUFF =$116           ;TAPE BUFFER
                    GAP    =$A409          ;TAPE INTER-BLOCK GAP
                    ADDR   =$A41C          ;ADDRESS INPUT AREA
                    TAPIN  =$A434          ;INPUT DRIVE
                    TAPOUT =$A435          ;OUTPUT DRIVE
                    TAPTR2 =$A437          ;TAPE BUFF PTR

                    ;    AIM SUBROUTINES

                    ADDIN  =$EAAE          ;INPUT ADDRESS
                    TIBY1  =$ED53          ;INPUT BLOCK TO TAPE
                    REDOUT =$E973          ;GET CHAR FROM KBD
                    CRLOW  =$EA13          ;CR/LF TO DISPLAY
                    OUTPUT =$E97A          ;ACCUM TO DISPLAY
                    CKER0  =$E38E          ;ERROR MESSAGE
                    BKCKSM =$F1E7          ;COMPUTE CHECKSUM
                    TAOSET =$F21D          ;START OUTPUT DRIVE
                    OUTTAP =$F24A          ;SEND CHAR TO TAPE
                    CKBUFF =$F1D2          ;LOAD FROM ACTIVE BUFFER

                    ;    AIM 6532 TIMER

                    RINT   =$A485          ;TIME OUT FLAG
                    DI1024 =$A497          ;1024 MS TIMER

                    ;    TAPE I/O PORT

                    DRB    =$A800          ;DATA REG
                    ACR    =$A80B          ;AUX CONTROL REG

                    ;    EQUATES

                    BELL   =7              ;ASCII BELL CHAR

                           *=$200

0200  D8            TAPDUP CLD
0201  A9 20                LDA #$20        ;SET TAPE GAP
0203  8D 09 A4             STA GAP
0206  20 13 EA      REDRV  JSR CRLOW       ;NEW LINE
0209  A2 15                LDX #DRVMSG-LITS
020B  20 FC 02             JSR KEPX        ;REQUEST INPUT DRIVE
020E  20 73 E9      GETANS JSR REDOUT      ;GET REPLY
0211  C9 31                CMP #'1'        ;ALLOW ONLY 1 OR 2
0213  F0 0A                BEQ TAPOK
0215  C9 32                CMP #'2
0217  F0 06                BEQ TAPOK
0219  20 8E E3             JSR CKER0       ;ELSE SEND ERROR MSG
021C  4C 06 02             JMP REDRV       ;AND TRY AGAIN
021F  29 03         TAPOK  AND #3          ;CLEAR HI BITS
0221  AA                   TAX
0222  CA                   DEX             ;CONVERT TO INTERNAL FORMAT
```

The AIM 65 provides the user with a flexible and reliable tape storage system. AIM records data on tape in 80-byte blocks. A program sends data to the AIM firmware a character at a time, and then the firmware stores it in a buffer on page zero or one. When the buffer is full it is automatically written to the tape. If the remote control feature is connected, the tape recorder is started and stopped as necessary.

AIM can control two recorders at once; one for input and one for output. This makes it easy to read data from one recorder and write data to another at the same time. It also makes it simple to write a program to copy tapes a character at a time. But there is no consistent way to detect the end of the input file (EOF). The AIM routines detect EOF from the data itself. The editor uses a null line; BASIC uses a control-Z ($1A); and the binary memory dump/load routines use a zero length record. The user may also store his own data on tape and use yet another method to signal EOF. I wanted a program that would copy any AIM tape regardless of the type of data recorded on it. The result is TAPDUP.

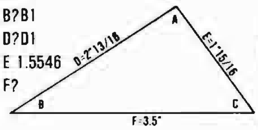
I tried to make TAPDUP as flexible as possible. There are two major reasons for copying a tape: backup of important data, and duplication of tapes for distribution. The former might require the ability to copy long multi-file tapes. In this case I wanted to detect automatically the end of the file

```
0223  8E 34 A4         STX TAPIN        ;SAVE INPUT DRIVE#
0226  F0 03            BEQ INKX         ;OUTPUT DRIVE# IS
0228  CA               DEX              ;THE OTHER ONE
0229  F0 01            BEQ STOUT
022B  E8        INKX   INX
022C  8E 35 A4  STOUT  STX TAPOUT
022F  20 13 EA         JSR CRLOW        ;NEW LINE
0232  A2 23     REGAP  LDX #GAPMSG-LITS;REQUEST FILE GAP
0234  20 FC 02         JSR KEPX
0237  20 73 E9         JSR REDOUT       ;GET REPLY
023A  C9 30            CMP #'0'         ;ALLOW 0-9
023C  90 04            BCC BADGAP
023E  C9 3A            CMP #$3A
0240  90 06            BCC GAPOK
0242  20 8E E3  BADGAP JSR CKER0        ;ERR MSG
0245  4C 32 02         JMP REGAP        ;TRY AGAIN
0248  29 0F     GAPOK  AND #$F          ;CLEAR HI BITS
024A  0A               ASL A            ;MULTIPLY BY 4
024B  0A               ASL A
024C  85 00            STA FILGAP       ;SAVE
024E  20 13 EA         JSR CRLOW        ;NEW LINE
0251  A2 40     REBLK  LDX #BLKMSG-LITS;REQUEST BLOCK COUNT
0253  20 FC 02         JSR KEPX
0256  20 AE EA         JSR ADDIN        ;GET REPLY
0259  B0 F6            BCS REBLK        ;ERROR - RETRY
025B  AD 1C A4         LDA ADDR         ;SAVE IT
025E  85 01            STA BLKCNT
0260  20 13 EA         JSR CRLOW

0263                   ;NOW READ AND WRITE BLOCKS FOREVER

0263  A9 00     BLKLUP LDA #0           ;CLEAR BLOCK COUNT
0265  8D 15 01         STA BLK
0268  20 53 ED         JSR TIBY1        ;READ A BLOCK
026B  AD 16 01         LDA TABUFF       ;GET BLOCK COUNT
026E  D0 03            BNE NOBLK0       ;SKIP IF NON-ZERO
0270  20 C2 02         JSR RDYOUT       ;READY OUTPUT DRIVE
0273  20 98 02  NOBLK0 JSR WRTBLK       ;WRITE IT
0276  A5 01            LDA BLKCNT       ;COUNTING BLOCKS?
0278  F0 E9            BEQ BLKLUP       ;NO
027A  CD 16 01         CMP TABUFF       ;IS THIS THE LAST?
027D  D0 E4            BNE BLKLUP       ;NO, CONTINUE
027F  20 13 EA         JSR CRLOW        ;NEW LINE
0282  A2 33            LDX #TAPMSG-LITS;NOTIFY USER
0284  20 FC 02         JSR KEPX
0287  AD 00 A8         LDA DRB          ;TURN ON TAPES
028A  09 30            ORA #$30
028C  8D 00 A8         STA DRB
028F  20 73 E9         JSR REDOUT       ;WAIT FOR SIGNAL
0292  20 13 EA         JSR CRLOW        ;THEN RESUME COPY
0295  4C 63 02         JMP BLKLUP       ;DO IT AGAIN

0298                   ;   WRTBLK : WRITE BLOCK TO TAPE

0298  20 E7 F1  WRTBLK JSR BKCKSM       ;COMPUTE CHECKSUM
029B  20 1D F2         JSR TAOSET       ;START DRIVE
029E  A9 23            LDA #'#'         ;CHAR FOR BEGINNING OF BLOCK
02A0  20 4A F2         JSR OUTTAP
02A3  20 D2 F1  TABY2  JSR CKBUFF       ;GET A CHAR FROM BUFFER
02A6  20 4A F2         JSR OUTTAP       ;SEND IT
02A9  E8               INX
02AA  E0 53            CPX #83          ;2 BLK CKSUM CHARS +1 CHAR
02AC  D0 F5            BNE TABY2
02AE  AD 00 A8         LDA DRB
02B1  29 CF            AND #$CF         ;TURN OFF TAPES
02B3  8D 00 A8         STA DRB
02B6  58               CLI              ;ENABLE INTERRUPT
02B7  A9 00            LDA #0
02B9  8D 37 A4         STA TAPTR2       ;CLEAR TAPE BUFF PTR
02BC  A9 00            LDA #0           ;RESET FREE RUNNING TO ONE SHOT
02BE  8D 0B A8         STA ACR
02C1  60               RTS

02C2                   ;   RDYOUT : TURN ON OUTPUT DRIVE AND WAIT
02C2                   ;            FOR CHARACTER FROM KEYBOARD
02C2                   ;            OR WAIT FOR DESIRED SECONDS

02C2  20 EA 02  RDYOUT JSR TOGOUT       ;ON DRIVE
02C5  A6 00            LDX FILGAP       ;ANY FILE GAP?
02C7  D0 11            BNE RETIME       ;YES, GO TO DELAY LOOP
02C9  20 13 EA         JSR CRLOW        ;NEW LINE
```

and insert a gap between files. The latter might require detecting the end of a tape and allowing the rewinding of the input tape and the inserting of a new output tape. TAPDUP attempts to fill both of these needs. The biggest challenge was to find a consistent way to detect EOF. The key to this turns out to be the AIM tape block count.

Each block in an AIM tape file contains a one-byte binary block number. The blocks within a file are numbered sequentially starting with zero. This block number is shown in the AIM display during reading or writing of a tape file. As long as a file has fewer than 256 blocks, each block in a file has a unique block number. If a file exceeds 256 blocks, the block number wraps to zero on the 257th block. TAPDUP uses the block number in a couple of different ways to detect EOF.

TAPDUP has three modes of operation. In the first mode, it automatically inserts a user-specified gap before writing any block number zero. This allows automatic spacing between files or automatic skip over the tape leader. The second mode is manual spacing mode. Here, TAPDUP stops before writing any block number zero, turns on the output recorder so the user can manually insert the desired spacing, and waits for a character from the



keyboard before continuing. The third mode is block count mode. In this mode the user specifies the number of the last block to be copied. After writing the specified block, TAPDUP stops and turns on both recorders, and waits for a character from the keyboard.

Operation

First turn off both recorders with the AIM 1 and 2 commands. Then ready both recorders by pressing play or record as desired. Execute TAPDUP at $200; TAPDUP prompts for all necessary information. The first prompt is 'INPUT DRIVE# = '. Enter the number of the input drive, 1 or 2 — the output drive is assumed to be the other one. The next prompt is 'INTER-FILE GAP = '. Enter the number of seconds to be inserted in front of each block zero. A single digit from 0 through 9 is allowed. If 0 is entered, manual spacing mode is assumed. If more than 9 seconds of spacing is needed TAPDUP could be modified to allow a larger number. Finally the prompt '# OF LAST BLOCK = ' is issued. Enter a one- or two-digit hex number of the last block to be copied followed by a return. Zero indicates that block count mode is

```
02CC  A2 00              LDX #RDYMSG-LITS;REQUEST DRIVE READY
02CE  20 FC 02           JSR KEPX
02D1  20 73 E9           JSR REDOUT       ;GET REPLY
02D4  20 13 EA           JSR CRLOW        ;NEW LINE
02D7  4C EA 02           JMP TOGOUT       ;GO TURN ON DRIVE

02DA  A9 F4       RETIME LDA #$F4         ;ABOUT A QUARTER SECOND
02DC  8D 97 A4           STA DI1024       ;INTO TIMER
02DF  8D 97 A4           STA DI1024       ;AGAIN IN CASE IT'S NOT LISTENING
02E2  2C 85 A4    WTIME  BIT RINT         ;TIME UP?
02E5  10 FB              BPL WTIME        ;NOPE, WAIT
02E7  CA                 DEX              ;COUNT
02E8  D0 F0              BNE RETIME       ;UNTIL ZERO

02EA              ;    TOGOUT : TOGGLE OUTPUT DRIVE LINE

02EA  AD 35 A4    TOGOUT LDA TAPOUT       ;GET OUTPUT DRIVE
02ED  F0 04              BEQ TOG1         ;BRANCH IF DRIVE #1
02EF  A9 20              LDA #$20         ;DRIVE 2
02F1  D0 02              BNE TOGIT
02F3  A9 10       TOG1   LDA #$10         ;DRIVE 1
02F5  4D 00 A8    TOGIT  EOR DRB
02F8  8D 00 A8           STA DRB          ;TOGGLE
02FB  60                 RTS

02FC              ;    KEPX ;MESSAGE WRITER

02FC  BD 08 03    KEPX   LDA LITS,X       ;GET A CHARACTER
02FF  F0 06              BEQ KEPDUN       ;QUIT ON NULL
0301  20 7A E9           JSR OUTPUT       ;SEND IT
0304  E8                 INX
0305  D0 F5              BNE KEPX
0307  60          KEPDUN RTS
0308                     .OPT GEN
0308              LITS   =*
0308  4F 55       RDYMSG .BYTE 'OUTPUT DRIVE READY?',BELL,0
030A  54 50
030C  55 54
030E  20 44
0310  52 49
0312  56 45
0314  20 52
0316  45 41
0318  44 59 3F
031B  07
031C  00
031D  49 4E       DRVMSG .BYTE 'INPUT DRIVE#=',0
031F  50 55
0321  54 20
0323  44 52
0325  49 56
0327  45 23 3D
032A  00
032B  49 4E       GAPMSG .BYTE 'INTER-FILE GAP=',0
032D  54 45
032F  52 20
0331  46 49
0333  4C 45
0335  20 47
0337  41 50 3D
033A  00
033B  52 45       TAPMSG .BYTE 'READY TAPES',BELL,0
033D  41 44
033F  59 20
0341  54 41
0343  50 45 53
0346  07
0347  00
0348  23 20       BLKMSG .BYTE '# OF LAST BLOCK',0
034A  4F 46
034C  20 4C
034E  41 53
0350  54 20
0352  42 4C
0354  4F 43 4B
0357  00

0358                     .END
```

keyboard before continuing. The third mode is block count mode. In this mode the user specifies the number of the last block to be copied. After writing the specified block, TAPDUP stops and turns on both recorders, and waits for a character from the keyboard.

Operation

First turn off both recorders with the AIM 1 and 2 commands. Then ready both recorders by pressing play or record as desired. Execute TAPDUP at $200; TAPDUP prompts for all necessary information. The first prompt is 'INPUT DRIVE# = '. Enter the number of the input drive, 1 or 2 — the output drive is assumed to be the other one. The next prompt is 'INTER-FILE GAP = '. Enter the number of seconds to be inserted in front of each block zero. A single digit from 0 through 9 is allowed. If 0 is entered, manual spacing mode is assumed. If more than 9 seconds of spacing is needed TAPDUP could be modified to allow a larger number. Finally the prompt '# OF LAST BLOCK = ' is issued. Enter a one- or two-digit hex number of the last block to be copied followed by a return. Zero indicates that block count mode is

```
02CC  A2 00               LDX #RDYMSG-LITS;REQUEST DRIVE READY
02CE  20 FC 02            JSR KEPX
02D1  20 73 E9            JSR REDOUT       ;GET REPLY
02D4  20 13 EA            JSR CRLOW        ;NEW LINE
02D7  4C EA 02            JMP TOGOUT       ;GO TURN ON DRIVE

02DA  A9 F4       RETIME LDA #$F4          ;ABOUT A QUARTER SECOND
02DC  8D 97 A4           STA DI1024        ;INTO TIMER
02DF  8D 97 A4           STA DI1024        ;AGAIN IN CASE IT'S NOT LISTENING
02E2  2C 85 A4    WTIME  BIT RINT          ;TIME UP?
02E5  10 FB              BPL WTIME         ;NOPE, WAIT
02E7  CA                 DEX               ;COUNT
02E8  D0 F0              BNE RETIME        ;UNTIL ZERO

02EA              ;    TOGOUT : TOGGLE OUTPUT DRIVE LINE

02EA  AD 35 A4    TOGOUT LDA TAPOUT        ;GET OUTPUT DRIVE
02ED  F0 04              BEQ TOG1          ;BRANCH IF DRIVE #1
02EF  A9 20              LDA #$20          ;DRIVE 2
02F1  D0 02              BNE TOGIT
02F3  A9 10       TOG1   LDA #$10          ;DRIVE 1
02F5  4D 00 A8    TOGIT  EOR DRB
02F8  8D 00 A8           STA DRB           ;TOGGLE
02FB  60                 RTS

02FC              ;    KEPX :MESSAGE WRITER

02FC  BD 08 03    KEPX   LDA LITS,X        ;GET A CHARACTER
02FF  F0 06              BEQ KEPDUN        ;QUIT ON NULL
0301  20 7A E9           JSR OUTPUT        ;SEND IT
0304  E8                 INX
0305  D0 F5              BNE KEPX
0307  60          KEPDUN RTS
0308                     .OPT GEN
0308              LITS   =*
0308  4F 55       RDYMSG .BYTE 'OUTPUT DRIVE READY?',BELL,0
030A  54 50
030C  55 54
030E  20 44
0310  52 49
0312  56 45
0314  20 52
0316  45 41
0318  44 59 3F
031B  07
031C  00
031D  49 4E       DRVMSG .BYTE 'INPUT DRIVE#=',0
031F  50 55
0321  54 20
0323  44 52
0325  49 56
0327  45 23 3D
032A  00
032B  49 4E       GAPMSG .BYTE 'INTER-FILE GAP=',0
032D  54 45
032F  52 2D
0331  46 49
0333  4C 45
0335  20 47
0337  41 50 3D
033A  00
033B  52 45       TAPMSG .BYTE 'READY TAPES',BELL,0
033D  41 44
033F  59 20
0341  54 41
0343  50 45 53
0346  07
0347  00
0348  23 20       BLKMSG .BYTE '# OF LAST BLOCK',0
034A  4F 46
034C  20 4C
034E  41 53
0350  54 20
0352  42 4C
0354  4F 43 4B
0357  00

0358                     .END
```

not to be used. The number of the last block in a file may be determined by examining memory location $116 after reading or writing the file. After writing a file, $116 contains the block number of the last block plus 1. After reading a file, $116 contains the actual number of the last block. Block count mode may be used with either automatic or manual spacing mode.

Next, the input recorder will run as TAPDUP searches for the first data block on the tape. When the first block has been read in, TAPDUP will either automatically space the output tape if a non-zero gap was specified, or display the message 'OUTPUT DRIVE READY?' and wait for a character from the keyboard if manual mode was requested. In the later case you should position the output recorder as desired and enter any character except escape. Then the input and output recorders will alternately run as each block of data is copied. The block numbers are shown in the AIM display as usual. If block count mode is used, TAPDUP will stop after writing the specified block, display the message 'READY TAPES', and wait for input from the keyboard. Both recorders are turned on so you can position them as desired and resume operation by typing any character except escape. TAPDUP never terminates and must be stopped with the reset button or by entering escape in response to one of the ready messages.

I use TAPDUP most often to make multiple copies of a tape for distribution. To do this I use automatic spacing mode with block count mode. I first put the master tape in the input recorder and a blank tape in the output recorder and start TAPDUP. When the file has been copied, the 'READY TAPES' message appears. I then rewind both tapes, put away the master tape, move the newly recorded tape to the input drive, and put a new blank tape into the output drive. I then start both drives and enter a character to notify TAPDUP to continue. By using the previously recorded tape as input, I verify each tape at no extra cost in time. I inserted an ASCII bell character in both of the ready messages to cause my terminal to beep when TAPDUP needs attention. This allows me to busy myself with other tasks while copying tapes. Unfortunately the bell character has no effect if you are using only the AIM display. TAPDUP sets the AIM inter-block gap value at $A409 to $20 so that I do not have to remember to do it. This larger-than-normal gap is required to allow time for starting and stopping the tape between blocks when using AIM's remote control feature.

Errors

If an invalid response is entered to any of the prompts, the standard AIM 'ERROR' message is displayed and the prompt re-issued. If an error is encountered while reading the tape, the 'ERROR' message is displayed and control returns to the AIM monitor.

Joel Swank may be contacted at 25730 Beach Dr., Rockaway, OR 97136.

Adventure of the Month

The Adventure is Waiting for You........

How would you like to go back in time to 19th century London to match wits with Jack the Ripper? Out into space to brave the swirling vortex of a black hole? Into the depths of the ocean, or on a quest to rescue a beautiful princess from the clutches of evil monsters?

You never know where **SoftSide Magazine's Adventure of the Month** might take you. But you can be sure that each month you will experience new delights and new challenges as you receive an original adventure on tape or disk, ready to load into your computer.

The cost? A six-month membership is only $29 for the adventures on tape ($4.83 each) or $49 on disk ($8.16 each). If you're not sure you can handle six full months of excitement, you can order a single adventure on tape for $7 or on disk for $10. You can choose from:

Arabian Adventure
Alien Adventure
Treasure Island Adventure
Jack the Ripper Adventure
Crime Adventure
Black Hole Adventure
Windsloe Mansion Adventure
Klondike Adventure
James Brand Adventure
Witches' Brew Adventure
Around the World in 80 Days

To order, use coupon provided or write to:

Adventure of the Month
6 South Street
Milford, NH 03055

ADVENTURE OF THE MONTH

■ **Six month subscription:**
□ Cassette — $29
□ Disk — $49

■ **Individual adventures (please specify)**

____

____

□ Cassette — $7 each
□ Disk — $10 each

■ **Three adventures on one super disk ($26 each):**
□ Arabian, Alien, and Treasure Island Adventures
□ Jack the Ripper, Crime, and Around the World Adventures
□ Black Hole, Windsloe Mansion, and Klondike Adventures

Please specify which computer:
□ Apple™ (req. 24K for tape, 32K for disk)
□ ATARI® (req. 32K for tape, 40K for disk)
□ TRS-80® (req. 16K for tape, 32K for disk)

Name ______________________________
Address ______________________________
City/State ____________________Zip__________

□ MasterCard □ VISA □ Payment enclosed
Name of Cardholder ______________________
MC# and Interbank#/VISA#________________
Exp. Date__________Signature______________

Prices subject to change without notice. Apple™, ATARI® and TRS-80® are registered trademarks of The Apple Computer Company, Warner Communications and The Tandy Corporation respectively.

MICRO™
Hardware Catalog

Name: **Joyport**
System: Apple
Language: Compatible with BASIC, Pascal, and Machine Language
Description: Apple Computer input device that allows you to use four Apple game paddles or two Atari joysticks. Easily accessible connectors eliminate need to open Apple case. Two switches to select between Apple-type paddles and Atari joysticks.
Price: $74.95
Includes one copy of Sirius Software's Computer *Foosball* and complete instructions
Available:
Sirius Software, Inc.
10364 Rockingham Dr.
Sacramento, CA 95827

Name: **Modem Driver Module Kit — MDM-2**
System: VIC-20
Memory: 5K
Language: BASIC
Hardware: RS-232-C Modem
Description: Kit for driving inexpensive modems, or most serial printers. Includes V-Term-20 terminal program that converts VIC-20 to a terminal with escape key, break key and control characters.
Price: $29.00 kit
$35.00 assembled and tested. Includes V-Term-20 terminal program.
Available:
RVR Systems
P.O. Box 265
Dewitt, NY 13214

Name: **SADI**
System: PET/CBM Computers
Language: BASIC
Hardware: Bi-directional Printer Adapter
Description: The CmC SADI communications controller is a microprocessor-based peripheral device for the Commodore PET and CBM computers allowing you to connect your computer to parallel and serial printers, CRTs, modems, accoustic couplers, hard copy terminals and other computers. Some other features are the RS-232 serial in and out, parallel printer output, Centronics-compatible, true ASCII conversion, 32-character buffer with X-off/x-on feature, 75 to 9600 baud and allows transfer of programs between PETs.
Price: $295.00
Includes case, PET IEEE cable and power supply.
Available:
Connecticut microComputer
36 Del Mar Dr.
Brookfield, CT 06804
(203) 775-4595

Name: **Station Master**
System: 48K Apple II or Apple II Plus
Memory: 16K
Language: Compatible with BASIC, Pascal 1.1, and CP/M
Hardware: One of the parallel printers listed below
Description: Universal parallel interface card with graphics on board. Allows user to dump hi-res screen by easy keyboard commands. Features include: dumping page 1 or 2, normal or expanded size, picture or plot and horizontal positioning. Printer required: Epson MX-80 with graphics, Epson MX-100, Anadex 9501/9500, Data South DS180, Centronics 739 or NEC PC8023.
Price: $175.00
Includes card, cable and practice pictures on diskette.
Available:
Computer Station
11610 Page Service Dr.
St. Louis, MO 63141
or your local dealer

Name: **8510 pro/writer printer**
Description: 120 CPS, tractor and friction feed, graphics and incremental printing. Proportional spacing (N×9), pica and elite compressed (9×7), character generator (8×8). Uni- and bi-directional compressible to 136 columns. Interface: parallel 8-bit Centronics or serial RS-232C with switch selections for x-on/x-off protocol.
Includes languages: U.S., U.K., Japanese, Swedish, German — in ROM.
Available:
Leading Edge Products, Inc.
225 Turnpike St.
Canton, MA 02021

Name: **Digibit**
Hardware: Digitizer, cursor, RS-232C interface power supply cable for operation at 300 baud
Description: The *Digibit* is a light-weight, self-contained digitizer for graphics analysis. It is compact with a working area of 11"×17" and a 0.01 resolution. It digitizes in either point to point or stream mode on any surface or angle including a CRT screen. It fits any system, converting graphic images into numeral values for the computer.
Price: $520.00 complete
Available:
NUMONICS
418 Pierce St.
Lansdale, PA 19446
(215) 362-2766

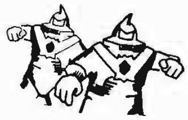
Name: **GIMIX Multiuser**
System: 6809 Winchester System
Memory: 120KB
Language: BASIC09, Pascal, CIS, COBOL, C
Description: GIMIX's 6809 system supports up to four terminals and features a 2MHz 6809 CPU, 120KB of static RAM, a 19MB (unformatted) 5¼" Winchester hard disk, a 1MB (unformatted) 5¼" floppy disk, and four serial I/O ports. Memory is expandable up to 632KB. Additional memory, mass storage capacity, and I/O for additional terminals and peripherals are optional. The system can select between two operating systems, under software control, making it useful for software development. The price includes OS-9 level 2, a UNIX-like multi-user, multi-tasking operating system and the OS-9 debugger, text editor, and assembler, the GMXBUG/FLEX monitor/operating system combination, and a single-user (56KB) operating system, capable of running any software written for FLEX.
Price: $8998.09
Available:
GIMIX, Inc.
1337 West 37th Place
Chicago, IL 60609

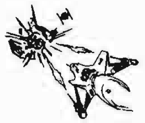
Name: **Parallel Printer Interface**
System: OSI, or any system with a 6850 ACIA
Hardware: Single printed circuit board
Description: This small PCB converts any OSI serial port to a centronics parallel printer port. Absolutely no software changes, no tracks to cut. Just unplug your 6850 ACIA and plug in the PCB.
Price: $9.95 PCB
$45.95 assembled and tested (US including P&P)
Includes PCB, either bare or assembled and tested, full instructions, one-year warranty (assembled and tested only).
Available:
G. Cohen
72 Spofforth St.
Holt, Act, 2615
Australia

Name: **ADA 1450**
System: PET/CBM Computers
Language: BASIC
Hardware: Printer Adapter
Description: A serial interface that allows PET/CBM computers to use standard serial printers. The ADA 1450 is addressable and set to work with ASCII-coded printers. It has a two-foot cable which plugs into the PET IEEE port. Another IEEE card edge connector is provided to connect other peripherals. The address is switch selectable for upper and lower case.
Price: $149.00
Available:
Connecticut microComputer
36 Del Mar Dr.
Brookfield, CT 06804
(203) 775-4595

Answers to June Crossword Puzzle

1 C	A	2 R	R	Y	■	3 A	4 D	■	5 V	E	R	6 I	F	7 Y	■
O	■	E	■	■	8 C	■	A	■	A	■	■	N	■	E	■
9 N	U	L	L	■	10 O	C	T	A	L	■	11 F	I	R	S	12 T
C	■	A	■	■	L	■	A	■	■	■	■	T	■	■	R
A	■	13 T	A	14 N	D	Y	■	15 B	N	E	■	16 I	N	C	A
T	■	I	■	E	■	■	■	I	■	■	■	A	■	■	N
E	■	V	■	X	■	17 M	O	T	O	18 R	O	L	A	■	S
N	■	E	■	T	■	■	■	■	■	E	■	I	■	■	A
A	■	■	■	■	19 B	■	■	■	■	A	■	20 Z	I	N	C
21 T	A	X	■	22 D	E	23 C	I	M	A	L	■	E	■	■	T
E	■	■	■	I	■	L	■	■	■	■	24 R	■	■	25 T	O
■	26 F	O	U	R	T	E	E	N	■	■	27 E	R	R	O	R
■	A	■	■	E	■	A	■	■	■	■	T	■	■	G	■
28 I	C	■	■	29 C	U	R	30 S	O	31 R	■	U	■	■	G	■
■	32 E	X	33 I	T	■	■	B	■	T	■	34 R	35 E	P	L	Y
■	■	■	F	■	■	■	36 C	O	S	I	N	E	■	E	■





MICRO Short Subjects

Interactive Random Generator *by Harry White*

Left of Equal Sign MID$ in Applesoft *by Gustavo Criscuolo*

Interactive Random Generator

Harry White, 7495 West 81st Avenue,
Arvada, Colorado 80003

The Applesoft RND function produces a series of tables of "random" numbers. The particular table depends on what negative number is used in RND to initialize the series. If you are using a turnkey system or running from a cold start, you get the same series of random numbers every time. Random, perhaps, but predictable.

Listing 1 solves the problem. It takes advantage of the fact that decimal memory locations 78 and 79 (hex $4E and $4F) contain a 16-bit number that is constantly being incremented at microprocessor speed as long as the machine is awaiting input. You cannot predict (in real time) what number is in that two-byte location while the flashing cursor is on the screen.

In listing 1, the input statement is needed not only to determine the desired parameters of the random range, but to interrupt the incrementing at locations 78 and 79. This lets the PEEK statements determine what was there when the interruption occurred. Hence, the "interactive" part of the title.

The program has its limitations. You must press some keys to get a number. Its range is limited to 1 through 65536 but it is somewhat flexible; if you get the rhythm just right and are quick enough, you can create a discernible pattern in the series of integers returned. However, if you just *use* it, it works fine.

You will find this program most useful in applications where you need to sample small quantities from varying ranges of parameters — quality control, spot checks, time studies, etc. Generating long tables of random numbers requires too much key punching. If, however, your upper range limit is consistent during a run of the program, line 190 can be changed to read: IF A$ = CHR$(32) THEN GOTO 140. Then, pressing the space bar will give you a different number within your limits each time.

The important line in the routine is number 140, which PEEKs 78 and 79 and retrieves the integer there. The input variable, A, is used in the formula to fit the random number within the limit set by A. The result is pseudo random but less predictable than Applesoft RND.

Listing 1

```
100  TEXT : HOME
110  PRINT  SPC( 4)"THE INTERACTIVE RANDOM GENERATOR"
120  FOR N = 1 TO 40: PRINT "*";: NEXT : PRINT
130  VTAB 4: INPUT "RANGE FROM 1 TO....? ";A
140 R =  INT ((( PEEK (79) * 256 +  PEEK (78)) / 65536) * A) + 1
150  VTAB 8: HTAB 15: PRINT "** ";R;" **
160  VTAB 16: PRINT "PLEASE PRESS SPACE BAR TO CONTINUE"
170  PRINT "OR ANDY OTHER KEY TO END."
180  GET A$
190  IF A$ =  CHR$ (32) THEN  VTAB 4: CALL  - 958: GOTO 130
200  HOME : VTAB 10: HTAB 18: PRINT "BYE !": END
```

Left of Equal Sign MID$ in Applesoft

Gustavo Criscuolo, Instituto Universitario, Pedagogico Experimental de Maturin, Venezuela

This utility will make life easy when you work with strings. It replaces a part of a string with another string by turning the original one into substrings (by means of the MID$ function) and concatenating. It takes a long time and a lot of memory to concatenate. Sooner or later memory becomes full of nonuseful strings and the garbage collection process spends time moving strings to the top of RAM.

Other BASIC implementations have a "left of equal sign" MID$ function. This function takes a string and puts it into the middle of another string. No more room is used; only some positions of RAM change their contents.

To extend Applesoft BASIC, the ampersand technique is used. Extensive use of Applesoft subroutines is made to get a short machine-language program but no attempt is made to optimize the code.

The program is loaded at $958B (decimal 38283) and is only $75 bytes long. It must be loaded with the proper disk command, protected by a HIMEM: 38283, and initialized using a CALL 38283. It works with any type of string, including subscripted ones. The general calling procedure is:

& MID$(A$(K),I,J) = B$(M)

with the following meanings:

A$ the string where the replacement takes place

K if A$ is a subscripted string, its subscript

I start of replacement

J number of characters to replace

B$ the string where the characters come from

M if B$ is a subscripted string, its subscript

(Continued on next page)

The A$ and B$ strings both can be double subscripted. No attempt was made to check for the conditions

```
J< =LEN(B$(M))
I+J< =LEN(A$(K))
```

Also, if A$ points to a string in the program text, it will be changed. With this trick, you can code a program that tells you how many times it has been run.

Mark and Release Procedures in Applesoft

In languages such as Pascal, there is a concept of a Heap. It is a part of memory where it is possible to define some variables and to dispose of them when they are no longer needed. It is possible to implement a similar technique in BASIC, using strings, and realizing that the pointer to the bottom of string storage is the zero page locations $6F,$70 (111,112 decimal).

First select some strings that will be the permanently allocated ones. Define them in the program, taking into account that you must force the string to be allocated in the top of RAM (e.g., use A$="$$$$"+"$"). Then save the pointer; you can use

```
IL=PEEK(111):IH=PEEK(112)
```

Now you can define other strings (e.g. in a subroutine). When the work is over and these strings are no longer useful a simple

```
POKE 111,IL:POKE 112,IH
```

will release the memory for future use.

You must take care not to activate the garbage collection procedure. Garbage collection is activated if the pointer to the end of numeric storage in zero page locations $6D,$6E (decimal 109,110) collides with the pointer to the bottom of string storage.

MICRO

***Listing 1:* Left/MID$**

```
003E          PT2      EPZ $3E
0071          FRESPC   EPZ $71                ;TEMPORARY POINTER FOR STRING
STORAGE ROUTINES
007B          PT3      EPZ $7B
0083          VARPNT   EPZ $83                ;POINTER TO VARIABLES DESCRIPT
OR(POINTS TO LENGTH)
00AD          STRG2    EPZ $AD                ;POINTER TO A STRING
00D0          EQUALS   EPZ $D0                ;TOKEN FOR EQUALS SIGN
00EA          MID$     EPZ $EA                ;TOKEN FOR MID$ COMMAND
00B1          CHRGET   EPZ $B1
00B7          CHRGOT   EPZ $B7
DFE3          PTRGET   EQU $DFE3              ;FIND VARPNT
D97C          UNDEF    EQU $D97C              ;DISPLAYS UNDEF STATEMENT ERRO
R
DEBB          CHKOPN   EQU $DEBB              ;CHECKS FOR "("
DEB8          CHKCLS   EQU $DEB8              ;CHECKS FOR ")"
DEBE          CHKCOM   EQU $DEBE              ;CHECKS FOR ","
DEC0          SYNCHR   EQU $DEC0              ;DISPLAYS ERROR IF CHARACTER
0800          ;POINTED TO BY TXTPTR <> ACCUMULATOR
E5E2          MOVSTR   EQU $E5E2              ;APPLESOFT'S STRING MOVER
E6F8          GETBYT   EQU $E6F8              ;EVALUATES FORMULA POINTED TO
0800          ;BY TXTPTR AND STORES THE RESULT IN FAC
00A1          FACLO    EPZ $A1
0800          *
958B                   ORG $958B
958B                   OBJ $800
958B          *
958B          *
958B A9 9B    TIMP     LDA #START             ;SET UP AMPERSAND
958D 8D F6 03 TEMP     STA $3F6               ;VECTOR ADDRESS
9590 A9 95             LDA /START
9592 8D F7 03          STA $3F7
9595 A9 60             LDA #$60               ;DUMMY RTS
9597 8D 8B 95          STA TIMP               ;AT TIMP
959A 60                RTS
959B 20 B7 00 START    JSR CHRGOT             ;GET TOKEN
959E C9 EA             CMP #MID$              ;IF MID$ TOKEN
95A0 F0 03             BEQ UNO                ;THEN CONTINUE
95A2 4C 7C D9          JMP UNDEF              ;ELSE ERROR
95A5 20 B1 00 UNO      JSR CHRGET             ;GET NEXT BYTE
95A8 20 BB DE          JSR CHKOPN             ;CHECK FOR "("
95AB 20 E3 DF          JSR PTRGET             ;GET DESCRIPTOR ADDRESS FOR PA
SSED VARIABLE

95AE A0 01             LDY #$1                ;GET ADDRESS OF TARGET STRING
95B0 B1 83             LDA (VARPNT),Y
95B2 8D 8D 95          STA TEMP               ;AND SAVE AT TEMP
95B5 C8                INY
95B6 B1 83             LDA (VARPNT),Y
95B8 8D 8E 95          STA TEMP+1
95BB 20 BE DE          JSR CHKCOM             ;CHECK FOR COMMA
95BE 20 F8 E6          JSR GETBYT             ;GET START OF REPLACEMENT
95C1 A5 A1             LDA FACLO
95C3 85 3E             STA PT2                ;AND STORE AT PT2
95C5 20 BE DE          JSR CHKCOM             ;CHECK FOR COMMA
95C8 20 F8 E6          JSR GETBYT             ;GET LENGTH OF REPLACEMENT
95CB A5 A1             LDA FACLO
95CD 85 7B             STA PT3                ;AND STORE AT PT3
95CF 20 B8 DE          JSR CHKCLS             ;CHECK FOR ")"
95D2 A9 D0             LDA #EQUALS            ;NEXT SHOULD BE EQUALS SIGN
95D4 20 C0 DE          JSR SYNCHR             ;ELSE SYNTAX ERROR
95D7 20 E3 DF          JSR PTRGET             ;GET ADDRESS OF SOURCE STRING
95DA A0 01             LDY #$1
95DC B1 83             LDA (VARPNT),Y
95DE 85 AD             STA STRG2              ;AND STORE AT STRG2
95E0 C8                INY
95E1 B1 83             LDA (VARPNT),Y
95E3 85 AE             STA STRG2+1
95E5 C6 3E             DEC PT2
95E7 AD 8D 95          LDA TEMP               ;RECOVER ADDRESS OF TARGET STR
ING


95EA 18                CLC
95EB 65 3E             ADC PT2                ;ADD TO START OF REPLACEMENT
95ED 85 71             STA FRESPC             ;AND STORE AT FRESPC
95EF AD 8E 95          LDA TEMP+1
95F2 69 00             ADC #$0
95F4 85 72             STA FRESPC+1
95F6 A4 AE             LDY STRG2+1            ;LOAD Y WITH SOURCE ADDRESS(HI
GH)
95F8 A6 AD             LDX STRG2              ;LOAD X WITH SOURCE ADDRESS(LO
W)
95FA A5 7B             LDA PT3                ;LOAD ACCUM W/LENGTH OF REPLAC
EMENT
95FC 20 E2 E5          JSR MOVSTR             ;DO MOVE
95FF 60                RTS
9600                   END
```

MICRO™

Dr. William R. Dial
438 Roslyn Avenue
Akron, OH 44320

6502 Bibliography

1.Creative Computing 8, No. 2 (February, 1982)

Haley, Kenneth M., "Picture Packer Revisited," pg. 116-124.

A real advancement in the technique of storage and fast loading of Apple hi-res picture screens. It is possible to have 20-40 pictures stored on a single Apple disk side.

2. The Apple Barrel 5, No. 1 (February. 1982)

Kramer, Mike, "EXEC Files on the Apple II," pg. 14-17.

Using EXEC files which can contain either lines of BASIC program code or a sequence of keyboard commands. Useful in setting up an automatic operation mode.

3. The Michigan Apple-Gram 4, No. 2 (February, 1982)

Thomka, Chuck, "Mods for the Apple Two-Piece Keyboard," pg. 22-25.

Detailed information on the Apple keyboard and how to take advantage of its features.

4. Computing Today 3, No. 12 (February 1982)

Smith, Bruce F., "Graf-Rite," pg. 73-75.

Define and store your own characters in the 6502-based British microcomputer, Acorn Atom. Ideal for labeling graphs or annotating diagrams.

5. Interactive Issue No. 7 (January, 1982)

Hance, Joe, "Centronics-Type Printer Driver," pg. 7.

An interface to use an Epson MX-80 printer with an AIM 65. Gives hardware connections and a brief machine-language routine.

6. MICRO No. 45 (February, 1982)

Flynn, Christopher J., "Formatting AIM Assembler Listings: A PL/65 Approach," pg. 19-26.

A program to reformat AIM assembler listings. The new listings are much easier to read than the standard 20-column assembler format.

7. Sym-Physis 2, No. 4, Issue No. 10 (Oct/Nov/Dec, 1981)

Anon., "How to Power-Up into a Running BASIC Program," pg. 2-5.

A utility routine for the SYM to implement a turnkey system, including protection provisions.

8. PEEK(65) 3, No. 2 (February, 1982)

Manley, D.R., "Two Random Access Files," pg. 9-13.

How to convert OSI BASIC to use Device #& as a random access file device, just like #6.

9. The Aardvark Journal 2, No. 6 (February, 1982)

Windes, Stanley, "OSI and the Shugart SA4800," pg. 2-5.

How to implement the 5¼-inch drive on the Superboard C1P from OSI. Includes power supply, data separator, interface board, controller to disk connections, and HEXDOS and OSI 65D software.

10. Compute! 4, No. 2 (February, 1982)

Macnaughton, Robert, "Measure Time Intervals with the PET Parallel User Port," pg. 160-165.

A machine-language program that can be used on the PET to measure seven successive small time intervals, using the CBM parallel user port and eight phototransistors.

MICRO

MICRO™

Software Catalog

Name: **Atmona-1**
Machine-Language Monitor
System: Atari 400/800
Memory: 16K RAM
Language: 6502 machine language
Hardware: Atari 400/800
Description: This monitor provides you with the firmware support that you need to get the most out of your powerful system. *Atmona-1* comes on a bootable cassette. No cartridges required. Disassemble, memory dump hex plus ASCII, change memory locations, blocktransfer, fill memory block, save and load machine-language programs, start machine language programs. (Printer optional.) Comes with introductory article on how to program the Atari computer in machine language. (Available also in ROM.)
Price: $19.95
Includes description and cassette
Available:
Elcomp Publishing, Inc.
53 Redrock Lane
Pomona, CA 91766
(714) 623-8314

Name: **Apple Flasher**
System: Apple II with Applesoft, DOS 3.3; Apple II Plus, DOS 3.3; Apple III (Emulation Mode)
Memory: 48K
Language: Assembly, Applesoft
Hardware: Disk II, paddles optional
Description: Flash graphics files directly to your TV screen as pictures at incredible speed with *Apple Flasher*. The program bypasses ordinary DOS routines in order to display files as pictures in about 1.5 seconds each. Display modes include: 1) single key selection of any file on disk, 2) continuous scan of all files on disk with new picture on screen every 1.5 seconds, 3) carousel projector simulation controlled by either of the game controllers (or the keyboard) to display screens from one or two drives with instant access to *both* next and previous "slide," 4) and continuous display of all screens on one or two drives (up to 30 pictures) with individual control of display time for each picture as used in advertising displays, etc. Unlabelled disks may be searched for presence and names of hi-res screen files with two key strokes per disk at rate of five to ten seconds per disk.
Price: $34.50 plus $1.00 handling for mail orders
(NY residents add tax)
Includes diskette, 8-page manual.
Author: Paul W. Mosher
Available:
Crow Ridge Associates
P.O. Box 90
New Scotland, NY 12127
(518) 765-3620

Name: **B.C. Animation**
System: OSI C1P/Superboard
Memory: 8K minimum
Language: 8K BASIC in ROM
Hardware: Blank cassette tape
Description: *B.C. Animation* is a BASIC/machine-language hybrid. It allows the user to create pictures stored as "frames" and may call or erase any of these frames to or from anywhere on the screen. Using this program, a large group of graphics characters can be displayed, erased, or moved as fast as one character can without it. It is simple to use and is user modifiable. If you have a different screen size or memory configuration, the entire program can be modified by changing only a few lines. The machine language routines and stored picture "frames" are totally relocatable. This package includes two programs: the "Editor," the program used to create and save picture "frames;" and the "Subroutine," the machine-language routine used with your programs to display the stored frames.
Price: $15.00 plus $1.00 for postage and handling
Includes cassette tape, detailed instructions for use and possible modifications, and a software catalog.
Author: Craig Zupke
Available:
B.C. Software
9425 Victoria Dr.
Upper Marlboro, MD 20772

Name: **DOW2000 & OPTION43**
System: Apple II
Memory: 48K
Language: Applesoft
Hardware: Disk 3.3/3.2, printer optional
Description: Stock Market Analysis will determine price projections based on a stock's BETA coefficient or Relative Strength Number and the Dow Jones Average. Projections are made as you vary the DOW (What if...); on one stock or entire portfolio with single scan, quick scan, or variable scan of values. The option program will give you the percent of increase of the option months to determine which month and strike price option to buy for a given stock. Included is the booklet "The Art of Timing Your Stock's Next Move." Author in market 17 years and former Registered Investment Advisor with S.E.C.
Price: $29.95
Includes booklet (booklet alone $5.95).
Author: CIAC: Patrick and David Calabrese
Available:
Bit 'n Pieces Series
P.O. Box 7035
Erie, PA 16510

Name: **Merlin Dial/Data**
System: Apple II, Apple II Plus
Memory: 48K
Language: BASIC (Applesoft)
Hardware: Two disk drives, micro model
Description: Allows Apple user immediate access to Merlin data base which has been used by investment professionals for more than a decade. Gives daily and historical price information for all securities, options and commodities on all major exchanges. Automatic accesss and file handling. All prices are updated daily and system is Compu-trac compatible. Also available to other micro users who wish to write their own programs.
Price: $75.00 minimum—Apple software
Daily pricing service—$45.00 monthly minimum plus monthly usage charges.
Includes manual, data base creation and maintenance plus automatic access to Merlin DIAL/DATA time sharing system for prices.
Available:
Remote Computing Corp.
Dept. MS
1044 Northern BLvd.
Roslyn, NY 11576
(516) 484-4545

Name: **Paulson Package**
System: OSI C1P
Memory: 4K
Description: *Crazy Bomber* (4K graphics). You are confronted by aliens in a gigantic ship that is directly above you. The alien ship has ten bomb racks with five bombs in each rack. When most of the bombs have dropped, the racks are refilled. Your mission is to destroy the bombs before they hit the ground. Each time the racks are refilled, the bombs come down faster. You lose if 10 bombs get past you. *UFO Attack* (4K graphics). You control a killer satellite to defend against a fleet of UFO ships. The top five scores are displayed at the end of each game. Fast moving fun with excellent use of graphics. *Meteor Fallout* (4K graphics). Looks like it stepped right out of the arcade! You destroy moving meteors before they hit the surface of your planet. Each meteor falls at a different angle and speed. The meteors also make unexpected changes in direction at times and will test your skill in making decisions quickly. The graphics are excellent!
Price: $9.95 Crazy Bomber
$8.95 Meteor Fallout
$8.95 UFO Attack
Author: Thomas A. Paulson
Available:
Aurora Software Associates
37 S. Mitchell
Arlington Heights, IL 60005
(312) 259-3150

Software Catalog *(continued)*

Name: **CIPHER/K**
System: Apple, OSI, other 6502 systems
Memory: 8K
Language: BASIC and 6502 ASM
Hardware: Cassette tape, disk
Description: State of the art public key cryptographic system. Capable of serial communication. *CIPHER/K* provides a highly secure data encryption and decryption system for personal and business communication.
Author: D. Wolf, Ph.D.
Available:
D. Wolf, Ph.D.
Box 565
Port Hueneme, CA 93041

Name: **ESTHER**
An Exericse in Artificial Intelligence
System: Any FLEX-based 6809 or 6800 system
Memory: 16K minimum
Language: 6809 Assembly Language
Hardware: 6809 running FLEX
Description: *ESTHER* is one of *ELIZA's* best students. It is based on the classic MIT program. A few features have been added. *ESTHER* remembers names, drops names, uses the player's name, answers third person replies, echos keywords, and much more. The system includes more than 75 keywords and more than 48 replies. Features auto formatting according to the line length of your terminal. The program is both educational and fun. The source code will show you how to experiment with AI.
Price: $39.95 for object code
$59.95 for object and source code
Includes FLEX disk and manual.
Available:
Frank Hogg Laboratory
130 Midtown Plaza
Syracuse, NY 13210
(315) 474-7856

Name: **Machine-Language Graphics Writer**
System: OSI C1P/Superboard
Memory: 8K
Language: BASIC in ROM
Description: The program contains a draw routine. Using the polled keyboard, you can draw pictures on the screen with any of the OSI graphic characters except for the 32-dec. blank space. Once a picture is drawn, the computer will generate a machine-language program to write the same picture that is on the screen. The machine-language program is then saved on tape for later use.
Price: $9.00
Includes cassette and detailed instructions.
Author: Brian Zupke
Available:
BC Software
9425 Victoria Drive
Upper Marlboro, MD 20772

Name: **Omniware**
System: Apple II Plus
Memory: 48K
Language: BASIC
Hardware: 3.3 DOS, disk drive
Description: *Omniware* consists of "Omnifile," a full featured file manager and report generator; "Omnitrend," a powerful multiple regression trend analysis program with statistical calculation and extensive hi-res graphics; and "Omnigraph," a flexible data plotting program that allows X-Y plots, bar charts and pie charts.
Price: $129.95
Includes "Omnifile," "Omnitrend," and "Omnigraph."
Author: Keith Booker
Available:
Educational Computing Systems, Inc.
106 Fairbanks
Oak Ridge, TN 37830
(615) 483-4915

Name: **Menu Generator**
System: Apple II Plus
Memory: 48K
Language: Applesoft BASIC
Hardware: One disk drive with DOS 3.3, printer optional
Description: *Menu Generator* makes it easy for you to create custom computer menus for your Apple II. Just fill out one screen form to define each menu option and what action to take when that option is selected. Your menus can run BASIC or machine-language programs, boot another disk, execute user-written BASIC statements and perform any valid DOS operation. *Menu Generator* will compile your inputs into a neat, attractively formatted screen menu and then autmatically write a documented bug-free BASIC program to generate and process the menu.
Price: $39.95
Includes program disk, back-up disk, and 40-page manual.
Author: Robert N. Crane
Available:
Crane Software, Inc.
16835 Algonquin
Suite 611
Huntington Beach, CA 92649

Name: **Word Games**
System: Apple II 3.2 or 3.3 DOS
Memory: 48K
Language: Applesoft
Description: *Word Games* includes three games for fun and vocabulary building, for ages eight to adult. In each game the user asks for clues and makes guesses to find a related word pair. Original entries can be added.
Price: $24.95
Includes Flip Flop, Flip-E Flop-E, Code Rhyme.
Author: Mary Berry
Available:
Merry Bee Communications
815 Crest Drive
Omaha, NE 68046

VOLUME 2
now available!

NIBBLE *EXPRESS*

TABLE OF CONTE

NIBBLE EXPRESS VOLUME 2
NOW AVAILABLE!

The 1981 Anthology of the major articles and programs is underway! It contains *up-to-date enhancements and corrections to programs which appeared in Volume 2 of NIBBLE.*

Even if you have all 8 issues of NIBBLE in 1981, you'll want The Express to have updated program listings in one convenient package!

200 solidly packed pages with major programs and articles for your Apple! A MUST for your Library!

NIBBLE EXPRESS will be an invaluable reference for your Apple now and for years to come!

for only $14.95 (plus shipping) you can make Nibble Express a permanent part of your library.

ORDER YOUR COPY NOW!

NIBBLE
P.O. Box 325
Lincoln, MA 01773

Yes! I want NIBBLE EXPRESS Vol. 2 in my library!
Here's my □ Check □ Money order
for $14.95 plus $1.75 postage/handling. (Outside U.S. add $2.75 for postage/handling).

□ Also send me NIBBLE EXPRESS Vol. 1 at $12.95 plus $1.75 postage/handling (outside U.S. add $2.75 for postage/handling).

Master Card & Visa Accepted

Card # ____________________ Expires ______
Telephone ____________________
Name ____________________
Street ____________________
City ____________ State ______ Zip ______

Your check or money order must accompany your order to qualify.
Outside U.S.: Checks must be drawn on a U.S. Bank.
*Apple is a registered trademark of Apple Computer Company.

Software Catalog *(continued)*

Name: **Readtest — An English Text Analysis Program**
System: 6809 FLEX
6800 FLEX
Memory: 16K required (lower memory)
Language: 6809 assembly language
Hardware: Any 6809 system that runs FLEX
Description: *Readtest* is a powerful tool designed to help the student writer as well as the experienced writer keep a check on his readability. It is based on the readability research of Dr. Rudolf Flesch. It tells you how many words you have written, how many sentences you have used, and computes the average sentence length. It checks to see how many times you use key personal words and counts the number of names (proper nouns) in your writing. It also checks the number of affixes in your writing. It then rates your text according to its difficulty.
Price: $54.95 for object code
$74.95 for object and source code
Includes FLEX disk, manual and program
Available:
Frank Hogg Laboratory
130 Midtown Plaza
Syracuse, NY 13210
(315) 474-7856

Name: **Client Write-Up System**
System: Commodore 8032 CBM
Memory: 32K RAM
Language: Commodore 4.0 BASIC and Assembler
Hardware: 8050 dual disk drives
Description: INI's *Client Write-Up System* allows accountants to maintain the books and produce all financial statements for up to 99 different clients. The accountant can customize each client's chart of accounts, and each client's reports. (INI's system features a completely user-definable report generator — both content and format.) Powerful budget reports can be produced using a unique interface with VisiCalc.
Price: $850.00
Includes system disk, one demo client, user's manual, plus software support.
Available:
INI Inc.
4013 Chestnut St.
Philadelphia, PA 19104

Name: **Graphtrix™ 1.3**
System: Apple II
Memory: 48K
Language: Applesoft in ROM
Hardware: See description
Description: *Graphtrix 1.3* Matrix Graphics System is the latest version of Data Transforms' multi-printer graphics screen dump for the Apple II. *Graphtrix 1.3* will make hard copy of any Apple II hi-res graphics screen, and will place the graphic anywhere in the text the user wishes. *Graphtrix* Matrix Graphics System transforms the Applewriter into the most powerful text editing system available. It requires one of the following printers; Anadex 9500/9501/9000/9001, Centronics 739/122, Epson MX-100/MX-80/MX-70, IDS 440G/445G/460G/560G, ITOH 8510, MPI 88G, NEC 8023, Okidata 82A/83A, Silentype. Also required are one of the following parallel interface cards: Apple standard/centronics, CCS 7728, Epson APL, Grappler, Mountain CPS, Prometheus PRT-1/Versacard SSS-AIO, TYMAC.
Price: $65.00
Available:
Data Transforms
616 Washington St.
Denver, CO 80203

Name: **AccuRec, The Integrated Time Recorder/Wage Summary Program**
System: Apple II
Memory: 48K
Language: Applesoft
Hardware: Single disk drive, Time/Clock Interface Board, Printer
Description: *AccuRec* is an advanced attendance recorder/reporting system. Employees clock in and out; it generates a report (upon command) of the daily/weekly/total hours and gross wages (including overtime). Eliminates time-consuming conversion of time cards into paycheck. Daily/weekly records can be displayed for reference and monitoring. Also functions as a job cost recorder, recording times and computing job costs.
Price: $179.95
Includes shipping charges.
Author: Daniel J. Cassidy
Available:
Individualized Operand (Io)
Division of Cassidy Research Corp.
P.O. Box 3030
San Rafael, CA 94912
(415) 459-3383

Name: **Waterloo microBASIC**
System: Commodore SuperPET, Voker-Craig 2900, 3900, 4900, Northern Digital microWAT
Description: *Waterloo microBASIC* includes ANS Minimal BASIC, with certain minor exceptions, and several extensions such as structured programming control, long names for variables and other program entities, character-string manipulation, callable procedures and multi-line functions, sequential and relative file capabilities, integer arithmetic, debugging facilities, and convenient program entry and editing facilities.
Available:
Waterloo Computing Systems Limited
158 University Ave. W.
Waterloo, Ontario
Canada N2L 3E9

What would you give to have your Apple II able to configure to any peripheral?
SOFTECH
MICROSYSTEMS

Name: **Southern Command**
System: Apple II
Memory: 48K
Language: Applesoft in ROM
Hardware: One disk drive
Description: Battalion-level simulation of the Israeli counter-attack to cross Suez Canal during October War of 1973 against Egypt. Displayed on 28 × 39 hex grid map.
Price: $39.95
Includes diskette, rule book, map, player-aid card.
Available:
Strategic Simulations, Inc.
465 Fairchild Dr.
Suite 108
Mountain View CA 94043

Name: **Comparative Buying**
System: Apple II, Apple II Plus with Applesoft in ROM
Memory: 48K
Language: BASIC
Hardware: One disk drive, monitor or TV
Description: This new informative consumer program explains the concepts of comparative buying. The diskette subjects are: 1) concepts of comparative buying, 2) decisions before buying, 3) effective sales buying, and 4) cash buying *versus* credit buying.
Price: $165.00
Includes documentation, supportive material, four disks.
Author: Dr. Florence Taber
Available:
Interpretive Education, Inc.
157 S. Kalamazoo Mall
Suite 250
Kalamazoo, MI 49007

Name: **Capitalization**
System: Apple II, Apple II Plus, Bell and Howell Apple
Memory: 48K
Language: Applesoft
Hardware: Apple II, one disk drive (either DOS 3.2 or 3.3), printer optional
Description: Two-disk program which teaches the basic rules of capitalization. Rules are presented followd by 25 practice sentences. Upper/lower case characters are used. Teacher can add/delete/modify sentences for each practice lesson. Test disk keeps detailed records of each student's errors. Immediate feedback for each response with varied graphic reinforcers.
Price: $49.95
Includes two diskettes and teacher's guide.
Available:
Hartley Courseware, Inc.
P.O. Box 431
Dimondale, MI 48821
(616) 942-8987

Name: **The Vaults of Zurich**
System: PET, Atari
Memory: 16K PET
24K Atari
Language: BASIC
Description: Zurich is the banking capital of the world. The rich and powerful deposit their wealth in its famed impregnable vaults. But you, as a master thief, have dared to undertake the boldest heist of the century. You will journey down a maze of corridors and vaults, eluding the most sophisticated security system in the world. Your goal is to reach the Chairman's Chamber to steal the most treasured possession of all: the OPEC oil deeds!
Price: $21.95 cassette
$25.95 diskette
Author: Felix and Grcg Herlihy
Available:
Artworx Software Co.
150 N. Main St.
Fairport, NY 14450
(716) 425-2833
(800) 828-6573

Name: **Lock-It-Up 4.1**
System: Apple II or Apple II Plus
Memory: 48K
Language: Applesoft
Hardware: DOS 3.3
Description: *Lock-It-Up 4.1* is a sophisticated copy-protection system including over thirty state of the art protection features. It prevents copying of diskettes with any standard or "nibble" copiers including *Locksmith 4.1*. In addition, it allows the rapid duplication of diskettes protected with the system.
Price: $195.00
Includes two diskettes, manual, non-exclusive licensing agreement.
Author: Jeff Gold
Available:
Double-Gold Software
13126 Anza Drive
Saratoga, CA 95070
(408) 257-2247

Name: **Histogram Plot**
System: Apple II, DOS 3.3
Memory: 48K
Language: Applesoft
Hardware: Disk drive, printer optional
Description: *Histogram Plot* is an easy to use statistics package for the researcher, student and business man/woman in need of a quick, simple to use statistical data system. *Histogram* features input, save, and edit data options; variable graph size, demo files, display or printout of raw data, computed data, mean, median, standard deviation, expected cell frequencies, chi square, etc.
Price: $39.95
Includes 8-page manual
Author: J. McFarland
Available:
Andent, Inc.
1000 North Ave.
Waukegan, IL 60085

Name: **S-C Macro Assembler**
System: Apple II or Apple II Plus, DOS 3.3
Memory: 32K or more
Language: Machine code
Hardware: Disk II
Description: New version of our most popular product; adds macros, conditional assembly, and easier editing. Has 20 directives and 29 commands. Powerful EDIT command with 15 subcommands. Co-resident editor/assembler allows fast modification, re-assembly, and testing. Assembles up to 6000 lines per minute. Source programs may be as large as your disk space. Comes with a 100-page manual and both standard memory and Language Card versions. Liberal upgrade policy for registered owners of previous versions.
Price: $80.00
Available:
S-C Software Corporation
P.O. Box 280300
2331 Gus Thomasson
Suite 125
Dallas, TX 75228
(214) 324-2050

Name: **Mailing Label Package**
System: OS65U
Memory: 48K
Language: BASIC
Hardware: Ohio Scientific C-2 or C-3 Series
Description: This elaborate mailing program contains a direct cursor-aided input/edit feature plus automated internal/external file sorting and packing selections. Many other features.
Price: $75.00
Includes program disk and user's manual.
Available:
Electronic Information Systems, Inc.
P.O. Box 5893
Athens, GA 30604
(404) 353-2858

Name: **PAL (Personal Aid to Learning)**
System: Apple II
Memory: 48K
Language: BASIC
Hardware: One or two disk drives
Description: *PAL* is the first diagnostic/remediation program ever written for reading education. *PAL* covers the entire scope and sequence of reading education for each grade two through six. *PAL* actually diagnoses the cause of each reading problem, then provides remediation directly targeted at those problems.
Price: $99.95 for master disk package;
$99.95 for each grade level package
$9.95 for demo-disk package
Author: Stanley Crane
Dr. Dale Foreman
Daniel Myers
Available:
Universal Systems for Education, Inc.
2120 Academy Circle
Suite E
Colorado Springs, CO 80909
(303) 574-4575

Software Catalog *(continued)*

Name: **Client Records/Bill Preparation Order #0248AD-C10**
System: Apple II or Apple II Plus
Memory: 32K
Language: BASIC
Hardware: One disk drive
Description: *Client Records/Bill Preparation* is designed to help lawyers, doctors, consultants, and other service business owners quickly and easily keep accurate records and prepare monthly bills. This program, which can be modified to suit an individual business, allows the user to record client name, address, phone number, zip code, and four descriptive comments. The bill preparation function automatically totals up all charges that have been added to a client's file since the last billing and lists these charges, in detail, on the new invoice.
Price: $49.95
Author: D.C. Goodfellow
Available:
Instant Software
Peterborough, NH 03458

Name: **XenoFile™**
System: UCSD p-System
Memory: 48Kb runtime environment; 64Kb development environment
Language: Written in UCSD Pascal
Hardware: 8086, Z80, 8080, 8085, 6502, 9900, 6809, 68000, LSI-11/PDP-11
Description: *XenoFile* allows you to access CP/M files and disks from UCSD p-System programs. Using *XenoFile* you can translate CP/M files to UCSD p-System files, as well as use CP/M program output as UCSD p-System input and *vice versa*.
Price: $50.00
Includes object code for *XenoFile*.
Available:
SofTech Microsystems, Inc.
9494 Black Mountain Rd.
San Diego, CA 92126
(714) 578-6105

Name: **VIP II**
System: Apple II or Apple II Plus
Memory: 32K
Language: Applesoft
Description: Prints VisiCalc formulas exactly as they appear in the model, in columns and rows. Provides hard copy of VisiCalc formulas, automatically segmented and printed in blocks. Operates on any symmetrical models up to 26 columns wide with any number of rows. Fast machine language read and sort.
Price: $29.95
Includes documentation booklet.
Author: Mike Harvey
Available:
Micro-SPARC Systems Div.
Dept. P, P.O. Box 325
Lincoln, MA 01773
(617) 259-9710

Software Catalog *(continued)*

Name: **VC-Plus**
System: Apple II
Memory: 48K
Language: Assembly
Hardware: Legend 64KC and/or 128KDE Cards

Description: Add 82K or 145K of free memory space to VisiCorp's™ VisiCalc™ program by using the VC-Plus program with one (1) or two (2) Legend 64KC or 128KDE cards. The program comes on the 128KDE card's demo disk and is available for the 64KC user. No language or other 16K card is required.

Price: $34.95
Includes disk and manual.

Available:
Legend Industries Ltd.
2220 Scott Lake Road
Pontiac, MI 48054
(313) 674-0953

Name: **Eureka Learning System**
System: Apple II or Apple II Plus
Memory: 48K
Language: Applesoft in ROM

Description: A courseware generator, enabling the creation of CAI courses without any programming experience. The *Eureka Learning System* utilizes graphics, special characters, and sound to present lesson material to students.

Price: $495.00
Includes programs, character and graphic editors, tutorial manual, demonstration lessons.

Author: EICONICS, Inc.

Available:
The Programmers, Inc.
211 Cruz Alta Road
P.O. Box 1207
Taos, New Mexico 87571
or local Eureka™ Learning System dealers

Name: **VersaForm**
System: Apple II and Apple III
Memory: Apple II - 64K; Apple III - 128K
Language: Pascal-based
Hardware: Two disk drives or hard disk plus one floppy disk

Description: *VersaForm* is a Pascal-based business forms processor for Apple Systems with diskette/hard disk. A user-friendly interactive design facility aids form conversion or creation. A powerful set of auto-fill, data entry checking and calculation features are provided. Forms are stored and retrieved. Forms data may be printed as displayed or directed to specific print locations. Management reports may be prepared from specified fields from selectively retrieved forms.

Price: $389 - Apple II soft disk; $495 - hard disk; $495 - Apple III.
Includes user guide, reference summary, tutorial package, program and tutorial diskettes.

Available:
Dealers throughout USA including some Computerlands and Distributors

Name: **BusiComp**
System: Apple II Plus or Apple with Integer Card
Memory: 48K
Language: BASIC
Hardware: Capable of expansion to hard disk

Description: Complete interactive business system including AR/AP/inventory/fixed assets/general ledger/payroll.

Price: $1500.00
Includes program and documentation

Available:
Computer stores nationwide

Name: **TPS Canadian Payroll System**
System: Apple II or Apple II Plus
Memory: 48K
Language: Applesoft BASIC

Description: The *TPS Canadian Payroll System* can be used throughout Canada accommodating up to 100 employees on two diskettes. Features include: costing, T4's, statements or cheques, UIC records of employment reporting, report generator, union maintenance and reporting.

Price: $550.00
Includes diskette and manual.

Available:
Time Proven Systems Inc.
1210 Sheppard Ave. E.
Suite 101
Willowdale, Ontario, M2K 1E3
or supporting Apple dealers

Name: **Planet Lander**
System: OSI C1P, C4P
Memory: 4K
Language: BASIC Machine

Description: This is a simulation of a lunar lander landing on a planet's surface. You land the ship by controlling its velocity horizontally and vertically. Game points vary depending on which landing pad you elect. There are two levels of play: novice and expert. The controls are easy and make this *Planet Lander* fun to play.

Price: $10.95

Author: Thomas Andrew Paulson

Available:
Aurora Software Associates
37 S. Mitchell
Arlington Heights, IL 60005
(312) 259-3150

Name: **The Illustrator**
System: Apple II
Memory: 48K
Language: 6502 Machine Language
Hardware: Disk Drive

Description: *The Illustrator* is a fully integrated graphics package including 17 billion possible color combinations, a pallette for mixing, image movement, fast color fill, hi-res brush sets, disk save, custom color menus and magnify feature.

Price: $95.00
Includes disk and manual.

Author: Steve Dompier

Available:
Island Graphics
Box V
Bethel Island, CA 94511

Name: **A2-EDI Whole Brain Spelling**
System: Apple II Plus
Memory: 48K
Language: Applesoft

Description: An educational software package designed to help the user develop internal visualization skills for improving spelling. As entertaining as it is educationally sound. Utilizes the color graphic capabilities of the Apple II Computer to provide positive user feedback, and is extremely user-friendly.

Price: $34.95
Includes 2,000 practice words, available in a variety of categories.

Author: David Manton, Susan Campanini, and Joe Weintraub

Available:
Sublogic Communications Corporation
713 Edgebrook Drive
Champaign, IL 61820

Name: **Lemmings**
System: Apple II or Apple II Plus
Memory: 48K RAM
Language: Assembly
Hardware: None required - compatible with Joyport

Description: Challenging new computer game based on controlling a rapidly expanding population of lemmings. Armed only with your wits, your mission is to lock up a pair of non-breeding lemmings in every building in town. You are aided by a SPCA truck that will take lemmings to the clinic for neutering. Failure to control breeding causes a mass suicide jump in the sea, ending the game.

Price: $24.95
Includes complete instructions.

Author: Dan Thompson

Available:
Sirius Software, Inc.
10364 Rockingham Drive
Sacramento, CA 95827

Name: **Mazerace**
System: Radio Shack Color Computer
Memory: 16K
Language: Extended BASIC
Hardware: Joysticks

Description: *Mazerace* is a fun board-type game that involves both chance and strategy. The playing field is an 18 x 18 hexagon matrix that is partially filled with obstacles. Either one person against the computer or two people can play, with the computer or the players scrambling the playing field randomly to keep the action exciting. *Mazerace* uses high-resolution graphics.

Price: $17.95 on cassette; $22.95 on disk
Includes program and playing instructions.

Author: Ross R. Humer

Available:
Computerware
Box 668
Encinitas, CA 92024
(714) 436-3512

New Publications

(continued from page 95)

CONTENTS: Introduction: The Enemy Is Us; Overview: What Are the Issues? Tales of Horror; The Home Front: You and the Computer; Conspicuous Computing; Software Engineering; Human Engineering; Privacy and Security; Economics of Computing; Functional Specifications and Contracts; Managing the Machines; Getting Educated; Funding Computing; What Is, What Ain't, What Will Be; Index.

Microprocessor Applications Handbook, by David F. Stout. McGraw-Hill Book Company (New York, NY), 1982, 472 pages, 284 illustrations, 6¼ × 9¼ inches, hardcover.
ISBN 0-07-061798-8 $35.00

This book is addressed to individuals who design — or would like to design — intelligent systems. A wide variety of microprocessor applications are presented, based on contributions from specialists in many diversified fields. Most chapters contain both hardware and software aspects of microprocessor system design and discuss specific design information gathered during the development of actual microprocessor applications.

CONTENTS: Survey of Microprocessor Technology; A Microprocessor-Based Interface for the IEEE-488 Bus; Hamming Code Error Correction for Microcomputers;A Microprocessor-Controlled Color TV Receiver; Microcomputer Data Acquisition Module; A Microprocessor-Controlled Lumber Grader; Programmable Video Games; Microprocessor — A/D Converter Interfaces; Microcomputer Applications In Telephony; A 32-Channel Digital Waveform Synthesizer; Digital Filters Utilizing Microprocessors; Parallel and Serial Microprocessor Data Interfaces; Keyboard Data Input Techniques; Voice Recognition; A Slow-Scan Television System Using a Microprocessor; Hardware-Oriented State-Description Techniques; Multiple Microcomputers in Small Systems.

MICRObits

(Continued from page 51)

6800/6809 Software

Includes compatible single-user, multiuser and network-operating systems, compilers, accounting and word processing packages. Free catalog.

Software Dynamics
2111 W. Crescent, Sta. G
Anaheim, CA 92801

Software Catalog *(continued)*

Name: **VC-Plus**
System: Apple II
Memory: 48K
Language: Assembly
Hardware: Legend 64KC and/or 128KDE Cards

Description: Add 82K or 145K of free memory space to VisiCorp's™ VisiCalc™ program by using the VC-Plus program with one (1) or two (2) Legend 64KC or 128KDE cards. The program comes on the 128KDE card's demo disk and is available for the 64KC user. No language or other 16K card is required.

Price: $34.95
Includes disk and manual.

Available:
Legend Industries Ltd.
2220 Scott Lake Road
Pontiac, MI 48054
(313) 674-0953

Name: **Eureka Learning System**
System: Apple II or Apple II Plus
Memory: 48K
Language: Applesoft in ROM

Description: A courseware generator, enabling the creation of CAI courses without any programming experience. The *Eureka Learning System* utilizes graphics, special characters, and sound to present lesson material to students.

Price: $495.00
Includes programs, character and graphic editors, tutorial manual, demonstration lessons.

Author: EICONICS, Inc.

Available:
The Programmers, Inc.
211 Cruz Alta Road
P.O. Box 1207
Taos, New Mexico 87571
or local Eureka™ Learning System dealers

Name: **VersaForm**
System: Apple II and Apple III
Memory: Apple II - 64K; Apple III - 128K
Language: Pascal-based
Hardware: Two disk drives or hard disk plus one floppy disk

Description: *VersaForm* is a Pascal-based business forms processor for Apple Systems with diskette/hard disk. A user-friendly interactive design facility aids form conversion or creation. A powerful set of auto-fill, data entry checking and calculation features are provided. Forms are stored and retrieved. Forms data may be printed as displayed or directed to specific print locations. Management reports may be prepared from specified fields from selectively retrieved forms.

Price: $389 - Apple II soft disk; $495 - hard disk; $495 - Apple III.
Includes user guide, reference summary, tutorial package, program and tutorial diskettes.

Available:
Dealers throughout USA including some Computerlands and Distributors

---

Name: **BusiComp**
System: Apple II Plus or Apple with Integer Card
Memory: 48K
Language: BASIC
Hardware: Capable of expansion to hard disk

Description: Complete interactive business system including AR/AP/inventory/fixed assets/general ledger/payroll.

Price: $1500.00
Includes program and documentation

Available:
Computer stores nationwide

Name: **TPS Canadian Payroll System**
System: Apple II or Apple II Plus
Memory: 48K
Language: Applesoft BASIC

Description: The *TPS Canadian Payroll System* can be used throughout Canada accommodating up to 100 employees on two diskettes. Features include: costing, T4's, statements or cheques, UIC records of employment reporting, report generator, union maintenance and reporting.

Price: $550.00
Includes diskette and manual.

Available:
Time Proven Systems Inc.
1210 Sheppard Ave. E.
Suite 101
Willowdale, Ontario, M2K 1E3
or supporting Apple dealers

Name: **Planet Lander**
System: OSI C1P, C4P
Memory: 4K
Language: BASIC Machine

Description: This is a simulation of a lunar lander landing on a planet's surface. You land the ship by controlling its velocity horizontally and vertically. Game points vary depending on which landing pad you elect. There are two levels of play: novice and expert. The controls are easy and make this *Planet Lander* fun to play.

Price: $10.95

Author: Thomas Andrew Paulson

Available:
Aurora Software Associates
37 S. Mitchell
Arlington Heights, IL 60005
(312) 259-3150

Name: **The Illustrator**
System: Apple II
Memory: 48K
Language: 6502 Machine Language
Hardware: Disk Drive

Description: *The Illustrator* is a fully integrated graphics package including 17 billion possible color combinations, a pallette for mixing, image movement, fast color fill, hi-res brush sets, disk save, custom color menus and magnify feature.

Price: $95.00
Includes disk and manual.

Author: Steve Dompier

Available:
Island Graphics
Box V
Bethel Island, CA 94511

Name: **A2-EDI Whole Brain Spelling**
System: Apple II Plus
Memory: 48K
Language: Applesoft

Description: An educational software package designed to help the user develop internal visualization skills for improving spelling. As entertaining as it is educationally sound. Utilizes the color graphic capabilities of the Apple II Computer to provide positive user feedback, and is extremely user-friendly.

Price: $34.95
Includes 2,000 practice words, available in a variety of categories.

Author: David Manton, Susan Campanini, and Joe Weintraub

Available:
Sublogic Communications Corporation
713 Edgebrook Drive
Champaign, IL 61820

Name: **Lemmings**
System: Apple II or Apple II Plus
Memory: 48K RAM
Language: Assembly
Hardware: None required - compatible with Joyport

Description: Challenging new computer game based on controlling a rapidly expanding population of lemmings. Armed only with your wits, your mission is to lock up a pair of non-breeding lemmings in every building in town. You are aided by a SPCA truck that will take lemmings to the clinic for neutering. Failure to control breeding causes a mass suicide jump in the sea, ending the game.

Price: $24.95
Includes complete instructions.

Author: Dan Thompson

Available:
Sirius Software, Inc.
10364 Rockingham Drive
Sacramento, CA 95827

Name: **Mazerace**
System: Radio Shack Color Computer
Memory: 16K
Language: Extended BASIC
Hardware: Joysticks

Description: *Mazerace* is a fun board-type game that involves both chance and strategy. The playing field is an 18 x 18 hexagon matrix that is partially filled with obstacles. Either one person against the computer or two people can play, with the computer or the players scrambling the playing field randomly to keep the action exciting. *Mazerace* uses high-resolution graphics.

Price: $17.95 on cassette; $22.95 on disk
Includes program and playing instructions.

Author: Ross R. Humer

Available:
Computerware
Box 668
Encinitas, CA 92024
(714) 436-3512

New Publications

(continued from page 95)

CONTENTS: Introduction: The Enemy Is Us; Overview: What Are the Issues? Tales of Horror; The Home Front: You and the Computer; Conspicuous Computing; Software Engineering; Human Engineering; Privacy and Security; Economics of Computing; Functional Specifications and Contracts; Managing the Machines; Getting Educated; Funding Computing; What Is, What Ain't, What Will Be; Index.

Microprocessor Applications Handbook, by David F. Stout. McGraw-Hill Book Company (New York, NY), 1982, 472 pages, 284 illustrations, 6¼ × 9¼ inches, hardcover.
ISBN 0-07-061798-8 $35.00

This book is addressed to individuals who design — or would like to design — intelligent systems. A wide variety of microprocessor applications are presented, based on contributions from specialists in many diversified fields. Most chapters contain both hardware and software aspects of microprocessor system design and discuss specific design information gathered during the development of actual microprocessor applications.

CONTENTS: Survey of Microprocessor Technology; A Microprocessor-Based Interface for the IEEE-488 Bus; Hamming Code Error Correction for Microcomputers;A Microprocessor-Controlled Color TV Receiver; Microcomputer Data Acquisition Module; A Microprocessor-Controlled Lumber Grader; Programmable Video Games; Microprocessor — A/D Converter Interfaces; Microcomputer Applications In Telephony; A 32-Channel Digital Waveform Synthesizer; Digital Filters Utilizing Microprocessors; Parallel and Serial Microprocessor Data Interfaces; Keyboard Data Input Techniques; Voice Recognition; A Slow-Scan Television System Using a Microprocessor; Hardware-Oriented State-Description Techniques; Multiple Microcomputers in Small Systems.

MICRO

MICRObits

(Continued from page 51)

6800/6809 Software

Includes compatible single-user, multiuser and network-operating systems, compilers, accounting and word processing packages. Free catalog.

Software Dynamics
2111 W. Crescent, Sta. G
Anaheim, CA 92801

MICRO

Would you give $295?

If you're currently using Apple Pascal* on your Apple II, you're probably aware of some noticeable limitations. And you'd probably give a lot for an upgrade package, including the UCSD p-System,* UCSD Pascal* and TURTLEGRAPHICS, that would get your Apple* to do what it's capable of.

Upgrade to the UCSD p-System Version IV from SofTech Microsystems. It's got all the features of Apple Pascal, and then some. For instance, Apple Pascal's UNITS must be linked in at each compilation, the p-System's do not. And instead of being limited to 32 UNITS, like Apple Pascal, the p-System allows a virtually unlimited number.

How about peripheral support? The p-System supports all the peripherals that Apple Pascal does, plus a clock, and a lower case adapter. And, we get more out of the peripherals you've already got—shiftware modification on the keyboard, alpha lock key, typeahead and characters not even on the Apple keyboard.

And when it comes to graphics, our TURTLEGRAPHICS has everything in Apple's graphics, plus automatic scaling and four graphic modes, including both HIRES and LORES.

Then there's portability. The p-System lets you develop genuinely portable, high-level applications for nearly any microcomputer around. It allows you to work in any combination of UCSD Pascal and BASIC (available as an add-on). And it provides support for dynamic memory management and multitasking, with a full arsenal of enhancements. And if that isn't enough, your existing Apple Pascal programs are upward compatible with the p-System, and simply have to be recompiled to execute. All your Apple II needs is 64K of RAM and two disk drives.

Last but not least, there's the price. Normally, you'd have to pay as much as $825 for such a package.

But, for the next two months, we're making this special upgrade offer to Apple Pascal users for a mere $295. That's a savings of over 60%.

So just send in the coupon below, with your proof of purchase and check, money order or Visa or MasterCard number, and you'll be on your way to getting more out of your Apple II than you ever dreamed of. But you'd better hurry. Your two months have already started.

SOFTECH
MICROSYSTEMS

Okay, SofTech Microsystems, here's my $295. I want my Apple II to have software it can really appreciate.
My check is enclosed □
Please charge to my Acct. #________________________
Visa □ Master Charge □ Expiration Date________ Name on card________________________
I hereby certify that I am an Apple Pascal Owner.
Signature________________________________

Please send coupon to: **Apple Upgrade Dept.**
P.O. Box 27179, San Diego, CA 92128
Or call (714) 578-6995

My proof of purchase is
□ invoice □ receipt □ disk label □ other

Name________________________ **Title**________________________
Company____________________ **Telephone**________________ **Ext.**______
Address____________________ **City**____________ **State**______ **Zip**______

OFFER VALID JULY 1 to AUGUST 31, 1982

(California residents please add 6% sales tax [California Transit District—6.5%] Massachusetts residents please add 5% sales tax.)

*UCSD p-System and UCSD Pascal are trademarks of the Regents of the University of California. Apple, Apple II, and Apple Pascal are registered trademarks of Apple Computer, Inc.

DR. DOBB'S JOURNAL

For Users of Small Computer Systems

with topics like:

Small-C
FORTH
Lisp
CP/M
S-100
Cross-assemblers
8086/8088
16-bit tech.
Pascal
Assembler
Compilers
Algorithms
8080/Z80
6502
etc.

Dr. Dobb's Journal
For Rubikubists an Ally in BASIC
Dr. Dobb's Journal
How to Run Tektronix Software on Micros . . .
. . . a Poor Man's FORTH Computer . . .
and Telecommunications News!

Each issue includes:

- **valuable software tools**
- **algorithms & problem solving**
- **industry news**
- **important product reviews**

With in depth coverage of:

- **telecommunications**
- **systems programming**
- **language development**
- **machine independent programs**

and much, much more!

Yes! Please enter my subscription for T4
□ 1 yr. (12 issues) $25 □ 2 yrs. $47 (save $13 off newsstand)
□ Please bill me □ I enclose check/money order
Name____________________
Address ____________________
City__________ State_____ Zip________

Are you ready?

DDJ, the world's foremost microcomputer publication, has been working for years to prepare its readers to be innovators, to lead the wave of breakthroughs in our changing technology.

Every issue of *Dr. Dobb's Journal* helps one to understand the nuts and bolts of small computer systems. We offer entire listings of valuable software: our pages have included compilers,' cross-assemblers, editors, new languages, hardware interfaces and more – usually before anyone else thinks of them!

Even more important!

As valuable and significant as all those things are, there is an even greater reason for you to join forces with *DDJ*. That is the keen, responsive readership. Our subscribers share insights, correspond, and contribute to one another's work, more than any other group we know. They treat *Dr. Dobb's Journal* as a "hands-on" publication.

This warm cooperation has done more to refine software products, and generally to advance the state of microcomputer technology, than perhaps any other resource. And it is available to you through our pages!

For the straight Facts . . .

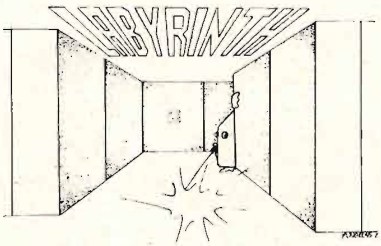
If you are a serious computing professional or enthusiast, then you should take a very close look at what *DDJ* offers you. We've been on the cutting edge since 1976.

Dr. Dobb's Journal • A Publication of People's Computer Company, 1263 El Camino Real, P.O. Box E, Menlo Park, CA 94025

QUALITY SOFTWARE FOR TRS-80 COLOR AND OSI

QUEST – A NEW IDEA IN ADVENTURE GAMES! Different from all the others. Quest is played on a computer generated map of Alesia. Your job is to gather men and supplies by combat, bargaining, exploration of ruins and temples and outright banditry. When your force is strong enough, you attack the Citadel of Moorlock in a life or death battle to the finish. Playable in 2 to 5 hours, this one is different every time.
16K COLOR-80 OR TRS-80 or 12KOSI.
$14.95.

BASIC THAT ZOOOMMS!!
AT LAST AN AFFORDABLE COMPILER FOR OSI AND TRS-80 COLOR MACHINES!!! The compiler allows you to write your programs in easy BASIC and then automatically generates a machine code equivalent that runs 50 to 150 times faster.
It does have some limitations. It takes at least 8K of RAM to run the compiler and it does only support a subset of BASIC–about 20 commands including FOR, NEXT, END, GOSUB, GOTO, RETURN, END, PRINT, STOP, USR(X), PEEK, POKE, *, /, +, -, X , X , = , VARIABLE NAMES A-Z, A SUBSCRIPTED VARIABLE, and INTEGER NUMBERS FROM 0 - 64K.
TINY COMPILER is written in BASIC. It generates native, relocatable 6502 or 6809 code. It comes with a 20 page manual and can be modified or augmented by the user. $24.95 on tape or disk for OSI or TRS-80 Color.



LABYRINTH – 16K EXTENDED COLOR BASIC – With amazing 3D graphics, you fight your way through a maze facing real time monsters. The graphics are real enough to cause claustrophobia. The most realistic game that I have ever seen on either system. $14.95. (8K on OSI)

VENTURER!–A fast action all machine code Arcade game that feels like an adventure. Go berserk as you sneak past the DREADED HALL MONSTERS to gather treasure in room after room, killing the NASTIES as you go. Great color, high res graphics, sound and Joystick game for the TRS-80 Color or OSI machines. (black and white and silent on OSI.) Tape only. $19.95.

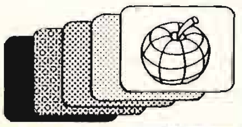
AARDVARK - 80
2352 S. Commerce, Walled Lake, MI 48088
TRS 80 COLOR (313) 669-3110 OSI

COSMOS

560 SCREEN MIXER

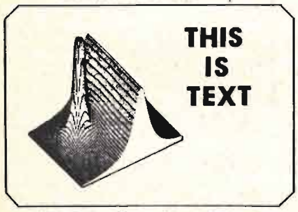
560
DOTS/LINE

Double Density HGR



NORMAL
HIGHLIGHT

Highlight



Hi-Res Graphics+Text

THIS IS TEXT
LGR

Lo-Res Graphics+Text

SCREEN MIXER is a set of three modules for APPLE-II. Just plug-in these modules to your Apple-II, and you will have the Apple-II with more features you could not expect till now----

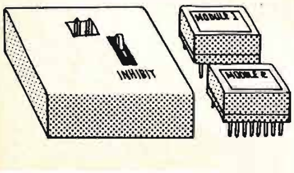
SCREEN MIXER provides:

- ●The mixed screen of any two of screens available for the Apple-II. Please note that all of HGR, LGR and Text screen has two pages. The mixing is done with hardware, not like Hi-Res Text Generater Programs, thus you need no software and the scroll speed is not reduced. Also, you can scroll the text without any effect to the graphic patterns.
- ●The Double Density High Resolution Graphics. Yes, you can plot 580 dots in one line. You have only 280 dots in one line on ordinary Apple-II. (Software is required)
- ●One of the most advanced character display. Besides Normal and Inverse characters which are already built-in you will have the choice of Half-Intensity and Hilighted characters. And more, you may Over-Write or Over-Type any character to other character if you want to do so! (Software is included)

NORMAL
HALF
INTENSITY

Half Intensity

Over Write



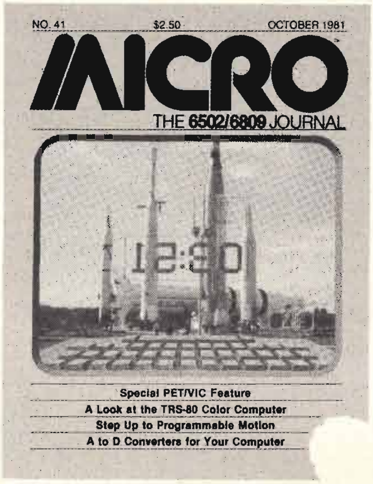
$60 MANUAL — $10.00, applied towards purchase.

COSMOS™
WORLD WIDE COMPUTER SUPER SHOP

Dealers inquiries invited.
For more information call or write to ASTAR INTERNATIONAL CO.
5676 FRANCIS AVE., CHINO, CA 91710 Phone 714-627-9887

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

A feast of computing ideas.


NO. 41
$2.50
OCTOBER 1981
MICRO
THE 6502/6809 JOURNAL
12:30
Special PET/VIC Feature
A Look at the TRS-80 Color Computer
Step Up to Programmable Motion
A to D Converters for Your Computer

You'll love every byte.

If you work with a 6502/6809-based system, you're probably hungry for the facts and ideas that will help you understand the inner workings of your computer. You want to go beyond canned software—use your computer for more than games—learn the advanced programming techniques that enable you to get the most out of your 6502/6809 system.

MICRO, The 6502/6809 Journal, gives you page after page, month after month, of solid information to sink your teeth into. **MICRO** is the premier how-to magazine for serious users of the Apple, PET/CBM, OSI, Atari, AIM, SYM, KIM, and all 6809 based systems including the TRS-80 Color Computer. It's a resource journal internationally respected by professionals in business, industry, and education. Every issue of **MICRO** keeps you informed with up-to-the-minute data on new products and publications:

- **hardware catalog** with organized, concise description
- **software catalog** in an easy-to-use format
- **new publications** listed and annotated
- **reviews and evaluations** of significant products

And there's much more:

- **In-depth hardware tutorials** bring expert advice into your home or office.
- **Detailed discussions of programming languages** deepen and broaden your programming ability.
- **Complete program listings** enable you to increase your machine's capabilities.
- **Bibliography of 6502/6809 information** helps you to find pertinent articles in a timely manner.
- **Special monthly features** with in-depth treatment of one subject or system increase your knowledge of the field.
- **Balanced mix of machine-specific and general articles** for your everyday use as well as long-range reference needs.
- **Informative advertising** focused specifically on 6502/6809 machines keeps you abreast of latest developments.
- **Reader feedback** puts you in touch with other micro-computerists.

MICRO is the magazine you need to get the most from you. own 6502/6809 system!

To order, send your check or international money order (payable to MICRO) and the order form at left, to:

Subscription Fulfillment
MICRO, Dept. MI
34 Chelmsford Street
P.O. Box 6502
Chelmsford, MA 01824

Or, for your convenience, call our toll-free number:

1-800-345-8112

(In Pennsylvania, 1-800-662-2444)

and charge your subscription to your MasterCard or VISA. (All orders must be prepaid in U.S. dollars or charged to your MasterCard or VISA.)

SUBSCRIPTION RATES (U.S. dollars)
Yearly subscription (ISSN 027-9002) saves 20% off the single-issue price.

U.S. $24*
Canada $27
Europe $27 ($42 by air mail)
Mexico, Central America, Mideast, North and Central Africa $27 ($48 air)
South America, Far East, South Africa, Australasia $27 ($72 air)

* SPECIAL OFFER—U.S. ONLY:
Save even more—30% off single-issue price: 2 years, $42

YES! I want to get more from my microcomputer. Please send me

___ year(s) of MICRO at $_____ /year.
(Outside U.S. and Canada, please indicate via □ surface or □ air mail.)

Name __________

Company __________

Street __________

City _____ State _____ Zip Code _____

□ Check enclosed for $ _____
□ Charge my credit card account
□ VISA □ MasterCard

Signature __________

Card number __________

Expiration date __________

Apple II

Apple II and Apple II Plus

6502-Based Microcomputer

Available in 16K, 20K, 24K, 32K, 36K, 48K and 64K (with Language Card) configurations.

Apple II comes standard with Integer BASIC
Apple II Plus comes standard with Applesoft BASIC

There are eight expansion slots for use with peripheral devices.

Additional CPU boards are available to run the 6809, the Z80, and the 8088.

Although the Apple II was designed about five years ago, it remains one of the fastest-selling models on the market today. The Apple II has proven itself to be a reliable and extremely flexible unit. The Apple II enjoys an envious position of having the largest software base of any computer on the market today.

Memory Map (48K)

LOCATION		
DECIMAL	HEX	USAGE
0-255	$0-$FF	Zero-page system storage
256-511	$100-$1FF	System stack
512-767	$200-$2FF	Keyboard character buffer
768-975	$300-$3CF	Often available as free space for user programs
976-1023	$3D0-$3FF	System vectors
1024-2047	$400-$7FF	Text and lo-res graphics page 1
2048-LOMEM	$800-LOMEM	Program storage
2048-3071	$800-$BFF	Text and lo-res graphics page 2 or free space
3072-8191	$C00-$1FFF	Free space unless RAM Applesoft is in use
8192-16383	$2000-$3FFF	Hi-res page 1 or free space
16384-24575	$4000-$5FFF	Hi-res page 2 or free space
24576-38399	$6000-$95FF	Free space and string storage
38400-49151	$9600-$BFFF	DOS
49152-53247	$C000-$CFFF	I/O, hardware (reserved)
53248-57343	$D000-$DFFF	Applesoft in Language Card or ROM
57344-63487	$E000-$F7FF	Applesoft or Integer BASIC in Language Card or ROM
63488-65535	$F800-$FFFF	System monitor

Handy PEEKs, POKEs and CALLs

(E = PEEK, O = POKE, C = CALL)

LOCATION				
DECIMAL	HEX	USAGE	COMMAND(S)	VALUES
32	$20	Left column of text window	E,O	0-39
33	$21	Width of text window	E,O	1-40
34	$22	Top of text window	E,O	0-24
35	$23	Bottom of text window	E,O	0-24
36	$24	Cursor's horizontal position	E,O	0-39
37	$25	Cursor's vertical position	E,O	0-23
50	$32	Video inverse/normal/flashing control	E,O	255 = normal 127 = flashing 63 = inverse
216	$D8	> 127 if ONERR GOTO active	E,O	0 = inactive > 127 = active
218-219	$DA-$DB	Applesoft line # where error occurred	E	—
222	$DE	Error code (0,16-255 see Applesoft B.P.R.M., page #81) (1-15 See DOS Manual page #114)		
1013-1015	$3F5-$3F7	Applesoft's Ampersand Vector Address. Holds JMP instruction to subroutine which handles & command.	O	$4C, $LO, $HI
49152 (−16384)	$C000	Keyboard input. If > 127 key has been pressed since last strobed (valid value).	E	Negative ASCII
49168 (−16368)	$C010	Keyboard strobe. Access this location to set high order bit of $C000 to zero.	E	—
49184 (−16352)	$C020	Toggle cassette output	E	—
49200 (−16336)	$C030	Toggle speaker	E	—
49232 (−16304)	$C050	Set from text to graphics mode	O	0
49233 (−16303)	$C051	Set from graphics to text mode	O	0
49234 (−16302)	$C052	Reset to full screen graphics	O	0
49235 (−16301)	$C053	Set to mixed text/graphics	O	0
49236 (−16300)	$C054	Display page 1	O	0
49237 (−16299)	$C055	Display page 2	O	0
49238 (−16298)	$C056	Set to lo-res or text	O	0
49239 (−16297)	$C057	Set to hi-res	O	0
49240 (−16296)	$C058	Turn annunciator 0 off	O	0
49241 (−16295)	$C059	Turn annunciator 0 on	O	0
49242 (−16294)	$C05A	Turn annunciator 1 off	O	0
49243 (−16293)	$C05B	Turn annunciator 1 on	O	0
49244 (−16292)	$C05C	Turn annunciator 2 off	O	0
49245 (−16291)	$C05D	Turn annunciator 2 on	O	0
49246 (−16290)	$C05E	Turn annunciator 3 off	O	0
49247 (−16289)	$C05F	Turn annunciator 3 on	O	0
49248 (−16288)	$C060	Cassette input. > 127 = binary 1, < 128 = binary 0.	E	—
49249 (−16287)	$C061	Read pushbutton 0. > 127 = pushbutton pressed.	E	—
49250 (−16286)	$C062	Read pushbutton 1. > 127 = pushbutton pressed.	E	—
49251 (−16285)	$C063	Read pushbutton 2. > 127 = pushbutton pressed.	E	—
49252 (−16284)	$C064	Read game paddle 0	E	0-255
49253 (−16283)	$C065	Read game paddle 1	E	0-255
49254 (−16282)	$C066	Read game paddle 2	E	0-255
49255 (−16281)	$C067	Read game paddle 3	E	0-255
63538 (−1998)	$F832	Clear lo-res screen	C	—
63542 (−1994)	$F836	Clear lo-res screen (top 40 lines)	C	—
64473 (−1063)	$FBD9	Send a BELL character to current output device	C	—
64578 (−958)	$FC42	Clear text window from present cursor position to lower right of screen	C	—
64600 (−936)	$FC58	Clear entire screen and move the cursor to upper lefthand corner	C	—
64614 (−922)	$FC66	Move cursor down one line (no.change horizontally)	C	—
64668 (−868)	$FC9C	Clear text to end of line	C	—

Apple II

Apple II and Apple II Plus

6502-Based Microcomputer

Available in 16K, 20K, 24K, 32K, 36K, 48K and 64K (with Language Card) configurations.

Apple II comes standard with Integer BASIC
Apple II Plus comes standard with Applesoft BASIC

There are eight expansion slots for use with peripheral devices.

Additional CPU boards are available to run the 6809, the Z80, and the 8088.

Although the Apple II was designed about five years ago, it remains one of the fastest-selling models on the market today. The Apple II has proven itself to be a reliable and extremely flexible unit. The Apple II enjoys an envious position of having the largest software base of any computer on the market today.

Memory Map (48K)

LOCATION		
DECIMAL	HEX	USAGE
0-255	$0-$FF	Zero-page system storage
256-511	$100-$1FF	System stack
512-767	$200-$2FF	Keyboard character buffer
768-975	$300-$3CF	Often available as free space for user programs
976-1023	$3D0-$3FF	System vectors
1024-2047	$400-$7FF	Text and lo-res graphics page 1
2048-LOMEM	$800-LOMEM	Program storage
2048-3071	$800-$BFF	Text and lo-res graphics page 2 or free space
3072-8191	$C00-$1FFF	Free space unless RAM Applesoft is in use
8192-16383	$2000-$3FFF	Hi-res page 1 or free space
16384-24575	$4000-$5FFF	Hi-res page 2 or free space
24576-38399	$6000-$95FF	Free space and string storage
38400-49151	$9600-$BFFF	DOS
49152-53247	$C000-$CFFF	I/O, hardware (reserved)
53248-57343	$D000-$DFFF	Applesoft in Language Card or ROM
57344-63487	$E000-$F7FF	Applesoft or Integer BASIC in Language Card or ROM
63488-65535	$F800-$FFFF	System monitor

Handy PEEKs, POKEs and CALLs

(E = PEEK, O = POKE, C = CALL)

LOCATION				
DECIMAL	HEX	USAGE	COMMAND(S)	VALUES
32	$20	Left column of text window	E,O	0-39
33	$21	Width of text window	E,O	1-40
34	$22	Top of text window	E,O	0-24
35	$23	Bottom of text window	E,O	0-24
36	$24	Cursor's horizontal position	E,O	0-39
37	$25	Cursor's vertical position	E,O	0-23
50	$32	Video inverse/normal/flashing control	E,O	255 = normal 127 = flashing 63 = inverse
216	$D8	> 127 if ONERR GOTO active	E,O	0 = inactive > 127 = active
218-219	$DA-$DB	Applesoft line # where error occurred	E	—
222	$DE	Error code (0,16-255 see Applesoft B.P.R.M., page #81) (1-15 See DOS Manual page #114)		
1013-1015	$3F5-$3F7	Applesoft's Ampersand Vector Address. Holds JMP instruction to subroutine which handles & command.	O	$4C, $LO, $HI
49152 (−16384)	$C000	Keyboard input. If > 127 key has been pressed since last strobed (valid value).	E	Negative ASCII
49168 (−16368)	$C010	Keyboard strobe. Access this location to set high order bit of $C000 to zero.	E	—
49184 (−16352)	$C020	Toggle cassette output	E	—
49200 (−16336)	$C030	Toggle speaker	E	—
49232 (−16304)	$C050	Set from text to graphics mode	O	0
49233 (−16303)	$C051	Set from graphics to text mode	O	0
49234 (−16302)	$C052	Reset to full screen graphics	O	0
49235 (−16301)	$C053	Set to mixed text/graphics	O	0
49236 (−16300)	$C054	Display page 1	O	0
49237 (−16299)	$C055	Display page 2	O	0
49238 (−16298)	$C056	Set to lo-res or text	O	0
49239 (−16297)	$C057	Set to hi-res	O	0
49240 (−16296)	$C058	Turn annunciator 0 off	O	0
49241 (−16295)	$C059	Turn annunciator 0 on	O	0
49242 (−16294)	$C05A	Turn annunciator 1 off	O	0
49243 (−16293)	$C05B	Turn annunciator 1 on	O	0
49244 (−16292)	$C05C	Turn annunciator 2 off	O	0
49245 (−16291)	$C05D	Turn annunciator 2 on	O	0
49246 (−16290)	$C05E	Turn annunciator 3 off	O	0
49247 (−16289)	$C05F	Turn annunciator 3 on	O	0
49248 (−16288)	$C060	Cassette input. > 127 = binary 1, < 128 = binary 0.	E	—
49249 (−16287)	$C061	Read pushbutton 0. > 127 = pushbutton pressed.	E	—
49250 (−16286)	$C062	Read pushbutton 1. > 127 = pushbutton pressed.	E	—
49251 (−16285)	$C063	Read pushbutton 2. > 127 = pushbutton pressed.	E	—
49252 (−16284)	$C064	Read game paddle 0	E	0-255
49253 (−16283)	$C065	Read game paddle 1	E	0-255
49254 (−16282)	$C066	Read game paddle 2	E	0-255
49255 (−16281)	$C067	Read game paddle 3	E	0-255
63538 (−1998)	$F832	Clear lo-res screen	C	—
63542 (−1994)	$F836	Clear lo-res screen (top 40 lines)	C	—
64473 (−1063)	$FBD9	Send a BELL character to current output device	C	—
64578 (−958)	$FC42	Clear text window from present cursor position to lower right of screen	C	—
64600 (−936)	$FC58	Clear entire screen and move the cursor to upper lefthand corner	C	—
64614 (−922)	$FC66	Move cursor down one line (no.change horizontally)	C	—
64668 (−868)	$FC9C	Clear text to end of line	C	—

Apple II

MICRO™ Data Sheet #6

Decimal	Hex	ASCII	Screen	Integer BASIC	6502	Decimal	Hex	ASCII	Screen	Integer BASIC	6502	Decimal	Hex	ASCII	Screen	Applesoft BASIC	6502	Decimal	Hex	ASCII	Screen	Applesoft BASIC	6502
0	00	NUL	@I	HIMEM:	BRK	64	40	@	@I	$	RTI	128	80		@N	END		192	C0		@N	TAB(	CPY #
1	01	SOH	AI	EOS	ORA(I,X)	65	41	A	AF	$	EOR(I,X)	129	81		AN	FOR	STA(I,X)	193	C1		AN	TO	CMP(I,X)
2	02	STX	BI	—		66	42	B	BF	(		130	82		BN	NEXT		194	C2		BN	FN	
3	03	ETX	CI	:		67	43	C	CF	,		131	83		CN	DATA		195	C3		CN	SPC(	
4	04	EOT	DI	LOAD		68	44	D	DF	,		132	84		DN	INPUT	STY Z	196	C4		DN	THEN	CPY Z
5	05	ENQ	EI	SAVE	ORA Z	69	45	E	EF	;	EOR Z	133	85		EN	DEL	STA Z	197	C5		EN	AT	CMP Z
6	06	ACK	FI	CON	ASL Z	70	46	F	FF	;	LSR Z	134	86		FN	DIM	STX Z	198	C6		FN	NOT	DEC Z
7	07	BEL	GI	RUN		71	47	G	GF	;		135	87		GN	READ		199	C7		GN	STEP	
8	08	BS	HI	RUN	PHP	72	48	H	HF	,	PHA	136	88		HN	GR	DEY	200	C8		HN	+	INY
9	09	HT	II	DEL	ORA #	73	49	I	IF	,	EOR #	137	89		IN	TEXT		201	C9		IN	–	CMP #
10	0A	LF	JI	,	ASL A	74	4A	J	JF	,	LSR	138	8A		JN	PR#	TXA	202	CA		JN	*	DEX
11	0B	VT	KI	NEW		75	4B	K	KF	TEXT		139	8B		KN	IN#		203	CB		KN	/	
12	0C	FF	LI	CLR		76	4C	L	LF	GR	JMP	140	8C		LN	CALL	STY	204	CC		LN	^	CPY
13	0D	CR	MI	AUTO	ORA	77	4D	M	MF	CALL	EOR	141	8D		MN	PLOT	STA	205	CD		MN	AND	CMP
14	0E	SO	NI	,	ASL	78	4E	N	NF	DIM	LSR	142	8E		NN	HLIN	STX	206	CE		NN	OR	DEC
15	0F	SI	OI	MAN		79	4F	O	OF	DIM		143	8F		ON	VLIN		207	CF		ON	>	
16	10	DLE	PI	HIMEM:	BPL	80	50	P	PF	TAB	BVC	144	90		PN	HGR2	BCC	208	D0		PN	=	BNE
17	11	DC1	QI	LOMEM:	ORA(I),Y	81	51	Q	QF	END	EOR(I),Y	145	91		QN	HGR	STA(I),Y	209	D1		QN	<	CMP(I),Y
18	12	DC2	RI	+		82	52	R	RF	INPUT		146	92		RN	HCOLOR=		210	D2		RN	SGN	
19	13	DC3	SI	–		83	53	S	SF	INPUT		147	93		SN	HPLOT		211	D3		SN	INT	
20	14	DC4	TI	*		84	54	T	TF	INPUT		148	94		TN	DRAW	STY Z,X	212	D4		TN	ABS	
21	15	NAK	UI	/	ORA Z,X	85	55	U	UF	FOR	EOR Z,X	149	95		UN	XDRAW	STA Z,X	213	D5		UN	USR	CMP Z,X
22	16	SYN	VI	=	ASL Z,X	86	56	V	VF	=	LSR Z,X	150	96		VN	HTAB	STX Z,Y	214	D6		VN	FRE	DEC Z,X
23	17	ETB	WI	#		87	57	W	WF	TO		151	97		WN	HOME		215	D7		WN	SCRN(	
24	18	CAN	XI	>=	CLC	88	58	X	XF	STEP	CLI	152	98		XN	ROT=	TYA	216	D8		XN	PDL	CLD
25	19	EM	YI	>	ORA Y	89	59	Y	YF	NEXT	EOR Y	153	99		YN	SCALE=	STA Y	217	D9		YN	POS	CMP Y
26	1A	SUB	ZI	<=		90	5A	Z	ZF	,		154	9A		ZN	SHLOAD	TXS	218	DA		ZN	SQR	
27	1B	ESC	[I	<>		91	5B	[	[F	RETURN		155	9B		[N	TRACE		219	DB		[N	RND	
28	1C	FS	\I	<		92	5C		\F	GOSUB		156	9C		\N	NOTRACE		220	DC		\N	LOG	
29	1D	GS	]I	AND	ORA X	93	5D	]	]F	REM	EOR X	157	9D		]N	NORMAL	STA X	221	DD		]N	EXP	CMP X
30	1E	RS	^I	OR	ASL X	94	5E		^F	LET	LSR X	158	9E		^N	INVERSE		222	DE		^N	COS	DEC X
31	1F	US	_I	MOD		95	5F	–	–F	GOTO		159	9F		_N	FLASH		223	DF		_N	SIN	
32	20	SPC	I	^	JSR	96	60	,	F	IF	RTS	160	A0		N	COLOR=	LDY #	224	E0		N	TAN	CPX #
33	21	!	!I	+	AND(I,X)	97	61	a	!F	PRINT	ADC(I,X)	161	A1		!N	POP	LDA(I,X)	225	E1		!N	ATN	CMP(I,X)
34	22	"	"I	(		98	62	b	"F	PRINT		162	A2		"N	VTAB	LDX #	226	E2		"N	PEEK	
35	23	#	#I	,		99	63	c	#F	PRINT		163	A3		#N	HIMEM:		227	E3		#N	LEN	
36	24	$	$I	THEN	BIT Z	100	64	d	$F	POKE		164	A4		$N	LOMEM:	LDY Z	228	E4		$N	STR$	CPX Z
37	25	%	%I	THEN	AND Z	101	65	e	%F	,	ADC Z	165	A5		%N	ONERR	LDA Z	229	E5		%N	VAL	SBC Z
38	26	&	&I	,	ROL Z	102	66	f	&F	COLOR=	ROR Z	166	A6		&N	RESUME	LDX Z	230	E6		&N	ASC	INC Z
39	27	'	'I	,		103	67	g	'F	PLOT		167	A7		'N	RECALL		231	E7		'N	CHR$	
40	28	(	(I	"	PLP	104	68	h	(F	,	PLA	168	A8		(N	STORE	TAY	232	E8		(N	LEFT$	INX
41	29	)	)I	"	AND #	105	69	i	)F	HLIN	ADC #	169	A9		)N	SPEED=	LDA #	233	E9		)N	RIGHT$	SBC #
42	2A	*	*I	(	ROL A	106	6A	j	*F	,	ROR A	170	AA		*N	LET	TAX	234	EA		$N	MID$	NQP
43	2B	+	+I	!		107	6B	k	+F	AT		171	AB		+N	GOTO		235	EB		+N		
44	2C	,	,I	!	BIT	108	6C	l	,F	VLIN	JMP(I)	172	AC		,N	RUN	LDY	236	EC		;N		CPX
45	2D	–	–I	(	AND	109	6D	m	–F	,	ADC	173	AD		–N	IF	LDA	237	ED		–N		SBC
46	2E	.	.I	PEEK	ROL	110	6E	n	.F	AT	ROR	174	AE		.N	RESTORE	LDX	238	EE		.N		INC
47	2F	/	/I	RND		111	6F	o	/F	VTAB		175	AF		/N	&		239	EF		/N		
48	30	0	0I	SGN	BMI	112	70	p	0F	=	BVS	176	B0		0N	GOSUB	BCS	240	F0		0N		BEQ
49	31	1	1I	ABS	AND(I),Y	113	71	q	1F	=	ADC(I),Y	177	B1		1N	RETURN	LDA(I),Y	241	F1		1N		SBC(I),Y
50	32	2	2I	PDL		114	72	r	2F	)		178	B2		2N	REM		242	F2		2N		
51	33	3	3I	RNDX		115	73	s	3F	)		179	B3		3N	STOP		243	F3		3N		
52	34	4	4I	(		116	74	t	4F	LIST		180	B4		4N	ON	LDY Z,X	244	F4		4N		
53	35	5	5I	+	AND Z,X	117	75	u	5F	,	ADC Z,X	181	B5		5N	WAIT	LDA Z,X	245	F5		5N		SBC Z,X
54	36	6	6I	–	ROL Z,X	118	76	v	6F	LIST	ROR Z,X	182	B6		6N	LOAD	LDX Z,Y	246	F6		6N		INC Z,X
55	37	7	7I	NOT		119	77	w	7F	POP		183	B7		7N	SAVE		247	F7		7N		
56	38	8	8I	(	SEC	120	78	x	8F	NODSP	SEI	184	B8		8N	DEF	CLV	248	F8		8N		SED
57	39	9	9I	=	AND Y	121	79	y	9F	NODSP	ADC Y	185	B9		9N	POKE	LDA Y	249	F9		9N		SBC Y.
58	3A	:	:I	#		122	7A	z	:F	NOTRACE		186	BA		:N	PRINT	TSX	250	FA		:N		
59	3B	;	;I	LEN(		123	7B	{	;F	DSP		187	BB		;N	CONT		251	FB		;N		
60	3C	<	<I	ASC(		124	7C		<FF	DSP		188	BC		<N	LIST	LDY X	252	FC		<N		
61	3D	=	=I	SCRN(	AND X	125	7D	}	=F	TRACE	ADC X	189	BD		=N	CLEAR	LDA X	253	FD		=N		SBC X
62	3E	>	>I	,	ROL X	126	7E		>F	PR#	ROR X	190	BE		>	GET	LDX Y	254	FE		>N		INC X
63	3F	?	?I	(		127	7F	RUB	?F	IN#		191	BF		?N	NEW		255	FF		?N		

Screen Column		Key Column		6502 Column	
	I = Inverse F = Flashing N = Normal		C = CONTROL pressed simultaneously U = SHIFT pressed simultaneously		I = Indirect Z = Zero Page # = Immediate A = Accumulator X = X Register Y = Y Register

MICRO™ Data Sheet #6

Apple II

FOCUS ON THE 6809 MICRO WORTH SWITCHING FOR

FOCUS ... ON ITS SUPERIOR HARDWARE

- 6809E Microprocessor provides 8-bit economy with 16-bit capabilities, position independent code and many advanced features
- Commercial quality Keyboard with full UPPER/lower case ASCII, numeric keypad and cursor control keys
- Two double-sided double-density mini diskettes with 640K bytes of IBM-compatible storage
- 56K Bytes User RAM
- Full Video with programmable screen formats, bit mapped graphics, user definable character sets, reverse video and hardware scrolling, plus a light pen interface
- Full Communications support for RS-232 at 50 to 19.2K baud with programmable data formats
- Built-in I/O Drivers include 6 parallel I/O ports, 3 serial I/O ports, 6 timers/counters, 20 mA current loop, and a programmable cassette interface

FOCUS ... ON ITS EXTRAORDINARY SOFTWARE

- Choice of two 'Universal' 6809 Disk Operating Systems supported by many companies: FLEX ™ or OS-9™
- Extended Floating Point Disk BASIC Interpreted and Compiled
- Text Editor and Macro Assembler
- Full feature FlexiMon debugging monitor
- Built-in Word Processor

FOCUS ... ON ITS VERSATILE EXPANDABILITY

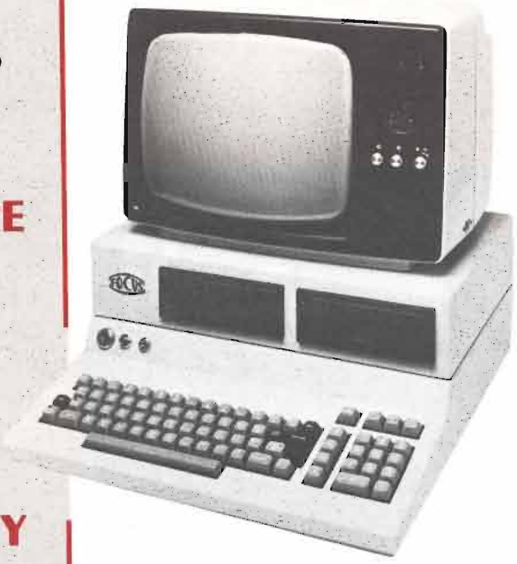
- IEEE-488 Bus Controller option for instrumentation
- Supports multiple 8" diskette drives
- PASCAL, FORTH and other high level programming languages are available
- Complete Application Packages are available from many software sources
- Add-on Hardware includes: Date/Time, Opto-Isolators, Stepper-Motor Controller, A/D and D/A, RAM and EPROM Memory, and others from several manufacturers

FOCUS ... ON YOUR APPLICATION REQUIREMENTS

- FOCUS provides a completely integrated system of hardware and software, so that you can concentrate on your application requirements

*Licensing arrangement for OS-9 pending.
Please contact The COMPUTERIST, Inc. for further details.

THIS IS FOCUS™



$3495^{00}

Very Complete

THE COMPUTERIST®

34 Chelmsford Street
Chelmsford, MA 01824
Phone: 617/256-3649
Telex: 955318 INTL DIV

GET FREE SOFTWARE FOR YOUR COMPUTER!

HOW? JUST ORDER ANY OF THE ITEMS BELOW, AND SELECT YOUR FREE SOFTWARE FROM THE BONUS SOFTWARE SECTION, USING THE FOLLOWING RULE: FOR THE FIRST $100.00 WORTH OF MERCHANDISE ORDERED TAKE 1 ITEM; FOR THE NEXT $200.00 WORTH OF MERCHANDISE ORDERED TAKE ANOTHER ITEM; FOR THE NEXT $300.00 TAKE A THIRD ITEM, ETC. ALL AT NO COST.

HARDWARE by APPLE COMPUTER

APPLE II + . 48K	1199	PASCAL	150
FLOPPY DR. + CNTRLR	545	FLOPPY DRIVE	465
APPLE III. 128K	2999	PILOT	125

We carry the rest of the APPLE line at low. low prices' CALL!

OTHER HARDWARE for APPLE

D.C. HAYES:

Micromodem II	285	Smartmodem	225

MICROSOFT:

Z80 Softcard	269	16K Ramcard	139

MOUNTAIN COMPUTER:

Expansion Chassis	559	Music System	339
A/D + D/A Card	299	CPS Multi-function	169
X/10 Control Card	169	Super Talker	169

CALL FOR MORE PRICES! WE CARRY FULL LINE!

CALIFORNIA COMPUTER SYSTEMS:

Centronics Par. Int.	115	A/D Converter	105
Async. Serial Int.	135	Calendar/Clock	101

CALL FOR MORE PRICES! WE CARRY FULL LINE!

VIOEX:

80 Col. Bd. & Softswitch	235	Enhancer II	125
Enhancer I	105	Softswitch	29

MORE OTHER HARDWARE FOR APPLE:

SSM AIO-II	195	Keybd. Co. Num. Keypad	129
SSM Serial AS 10	115	Sunshine Joystick	39
SSM Par. AP10	99	Game Paddles	29
Novation Apple Cat	319	Shadow/Vet	885
Versawriter Tablet	249	SUP 'R' MOD	29
Prac. Periph. Microbuff (32K)	249	Prac. Periph. Microbuff. (16K)	210

OTHER SOFTWARE for APPLE:

PERSONAL SOFTWARE/VISICORP:

Visicalc 3.3	195	Visifiles	199

CALL FOR MORE PRICES! WE CARRY FULL LINE!

MICROSOFT:

APPLE Fortran (Z80)	129	APPLE Cobol (Z80)	499
TASC Basic Compiler	139	MBASIC Compiler (Z80)	299
TIME Manager	125	MuMath	199
ALDS	99	M/SORT	149

MICRO-PRO:

Wordstar	225	Mail-Merge	99
Spellstar	149	Data-Star	199
Super-Sort	149	Calc-Star	149

PEACHTREE: BIZ Packages, all 199

CPA BIZ Packages, all 195

MORE OTHER SOFTWARE for APPLE:

DB Master	179	DB Master for CORVUS	399
Data Factory 5.0	239	PFS	85
ASCII Express	55	Dakin 5 Deprec. Planner	299
Sorcim Super Calc	189	Dakin 5 BIZ Bookkeeper	299
Howard Tax Prep.	115	Broderbund Payroll	325
Howard Real Estate Anal	129	BPI Accounting Pkgs/ea	325
Synergistic "Data Reporter" Manages, Plots & Edits Data!			189

WORO-PROCESSORS & SPELLERS for APPLE:

Wordstar CP/M	225	EZWriter Prof. Sys.	215
Magic Wand CP/M	299	MUSE Super Text 80	150
Executive Secretary	199	Wordpower	50
Letter Perfect	125	Hebrew II	55
Magic Window	85	Screenwriter II	110
Spellguard	219	Spellstar CP/M	149
Word Handler (Gives 80-Col & Lower-Case with no board!)			199

PRINTERS

EPSON:

MX80	449	MX80 F/T	549
MX100 w/Graftrax	729	MX70 w/Graftrax	285
APPLE Intfce/Cbl	85	GRAFTRAX	60
GRAPPLER Intfc	149	2K Buffer Serial Card	135
MX80 Ribbon	15	MX100 Ribbon	24

C.ITOH:

F-10 Daisy Wheel (Par)	1495	F-10 Daisy Wheel (Ser)	1495
Pro-Writer (Par/Ser)	599	Pro-Writer (Par)	499
F-10 Tractor Option	225	Printer Interfaces	CALL

NEC:

PC-8023A	495	NEC 7710 Daisy	2345

QUME:		**DIABLO:**	
SPRINT 9/45	1995	630 R/O	2099

OKIDATA:

Microline 82A	495	Microline 80	375
Microline Tractor	59	Okigraph I	79
Microline 83A	799	Microline 84 (Par)	1099

IDS:

560 with graphics	1095	Prism-Print Software	49
Prism 80 (Basic)	899	Prism 132 (Basic)	1050
Auto Sheet Feed	125	Prism Color	325
Sprint Mode (200 cps)	125	Dot Plot Graphics	85

ADD-ON MEMORY CARDS & DISK DRIVES FOR APPLE

MEMORY:

Microsoft 16K Ramcard	139	Saturn 32K Card	199
Legend 128K Ramcard	649	Saturn 64K Card	369
Legend 64K Ramcard	299	Saturn 128K Card	525
SVA 256K APL-Cache	1045	Prometheus 128K	439
AXLON 320K Ram Disk	1149	16K of 4116. 200 NS Mem.	25

APPLE-COMPATIBLE FLOPPIES by MICRO-SCI:

With Controller:		No Controller:	
A35 Exact Replacement	460	A35 Exact Replacement	415
A40 40-Track	489	A40 40-Track	399
A70 70-Track	599	A70 70-Track	499

8" FLOPPY DISK SYSTEMS:

Vista Dual SSDD	1299	Vista Dual DSDD	1599
SVA AMS8000 Dual SSDD	1945	SVA AMS8000 Dual DSDD	2595
SVA ZVX4 Quad Cntrllr.	495	SVA Disk 2 + 2 Cntrllr.	359

CORVUS HARD DISKS:

6 MB Hard Disk	2249	Apple Interface	175
11 MB Hard Disk	3945	Other Computer Intfce	CALL
20 MB Hard Disk	4769	Mirror Back-Up	675

MONITORS, PLOTTERS & PERIPHERALS

MONITORS:

Zenith 12" Green	125	Zenith 13" Color	359
Amdek 12" Green	135	Amdek 13" Color	359
BMC 12" Green	119	BMC 12" Color	349
Electrohome RGB Intfce	275	Electrohome RGB 13" Clr	725

PLOTTERS:

Watanabe 1-Pen	1150	Watanabe 6-Pen	1400
Strobe Plotter 1-Pen	699	Strobel Apple Intfce	99
Houston Inst. DMP-3	929	Houston Inst. DMP-4	1185

OTHER PERIPHERALS:

Scott Shadow/VET	885	Street Echo II Synth.	189
Votrax Type-N'-Talk	325	Computer Sta. Dithertizer	269
ALF 9-Voice Music Bd.	155	Comp. Sta. Video Camera	375
ALF 3-Voice Music Bd.	199	Comp. Stat. Both above	599

COMPUTER SYSTEMS

ATARI

ATARI 800 (16K)	629	ATARI 400 (16K)	335
810 Disk Drive	449	825 Printer	599
16K Ram Memory	89	850 Interface	159
Microsoft Basic	69	830 Modem	149
INTEC 32K Ram Memory	129	Telelink Cartridge	25
ATARI 800 (48K)	739	AXLON 128K Ram Disk	550
		ATARI 400 (48K)	435

OSBORNE

Osborne 1	1695	Printer Cable	55

XEROX

820-1 System w/5" Dr.	2450	820-2 System w/8" Dr.	2950
CP/M Op. Sys.	159	Wordstar	419
Super Calc	199	DIABLO 630 Printer	2099
Systems-Plus BIZ. SOFTWARE, PER MODULE			CALL!

NEC

Full Line at Low. Low Prices! CALL!

COMMODORE VIC

VIC 20 Computer	259	VIC 1515 Printer	335
VIC 1540 Disk Drive	499	VIC 1211A Supr-Expndr	59
VIC 1530 Datasette	69	VIC 1011A RS-232 Port	45

GENERAL CP/M SOFTWARE

MICROSOFT:

Basic 80	275	Edit 80	139
Basic Compiler	299	Mu Math/Mu Simp	199
Fortran 80	339	Mu Lisp/Mu Star	159
Cobol 80	499	M-Sort	128
Macro 80	139	Z-80 Softcard/Apple	269

MICROPRO:

Wordstar	275	Calcstar	199
Mailmerge	89	Supersort	165
Spellstar	165	Custom. Notes	275
Datastar	239		

ASHTON-TATE:

dBASE II	475	dBASE II Guide	29

SORCIM:

Super Calc	189		

FOX-GELLER:

Ouickscreen	129	dUTIL	69
Quickcode (Writes programs for dBASE II)			199

ISA:

Spellguard	219	SP/LAW	99

PEACHTREE:

Gen. Ledger	399/40	Inventory	399/40
Acct. Rec.	399/40	Magicalc	269/25
Acct. Pay.	399/40	Sales Invoicing	399/40
Payroll	399/40		

SYSTEMS PLUS

All Modules	415/EA		

S-100 BOARDS

HAYES S-100 MODEM	325	HAYES SMARTMODEM	219
HAYES CHRONOGRAPH	189	CCS 64K RAM BOARD	525

FLOPPY DISKS

Elephant 5.25"		Elephant 8"	
Soft SSSD/bx	25	Soft SSSD/6x	29
3M 5" Soft SSSD/bx	27	3M 8" Soft SSSD/bx	32
Maxell 5" Soft SSSD/bx	31	Maxell 8" Soft SSSD/bx	35
DYSAN 5" Soft SSSD/bx	37	DYSAN 8" Sof SSSD/bx	49

BONUS SOFTWARE SECTION!

Let us acquaint you with MESSAGE-MAKING SOFTWARE. Just place the disk in the APPLE, enter the text, and colorful, dynamic messages appear on the screens of TV sets connected to the computer. Use the software to broadcast messages on TV screens in schools, hospitals, factories, store window, exhibit booths, etc. The following program is our latest release:

SUPER MESSAGE: Creates messages in full-page "chunks". Each message allows statements of mixed typestyles, typesizes and colors, in mixed upper and lower case. Styles 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 and 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.
REOUIRES 48K & ROM APPLESOFT . . . . $50.

APPLE PLOTS YOUR DATA & KEEPS YOUR RECORDS TOO! APPLE DATA GRAPH 2.1: Plots up to 3 superimposed curves on the Hi-res Screen both 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!
REOUIRES 48K & ROM APPLESOFT . . . . $35.

APPLE RECORD MANAGER: Allows complex 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 of files can be printed, if desired. Additional modules coming are a STATISTICS INTERFACE, CHECKBOOK, MAILING LIST & DATA-ENTRY.
REOUIRES 48K & ROM APPLESOFT . . . . . $40.

APPLE LITERATURE DATABASE: Allows rapid retrieval (via keywords) of references from total APPLE literature thru 1980, on 5.25" disk. Each entry in the data base consists of the article, author-name, periodical-name, date of issue, & page nos. The database is intended to support large magazine files which would require lengthy manual searching to recover information. Annual updates will be available.
REQUIRES 48K & ROM APPLESOFT . . . . $60.

WORDPOWER: Is a simple, powerful, low cost, line-oriented word-processor program. It offers a fast machine language FIND & REPLACE. Text can be listed to screen or printer, with or without line-numbers. Lower-case adaptors are supported. You can merge files, move groups of lines, and easily add, change, or delete lines. WORDPOWER can be used to create and maintain EXEC files. It can also be used as a rapid, unstructured, information-storage and retrieval system via its rapid search capabilities.
REQUIRES 48K & ROM APPLESOFT . . . . $50.

LABELMAKER: Allows users to quickly create address labels. A given label may be generated in any quantity from 1 to 32767. Space is allowed on labels for a personal and company name, but the space is automatically closed up if only a personal name is entered. Space is also allowed for foreign countries. The program can also generate labels for price-tags, part numbers and mail-messages such as "RUSH", "FRAGILE", etc. A self-incrementing feature allows theatre-tickets to be produced, with a date, and numbers running from a000 to z999. An editor is provided for editing labels prior to printing. All labels may be saved to disk for instant recall.
REQUIRES 48K & ROM APPLESOFT . . . . $35.

Above software for APPLE DOS 3.2/3.3 only. Call for BONUSES for other systems.

TO ORDER: Use phone or mail. We accept VISA, MASTERCARD, COD's, personal checks & money orders. Add 4% for credit card. Customer pays handling on COD orders. Foreign orders must be in American Dollars & include 10% for handling. Connecticut residents add 7.5% sales tax. Prices subject to change without notice.

Not responsible for typographical errors. Prices subject to change without notice.

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

Advertiser's Index

MICRO INK is not responsible for claims made by its advertisers. Any complaint should be submitted directly to the advertiser. Please also send written notification to MICRO.

Next Month in MICRO

August: Programming Techniques

- **Structured Programming in 6502 Assembly Language —** This discussion of structured programming demonstrates how you can use it to improve and simplify your programming methods.

- **Pattern Matching with the 6502 —** Pattern matching algorithms attempt to find one or more occurrences of a given character string in a specified range of memory. Text editing is a common application. This article presents elementary and advanced algorithms.

- **Random Number Generator in Machine Language —** This simple routine can be easily implemented in a machine-language program whenever random numbers are needed.

- **New Character Set for VIC-20 —** This technique involves altering the character ROM pointer to point to RAM, copying characters from ROM, and defining your own characters, pixel-by-pixel!

Plus...

PET/AIM Connection
PET to AIM Download
VIC Duty Cycle Monitor
On Error GOTO (OSI)
Straightforward Garbage Collection for the Apple

And Our Regular Columns and Departments...

Apple Slices
PET Vet
Reviews in Brief
Short Subjects
Tech Data Sheet
And more!

20% OFF

Your money goes farther when you subscribe. During the course of a year, when you subscribe, you save 20% (in the U.S.).

Pay only $24.00 ($2.00 a copy) for 12 monthly issues of MICRO sent directly to your home or office in the U.S.

More MICRO for Less Money When You Subscribe

But on the newsstand — if you can locate the issue you want — you pay $30.00 a year ($2.50 a copy).

Special Offer — Subscribe for 2 years ($42.00) and get 30% off the single issue price.

Subscribe to MICRO today.

MICRO
34 Chelmsford Street
P.O. Box 6502
Chelmsford, MA 01824

Please send me MICRO for __ 1 year __ 2 years
NOTE: Airmail subscriptions accepted for 1 year only.

Check enclosed $________
Charge my ________ VISA account
________ Mastercard account

No. ________________________

Expiration date ________________________

Name ________________________

Address ________________________

____

City/State ________________ Zip ____

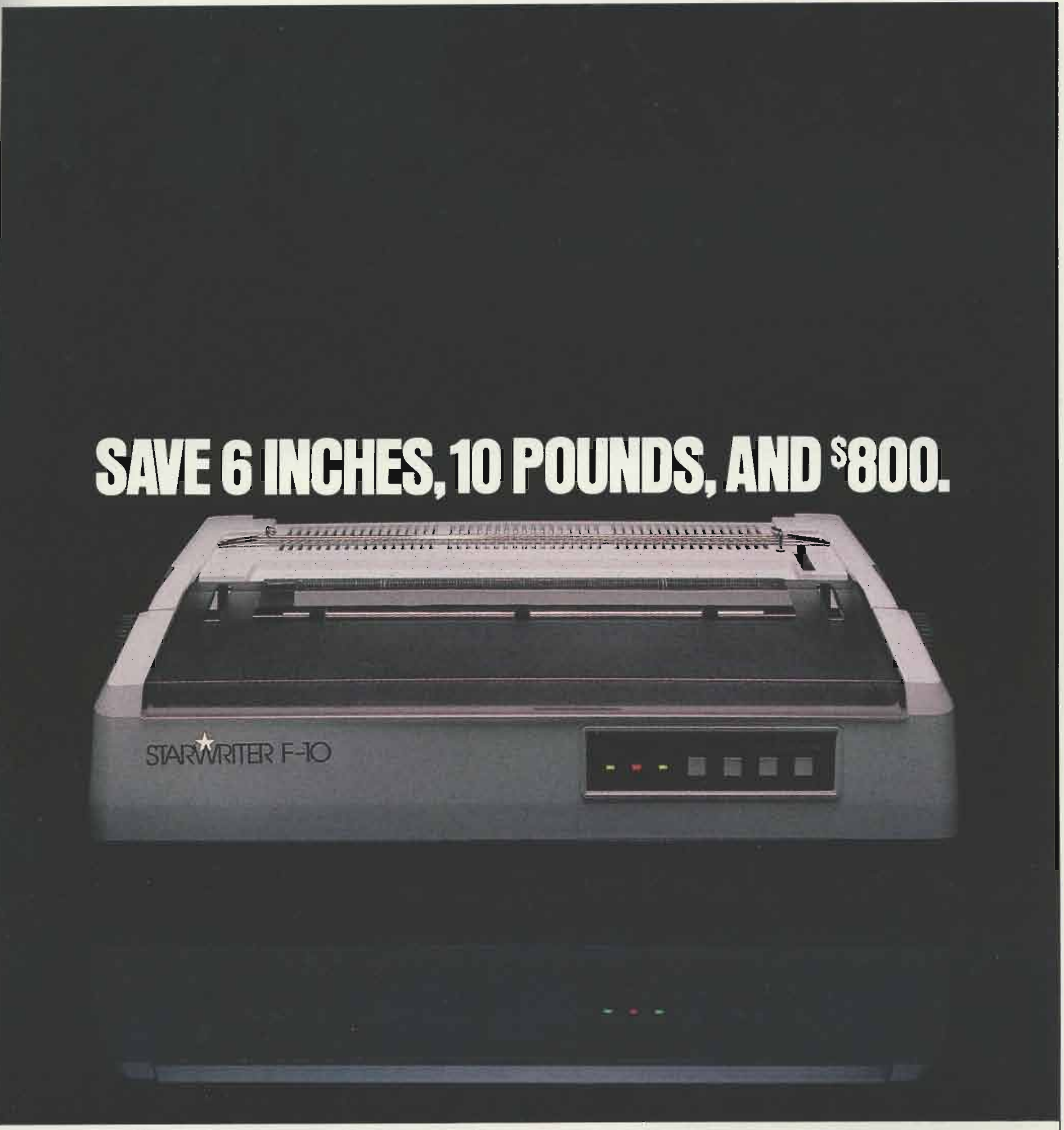
Subscription Rates Effective January 1, 1982

Country	Rate
United States	$24.00 1 yr. 42.00 2 yr.
Foreign surface mail	27.00
Europe (air)	42.00
Mexico, Central America, Mid East, N. & C. Africa	48.00
South Am., S. Afr., Far East, Australasia, New Zealand	72.00

* Airmail subscriptions accepted for only 1 year. For U.S. and Canadian 2-year rates, multiply by 2.

Job Title: ________________________

Type of Business/Industry: ________________________



On the new, slicked-up, trimmed-down Starwriter F-10.

It's C. Itoh's latest generation of letter-quality printers.

It cranks out flawless copy at 40 cps; and its full 15" carriage lets it double in brass for both letter processing and business applications. You can plug it into almost any micro on the market (serial or parallel) simply by plugging it in. And then make it keep on trucking with inexpensive, easily available Diablo compatible daisy wheels and ribbons.

In its serial mode, it can print just about anything (including boldface, underlines, subscripts and superscripts), and snap the carriage back to start the next line in less than a second. In its line mode, it prints in both directions, for even faster throughput.

(While making about as much noise as a cat walking on Kleenex.)

It's a nice, portable 30 pounds—about 10 pounds lighter than the Starwriters before it. And it stands exactly as tall (or precisely as small) as a dollar bill.

Speaking of which:

Incredibly, the Starwriter F-10 sells for about the same preposterously low price as its predecessors. Which is to say, about $800 less than a lot of other printers that don't even come close to measuring up. Or even better...

Measuring down.

Distributed Exclusively by Leading Edge Products, Inc., 225 Turnpike Street, Canton, Massachusetts 02021. Call: toll-free 1-800-343-6833; or in Massachusetts call collect (617) 828-8150. Telex 951-624.

LEADING
EDGE.™

HUNTINGTON COMPUTING

Apple®

ADVANCED OPERATING SYSTEMS

#9483	Apple-Aids	$44.89
#9482	Mostly Basic/Household	$26.89
#9481	Mostly Basic/Interface-Scientific	$26.89
#9480	Voyage of the Valkyrie	$35.89

ARTSCI

#6061	Basic Mailer	$59.49
#6060	Magic Window	$84.99
#6062	Magic Words	$62.89

AURORA

#4801	Executive Secretary	$212.49
#4803	Gradebook	$51.00
#4804	Hebrew II	$50.99
#4811	Hebrew II	$79.19
#4805	Quicktrace	$43.99

AVALON HILL

#2058	Major League Baseball	$25.49
#2052	North Atlantic Convoy (Cassette)	$12.69
#2060	Empire of the Overmind (Cassette)	$25.49
#2059	Empire of the Overmind	$29.69
#2061	Tanktics	$24.59

BEAGLE BROS.

#3257	Alpha Plot	$34.69
#3250	Dos Boss 3.3	$20.39
#3255	Dos Boss 3.2	$20.39
#3251	Game Pack #1	$20.39
#3252	Game Pack #2	$20.39
#3253	Game Pack #3	$20.39
#3254	Game Pack #4	$20.39
#3258	Tip Disk	$17.99
#3256	Utility City	$25.89

BLACKBURG

#9160	Active Filter Design	$35.89
#9162	Electronics I	$35.89
#9163	Electronics II	$35.89
#9164	Electronics III	$35.89
#9165	Video Display Graphics	$35.89

BRODERBUND

#1958	Alien Rain	$21.19
#1960	Alien Typhoon	$21.19
#1967	Arcade Machine	$39.49
#1966	David s Midnight Magic	$30.69
#1952	Galactic Empire	$22.99
#1951	Galaxy Wars	$22.99
#2000	Genetic Drift	$25.39
#1957	Golden Mountain	$16.89
#1961	Payroll	$335.69
#1949	Red Alert	$22.99
#1965	Space Quarks	$25.39
#1964	Space Warrior	$21.19
#1956	Tank Command	$12.99
#1955	Tawala s Last Redoubt	$25.39
#1959	Track Attack	$26.89

HARDWARE

#8035	Apple Clock (Mountain)	$246.39
#8200	Applesoft Renumberand Merge in ROM (Soft Control)	$33.89
#8206	Applesoft Utility ROM (Soft Control)	$33.89
#747	B S R Control (Novation)	$16.69
#2107	Basic Compiler (Microsoft)	$335.69
#8205	Basic in ROM (Soft Control)	$33.89
#8039	Blitz Bug (Omni)	$21.89
#8025	C P S Pascal (Mountain)	$16.99
#86	Cable Adaptor (Centronics)	$35.19
#24	Character Set Plus (Lazer)	$22.39
#6534	Cassette Control Device (Hartley)	$67.95
#9043	Data Link (Link)	$87.99
#217	DISASM/65 (Lazer)	$26.29
#2362	Dithertizer II with Camera (Computer Station)	$639.99
#8203	Dual DOS ROMS (Soft Control)	$46.69
#7951	E Z Port (Versa)	$21.19
#748	Expansion Mod (Novation)	$33.15
#1476	Game Paddles (TG)	$33.89
#1070	Joyport (Sirius)	$67.39
#12	Joystick (Mimco)	$59.95
#322	Large Character Eprom (Videx)	$34.29
#27	Lower Case Adaptor Rev. 7 (Paymar)	$29.99
#9021	Memory Expansion Module (Prometheus)	$148.69
#8060	Micro-Verter (ATV)	$31.49
#749	Modem Handset (Novation)	$24.65
#6240	Paddle-Adapple (Southern California)	$26.89
#39	560G Paper Tiger (IDS)	$1,225.00
#8020	MX100 Ft. Printer (Epson)	$749.00
#725	Pro Paddle (Rainbow)	$44.89
#35	Ram Card (Microsoft)	$169.00
#16	16K Ram Expansion Board (Andromeda)	$169.00
#325	Soft Switch (Videx)	$25.49
#36	Z-80 Softcard (Microsoft)	$299.00
#860	Super Fan II (R. H. Electronics)	$64.99
#31	Switchplate (Videx)	$17.99
#59	Vision 80 Card (Vista)	$297.49

MICRO LAB

#2253	Crown of Arthain	$29.69
#2250	Dogfight	$25.99
#2254	Mad Venture	$21.19
#2259	Palace in Thunderland	$21.19
#2263	Roach Hotel	$30.69

Atari®

ADVENTURE INTERNATIONAL

#239	Angle Worms	$12.64
#235	Galactic Empire	$16.74
#234	Lunar Lander	$12.64
#233	Sunday Golf	$12.64

ARTSCI

#204	Cypher Bowl	$43.94
#203	Gomoku (Cassette)	$16.94
#201	Poker Solitaire	$12.74
#202	Reversi	$16.94

ARTWORX

#834	Giant Slalom	$17.54
#823	Intruder Alert (Cassette)	$14.84
#822	Intruder Alert	$18.34
#825	Rings of the Empire (Cassette)	$14.84
#824	Rings of the Empire	$18.34
#832	Stud Poker	$16.64

ATARI

#285	Black Jack	$12.64
#276	Mailing List	$16.94
#280	Missile Command	$35.14
#273	Space Invaders	$16.94
#277	Star Raiders	$33.94

AVALON HILL

#302	Empire of the Overmind (Cassette)	$25.44
#301	Empire of the Overmind	$29.64
#308	North Atlantic Convoy (Cassette)	$12.74
#307	Nuke War (Cassette)	$12.74
#306	Planet Miners (Cassette)	$12.74
#304	Tanktics (Cassette)	$20.34
#311	Tanktics	$24.64

EDU-WARE

#385	Compu Math/Decimals (Cassette)	$26.34
#384	Compu Math Decimals	$35.14
#383	Compu Math/Fractions (Cassette)	$26.34
#382	Compu Math Fractions	$33.94
#381	Compu-Read	$25.44

EPYX

#409	Crush, Crumble & Chomp (Cassette)	$26.34
#408	Crush, Crumble & Chomp	$26.34
#407	Datestones of the Ryn	$17.54
#403	Invasion Orion (Cassette)	$21.14
#406	Invasion Orion	$21.94
#401	Rescue at Rigel (Cassette)	$25.44
#404	Star Warrior (Cassette)	$33.94
#405	Star Warrior	$33.94
#412	Temple of Apshai (Cassette)	$35.14
#400	Temple of Apshai	$35.14

MISCELLANEOUS

#560	Ali Baba & the 40 Thieves (Quality)	$28.94
#861	Andromeda (Gebelli)	$35.94
#761	Casino Blackjack (Manhattan)	$21.94
#763	Concentration (Manhattan)	$13.14
#466	Crossfire (On-Line)	$26.94
#746	Dominoes (Cassette) (Creative Computing)	$10.44
#721	Galactic Chase (Cassette) (Spectrum)	$21.94
#720	Galactic Chase (Spectrum)	$26.34
#760	Gin Rummy 3.0 (Manhattan)	$21.94
#740	Hail to the Chief (Creative Computing)	$21.94
#921	Kayos (Cassette) (Computer Magic)	$30.74
#920	Kayos (Computer Magic)	$30.74
#762	Labyrinth Run (Manhattan)	$13.14
#860	Match Racers (Gebelli)	$26.34
#862	Pathfinder (Gebelli)	$30.74
#748	Pool (Cassette) (Creative Computing)	$10.44
#322	Safari (Cassette) (CDS)	$25.44
#321	Safari (CDS)	$33.94
#1000	Shadow Hawk One (Horizon)	$44.94
#900	Shattered Alliance (Strategic Simulations)	$35.14
#567	Tank Trap (Cassette) (Quality)	$10.14
#568	Tank Trap (Quality)	$12.64
#561	Tari Trek (Cassette) (Quality)	$10.14
#562	Tari Trek (Quality)	$12.64
#461	The Wizard & The Princess (On-Line)	$27.94
#800	Yahtman (Compu-Core)	$17.54

COMING SOON!

Hodge Podge for the Atari®

Apple®

MISCELLANEOUS

#2182	Adventure in Time (Phoenix)	$25.39
#2301	Apple World (USA)	$52.99
#7890	Applecillin (XPS)	$33.89
#3700	Asteroid Field (Cavalier)	$22.49
#2700	Astroscope (AGS)	$26.49
#2180	Birth of the Phoenix (Phoenix)	$35.89
#2407	Bishop s Square (Datasoft)	$25.39
#1601	Bloody Murder (Stoneware)	$17.99
#3000	Bookkeeper (Delta)	$76.39
#3702	Bug Attack (Cavalier)	$26.29
#9700	Castles of Darkness (Logical)	$30.69
#3001	Checkwriter (Delta)	$33.99
#9860	Chinese Lessons (CTI)	$26.09
#7851	Cosmo-Mission (Cosmos)	$21.19
#9241	The Count (Pear)	$22.39
#2752	Creative Financing (Howardsoft)	$127.49
#2205	Creature Venture (Highland)	$21.19
#4401	Cyborg (Sentient)	$28.99
#1607	D B Master Version 3 (Stoneware)	$194.59
#4250	Falcons (Piccadilly)	$25.39
#7878	Financial Partner (Denver)	$220.49
#6080	Firebird (Gebelli)	$25.39
#3954	Form Letter (S. Sorcery)	$26.99
#7852	Hungry Boy (Cosmos)	$21.19
#3451	Interlude (Syntonic)	$16.99
#9840	Jabbertalky (Mind Toys)	$26.29
#3150	The Landlord (Min Micro)	$649.99
#9242	The Liberator (Pear)	$26.89
#2406	Lisp Interpreter (Datasoft)	$106.19
#3953	Mailing List (S. Sorcery)	$44.89
#9240	The Manipulator (Pear)	$31.39
#4552	Market Analyzer (RTR Software)	$224.99
#4550	Market Charter (RTR Software)	$110.39
#9100	Memory Page Editor (Compu-tron)	$44.89
#9580	The Menu (C & H Video)	$26.29
#9380	Menu Generator (Crane)	$35.09
#1804	Monty Plays Monopoly (Visicorp)	$31.99
#2211	Mummy s Curse (Highland)	$26.39
#2200	Oldorf s Revenge (Highland)	$16.89
#4400	OO-Topos (Sentient)	$27.99
#7870	Pascal Programmer (Denver)	$105.99
#7871	Pascal Tutor (Denver)	$105.99
#7650	Pornopoly (CCI)	$25.39
#3484	Pot of Gold I (Rainbow)	$33.89
#3485	Pot of Gold II (Rainbow)	$33.89
#2751	Real Estate Analyzer (Howardsoft)	$127.49
#6600	Rubik s Cube (Software Alternative)	$16.89
#6081	Russki Duck (Gebelli)	$31.39
#1150	Sargon II (Hayden)	$29.79
#2701	Sexoscope (AGS)	$26.49
#8901	Snack Attack (Datamost)	$25.39
#8520	Solitaire (Computek)	$25.39
#2302	Space Raiders (USA)	$25.39
#2306	Star Dance (USA)	$21.89
#3701	Star Thief (Cavalier)	$25.39
#7163	Star Venture (Compu-Thing)	$26.89
#9180	Starship Commander (Voyager)	$35.89
#4251	Suicide (Piccadilly)	$26.29
#3955	Text Editor (S. Sorcery)	$58.39
#8901	Thief (Datamost)	$25.39
#7853	Thunderbirds GX (Cosmos)	$21.19
#7102	Trickshot (IDSI)	$35.09
#1606	Utility Pak #1 (Stoneware)	$21.19
#4100	Visicard (Data S. C.)	$29.69
#1820	VisiCalc 3.3 (Visicorp)	$199.00
#1805	VisiFile (Visicorp)	$212.49
#9740	Vocanoes (Earthware)	$43.49
#2181	Zoom Grafix (Phoenix)	$35.89
#7880	Zork I (Infocom)	$33.89
#7881	Zork II (Infocom)	$33.99

SIRIUS

#1058	Autobahn	$25.39
#1068	Beer Run	$19.99
#1073	Borg	$26.89
#1074	Computer Foosball	$25.39
#1065	Copts & Robbers	$29.69
#1051	Cyber Strike	$33.89
#1067	Dark Forest	$19.99
#1060	Gamma Goblins	$25.39
#1061	Gorgon	$33.89
#1069	Hadron	$29.69
#1057	Pulsar II	$25.39
#1071	Snake Byte	$25.39
#1064	Sneakers	$25.39
#1056	Space Eggs	$19.99
#1050	Star Cruiser	$20.99
#1072	Twerps	$26.89

SYNERGISTIC

#1220	Adventure to Atlantis	$35.99
#1216	Data Reporter	$186.99
#1202	Dungeon Campaign	$14.79
#1210	Escape from Arcturus	$25.39
#1214	The Linguist	$33.99
#1200	Odyssey	$25.39
#1207	Program Line Editor	$33.99
#1201	Wilderness Campaign	$18.49

Call Toll-Free 800-344-5106 (outside California)

HUNTINGTON COMPUTING
Post Office Box 1297
Corcoran, California 93212

Foreign Orders 209-992-4481
In California 800-692-4146

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

Outside Calif. 800-344-5106

We take **MasterCard, American Express or VISA** (Include card # and expiration date). California residents add 6% tax. Include $2.00 for postage. Foreign and hardware extra. Foreign (excluding Canada): remit U.S. currency, checks on U.S. banks, use listed charge cards, or make direct wire transfers through Security Pacific Bank, Corcoran, for a $6.00 charge. All overseas orders shipped by air. Send for free catalog. Prices subject to change without notice.