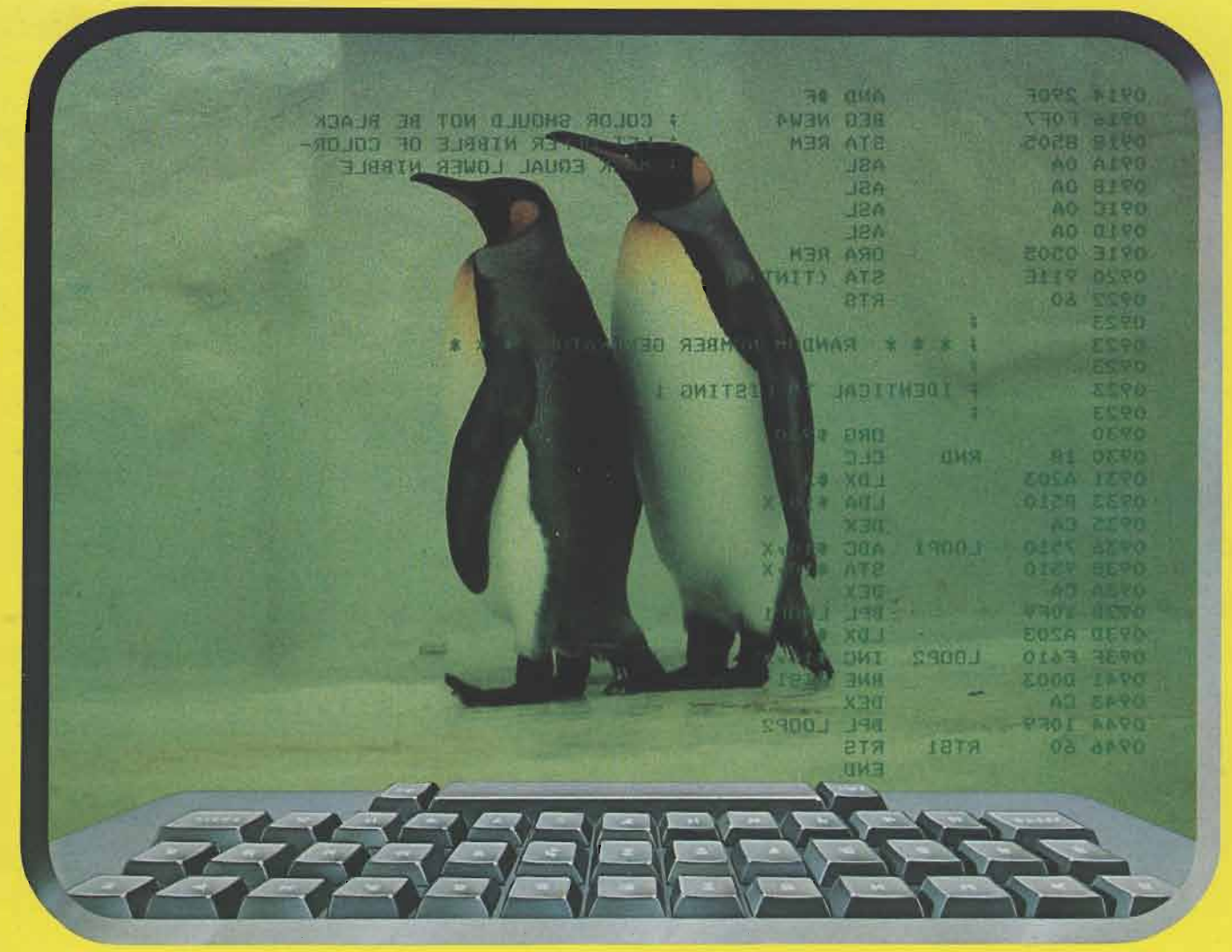
NO. 51 AUGUST 1982

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

MICRO™

THE 6502/6809 JOURNAL



Programming Techniques Feature

Apple Garbage Collection

OSI Extended Parser

POWER-Aid for PET

0 74470 11690 0

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

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

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

16K ON A PLUG-IN CARD.

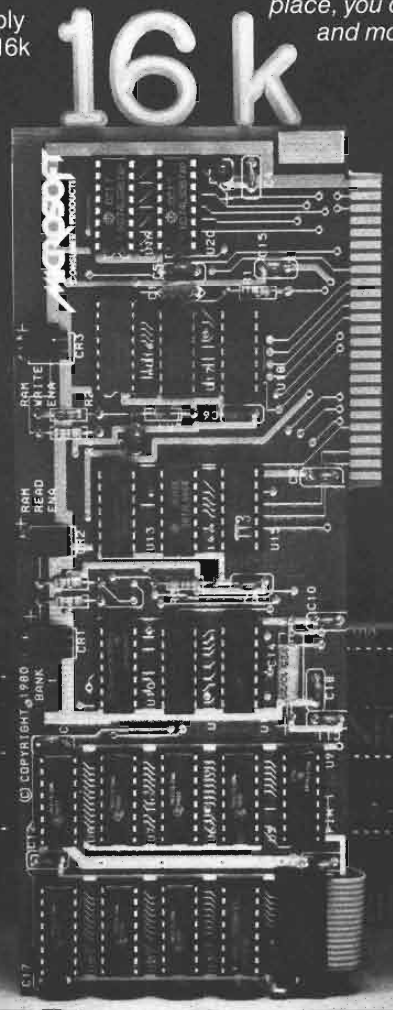
Microsoft's new RAMCard simply plugs into your Apple II,® and adds 16k bytes of dependable, buffered read/write storage.

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

GREAT SOFTWARE: YOURS, OURS, OR THEIRS.

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

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



JOIN THE SOFTCARD FAMILY.

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

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

Microsoft Consumer Products
10700 Northup Way
Bellevue, WA 98004
206-828-8080

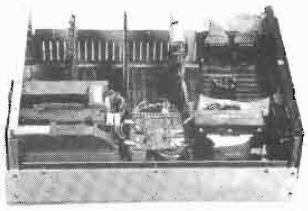
SoftCard, RAMCard and Microsoft are trademarks of Microsoft, Inc. Apple II is a registered trademark of Apple Computer Inc. Z-80 is a registered trademark of Zilog, Inc. CP/M is a registered trademark of Digital Research Inc. VisiCalc is a registered trademark of Personal Software, Inc. Microsoft Consumer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

MICROSOFT™

FLEX - OS-9 LEVEL ONE - UNIFLEX - OS-9 LEVEL TWO

ONLY GIMIX Systems can be configured to run any of these.

GIMIX systems utilize the most powerful 6809 operating systems: FLEX, UniFLEX, OS-9 LEVEL ONE and TWO -- the systems the PROs use. This means a wide selection of software to choose from as well the ability to develop sophisticated, multi-user/multi-tasking programs on your **GIMIX** System.



The GIMIX CLASSY CHASSIS™ consists of a heavy-weight aluminum mainframe cabinet which provides more than ample protection for the electronics and 1 or 2 optional 5¼" drives.

Backpanel connectors can be added for convenient connection of terminals, printers, drives and other peripherals.

A 3 position locking keyswitch enables users to disable the front panel reset button to prevent accidental or unauthorized tampering with the system.

The **GIMIX** system mother board provides fifteen 50 pin slots and eight 30 pin I/O slots -- the most room for expansion of any SS50 system available. The on board baud rate generator features 11 standard baud rates, 75 to 38.4K, for maximum versatility and compatibility with other systems. Extended address decoding allows the I/O block to be addressed anywhere in the 1 megabyte address space. All components feature Gold plated connectors for a lifetime of solid connections. All boards are fully buffered for maximum system expansion.

Each **GIMIX** Mainframe System is equipped with an industrial quality power supply featuring a **ferro-resonant constant voltage transformer** to insure against problems caused by adverse power input conditions such as A.C. line voltage fluctuations etc. The supply provides 8 volts at 30 amps and plus or minus 16 volts at 5 amps, more than enough capacity to power a fully loaded system and two internal drives.

The 2MHz GIMIX 6809 PLUS CPU board includes a time of day clock with battery back-up and 6840 programmable timer to provide the programmer with convenient, accurate time reference. Later addition of 9511 or 9512 arithmetic processors is provided for on the board. The unique **GIMIX** design enables software selection of either OS-9 or FLEX, both included in many complete **GIMIX** systems.

GIMIX STATIC RAM boards require no complicated refresh timing cycles or clocks for data retention. GIMIX memory boards are guaranteed for 2 MHz operation with no wait state or clock stretching required.

Our low power NMOS RAM requires less than 3/4 amp at 8V for a fully populated 64K board. For critical situations, our non-volatile 64K byte CMOS static RAM boards with built in battery back-up retain data even with system power removed. A fully charged battery will power this board for a minimum of 21 days. A write protect switch permits CMOS boards to be used for PROM/ROM emulation and software debugging.

The GIMIX DMA controller leaves the processor free to perform other tasks during disk transfers - an important feature for multi-user/multi-tasking systems where processor time allocation is critical. The DMA board will accomodate up to 4 drives 5¼" or 8" in any combination running single or double density single or double headed. Programmed I/O Disk Controllers are also available.

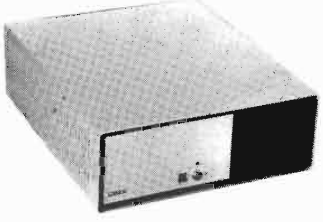
GIMIX systems are designed with ultimate **RELIABILITY** in mind. You can choose from the below featured systems or select from our wide variety of components to build a custom package to suit your needs.

GIMIX 2MHz 6809 System including: CLASSY CHASSIS, 6809 PLUS CPU BOARD, 56KB STATIC RAM, 2 SERIAL PORTS W/CABLES, GMXBUG MONITOR, FLEX, and OS-9 LEVEL 1 . . . . . **$3248.49**
FOR TWO 5¼" 40 TRACK DSDD DRIVES ADD . . . . . **$ 900.00**
GIMIX 128KB WINCHESTER SYSTEM including: CLASSY CHASSIS, 6809 PLUS CPU BOARD, 128KB STATIC RAM, 4 SERIAL PORTS W/CABLES, 5¼" 80 TRACK DSDD FLOPPY DISK DRIVE, 19MB 5¼" WINCHESTER HARD DISK, OS9 LEVEL 2, EDITOR AND ASSEMBLER **$8998.09**

50HZ Versions Available, 8" Drives Available — Contact GIMIX for Prices and Information.

The Sun Never Sets On A GIMIX!

GIMIX users are found on every continent, including Antarctica. A representative group of GIMIX users includes: **Government Research and Scientific Organizations** in Australia, Canada, U.K. and in the U.S.; NASA, Oak Ridge, White Plains, Fermilab, Argonne, Scripps, Sloan Kettering, Los Alamos National Labs, AURA. **Universities:** Carleton, Waterloo, Royal Military College, in Canada; Trier in Germany; and in the U.S.; Stanford, SUNY, Harvard, UCSD, Mississippi, Georgia Tech. **Industrial users** in Hong Kong, Malaysia, South Africa, Germany, Sweden, and in the U.S.; GTE, Becton Dickinson, American Hoechst, Monsanto, Allied, Honeywell, Perkin Elmer, Johnson Controls, Associated Press, Aydin, Newkirk Electric, Revere Sugar, HI-G/AMS Controls, Chevron. **Computer mainframe and peripheral manufacturers,** IBM, OKI, Computer Peripherals Inc., Qume, Floating Point Systems. **Software houses;** Microware, T.S.C., Lucidata, Norpak, Talbot, Stylo Systems, AAA, HHH, Frank Hogg Labs, Epstein Associates, Softwest, Dynasoft, Research Resources U.K., Microworks, Meta Lab, Computerized Business Systems.



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

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.

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

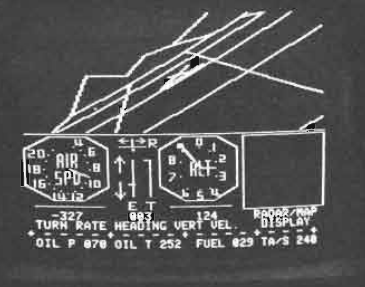
GIMIX inc.

The Company that delivers

Quality Electronic products since 1975.

© 1982 GIMIX Inc.

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



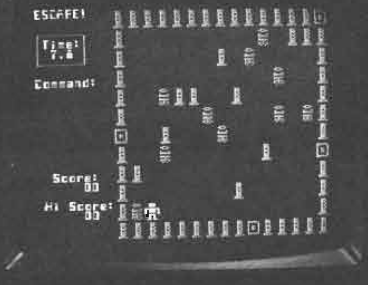
A2-FS1

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

NO. 51 AUGUST 1982

MICRO™

THE 6502/6809 JOURNAL

STAFF

President/Editor-in-Chief
ROBERT M. TRIPP

Publisher
MARY GRACE SMITH

Senior Editor
LAURENCE KEPPLE

Editorial Staff
PHIL DALEY — Technical editor
JOHN HEDDERMAN — Jr. programmer
MARJORIE MORSE — Editor
JOAN WITHAM — Editorial assistant
LOREN WRIGHT — Technical editor

Graphics Department
HELEN BETZ — Director
PAULA M. KRAMER — Production mgr.
EMMALYN H. BENTLEY — Typesetter

Sales and Marketing
CATHI BLAND — Advertising mgr.
CAROL A. STARK — Circulation mgr.
LINDA HENSDILL — Dealer Sales
MAUREEN DUBE — Promotion

Accounting Department
DONNA M. TRIPP — Comptroller
KAY COLLINS — Bookkeeper
EILEEN ENOS — Bookkeeper

Contributing Editors
DAVE MALMBERG
JIM STRASMA
RICHARD VILE

Advertising Sales Representative
KEVIN B. RUSHALKO
(603) 547-2970

Subscription/Dealer inquiries
(617) 256-5515

DEPARTMENTS

5 Editorial
7 News
8 Letters/Updates
39 Reviews in Brief
67 PET Vet
89 It's All Ones and Zeros
108 Apple Slices
112 Software Catalog
116 Hardware Catalog
118 6809 Bibliography
120 6502 Bibliography
123 Data Sheet
127 Advertiser's Index
128 Next Month in MICRO

PROGRAMMING TECHNIQUES

43 **Structured Programming in 6502 Assembly Language** ..... *Kim G. Woodward*
Add high-level, structured techniques to your assembly-language programming

49 **Pattern-Matching with the 6502 on the APPLE** ..... *Charles F. Taylor, Jr.*
Presents both elementary and advanced pattern-matching algorithms

57 **Random Number Generator in Machine Language for the APPLE** ..... *Arthur Matheny*
A simple subroutine for the APPLE

63 **A New Character Set for the VIC-20** ..... *Mike Bassman*
Design your own VIC characters

I/O ENHANCEMENTS

11 **Data Transfer from AIM to PET** ..... *Alan K. Christensen*
Send data and programs from the AIM to the PET through a simple cable

17 **PET to AIM Download** ..... *George Watson*
Use the PET to develop programs for simpler systems

23 **Expanding File Cabinet for the APPLE** ..... *David P. Allen*
Access File Cabinet for use with other programs

27 **Duty Cycle Monitor for the VIC-20** ..... *Bob Kovacs*
Determine the correct volume level for reliable recording of cassette tapes

33 **Interfacing the Color Computer** ..... *John Steiner*
Send and receive Morse code

BASIC AIDS

71 **POWER-Aid for the PET** ..... *F. Arthur Cochrane*
Add commands to Professional Software's BASIC programming utility, POWER

76 **SURCHANGE for OSI** ..... *Kerry Lourash*
SEARCH/CHANGE for OSI BASIC-in-ROM machines

85 **ON ERROR GOTO for OSI ROM BASIC** ..... *Earl Morris, Kerry Lourash*
A utility to trap errors without stopping program execution

90 **Straightforward Garbage Collection for the APPLE** ..... *Cornelis Bongers*
An APPLE-based linear garbage collection program

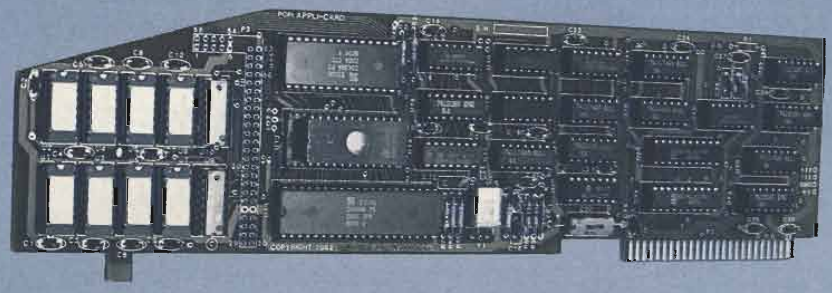
99 **OSI Extended I/O Processor** ..... *Michael J. Keryan*
Add more than thirty new functions to the OSI C1P

106 **Delete on the OSI** ..... *Earl Morris, Yasuo Morishita*
Omit one or more lines with only a few keystrokes

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.

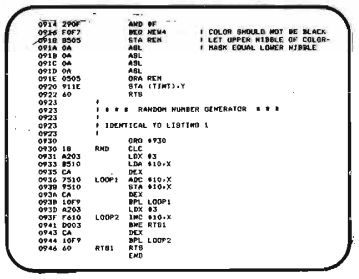
FEATURES	Z-CARD*	SoftCard*	APPLI-CARD
6 mhz Z-80 available	No	No	Yes
64K on-card memory	No	No	Yes
CP/M available	Yes	Yes	Yes
SB-80* with card	No	No	Yes
40 col. to 255 col. horizontal scroll	No	No	Yes
Choice of application	No	No	Yes
2K PROM on the card	No	No	Yes
Real time clock available on the card	No	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.

About the Cover



```
0914 290F           AND #F
0916 F0F7           BEQ NEW4        ; COLOR SHOULD NOT BE BLACK
0918 8505           STA REM         ; LET UPPER NIBBLE OF COLOR-
091A 0A             ASL             ; MASK EQUAL LOWER NIBBLE
091B 0A             ASL
091C 0A             ASL
091D 0A             ASL
091E 0505           ORA REM
0920 911E           STA (ITEMT),Y
0922 60             RTS
0923                ;
0923                ; * * *  RANDOM NUMBER GENERATOR  * * *
0923                ;
0923                ; IDENTICAL TO LISTING 1
0923                ;
0930                ORG $930
0930 18     RND     CLC
0931 A203           LDX #3
0933 B510           LDA $10,X
0935 CA             DEX
0936 7510   LOOP1   ADC $10,X
0938 9510           STA $10,X
093A CA             DEX
093B 10F9           BPL LOOP1
093D A203           LDX #3
093F F610   LOOP2   INC $10,X
0941 D003           BNE RTS1
0943 CA             DEX
0944 10F9           BPL LOOP2
0946 60     RTS1    RTS
            END
```

The penguins on this month's cover happily reside in one of the country's larger, and more modern zoos. Computers help keep these penguins — and their fellow zoo-mates — content by monitoring care and feeding procedures and storing information on the animals' environmental needs.

Cover photo: Jack Smith
Groton, Massachusetts

MICRO is published monthly by:
MICRO INK, 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 INK
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

Copyright© 1982 by MICRO INK
All Rights Reserved

MICRO™

Editorial

Higher than High-Level?

The Observation Level of Boston's John Hancock Tower was an appropriately lofty setting for the unveiling of Software Arts' latest product, TK!Solver (the "TK" stands for "Tool Kit"). The developers of VisiCalc have produced TK!Solver to settle the question, "Can they do it again?" As the lights dimmed, and company president Dan Bricklin sat down at the keyboard of an IBM Personal Computer, everyone knew that VisiCalc was going to be a tough act to follow.

Why have the creators of VisiCalc waited so long to launch a new product? Vice President Tracy Licklider explained that Software Arts has been developing "revolutionary" translation techniques. Using them, Software Arts can adapt a program to a new machine in a matter of days. The process transports an exact replica of a program into a new environment, neither introducing new bugs nor removing old ones. A user can therefore invest time in learning how to use a Software Arts product without worrying about whether those skills will be transportable. If a computer has a significant market share, Software Arts plans to make a compatible program version available.

When Dan Bricklin's large demonstration screen lit up, we saw a format suggestive of VisiCalc, but not organized according to rows and columns. TK!Solver allows the user to specify relationships between defined variables, give the computer data, and have it relate the data to the equations and print an answer. For example, in calculations dealing with real estate mortgages, the user would give the computer an equation that defines the relationship between principal, interest rate, monthly payment, and term of the loan. Given any three of the known values, TK!Solver can quickly calculate the unknown value. The program establishes on its own a sequence of problem-solving steps.

The grey eminence behind TK!Solver, Professor Milos Konopasek of North Carolina State University at Raleigh, has spent more than 10 years developing the artificial intelligence techniques used in the TK!Solver program. These techniques, Konopasek says, enable people to interact with computers at a higher level than they can with "high-level" languages like BASIC and Pascal. While such languages relieve the user of having to give the microprocessor any instructions, they impose a sequential method of problem-solving that is not natural to the human mind. Professor Konopasek calls TK!Solver a "non-procedural programming language," in which the user can conceptualize a problem as a network of defined relationships instead of a rigid sequence of procedures. The result is a more natural, higher-than-high-level, problem-solving computer environment.

Software Arts plans to market TK!Solver in the fourth quarter of 1982, for $299.00. It will first be available on the IBM Personal Computer and the Apple II. In addition to the TK!Solver program itself, Software Arts will produce a series of special application packages for use with the program. These will be available initially in the areas of mechanical engineering, financial analysis, high school science, and architectural design and construction. Each package will include the equations, tables, and values commonly used to solve typical problems in a given area.

MICRO supports the efforts Software Arts is making to bridge the compatibility gap between computers. The TK!Solver program, in combination with the application packages, may make it possible for non-specialists to tackle problems only a degreed professional could previously handle. If so, the computer revolution could be on the verge of realizing some of its enormous potential.

Laurence Kepple

MORE JUICE FOR YOUR APPLE®

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.

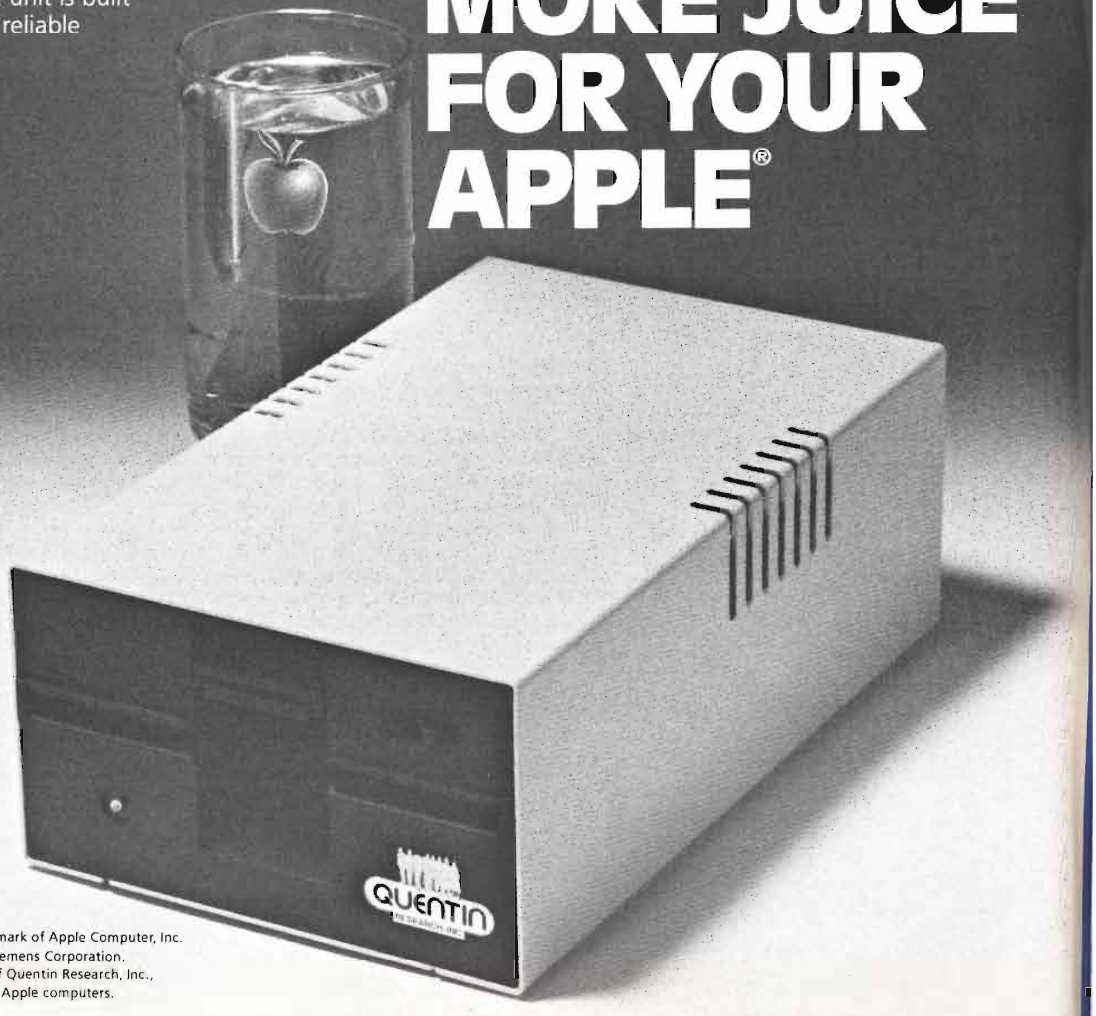
But the best news may be the price—only $335.00 (40 tracks). A special introductory offer when you order Apple-Mate directly from us.

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



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.

MICRO News

Steve Wozniak Creates Cultural Fall Festival

Three of the major cultural forces of the 80's — technology, music, and education — will meet in San Bernardino County over Labor Day weekend for an event called the "Us Festival." Steve Wozniak, cofounder and inventor of Apple Computer, predicts that the "Us Festival will be one of the most significant Labor Day weekend celebrations in the history of America." Wozniak has planned the festival to signal a shift from the "me decade" of the 70's to what he calls an "us decade" for the 80's. Festival promoters have signed top entertainers and expect to draw more than 300,000 people to the three-day event.

According to Wozniak, widespread popular understanding and acceptance of computers will be an important factor in helping us all work together to solve our problems. "The value of people combining their efforts in families, the work force, and society can't be overemphasized," Wozniak said. "My hope is that the Us Festival will underscore the importance of those individual efforts and the role that computers can play in making interaction and cooperation both more effective and more fun."

A Technology Fair will feature exhibits of exciting new applications of computer technology as it relates to communications, education, small business, music, and ecology. For those unable to attend, a specially created Us Network will telecast the event live to theaters and millions of homes throughout America. No tickets may be purchased at the door. Contact The Us Festival at P.O. Box 95108-1157, San Jose, CA 95108, or at #TCW 314 on The Source.

NewsNet: Newsletters for the Business Community

Up to 130 newsletters from 30 publishers are now available to the business community through NewsNet, an electronic information distribution and retrieval service.

Newsletters offered include *Communications Daily, Television Digest, Telecommunications Reports, Data Channels, Cablenews, Energy Daily, Coal Outlook, IRS Practices and Procedures, Education Funding News, Federal Grants and Contracts,* and *Hazardous Waste News.*

According to NewsNet president John H. Buhsmer, many areas of industry will be represented, including farming and food, environment, investment, health, and general business.

Write to NewsNet at 945 Haverford Rd., Bryn Mawr, PA 19010.

Low Cost Electronic Mail System

Small businessmen and computer hobbyists unable to afford expensive electronic mail accounts, can now enjoy this service by joining DCI.DEAFNET. Although primarily a nationwide electronic mail system for the deaf, DCI.DEAFNET now offers their service at considerably lower cost than other electronic mail accounts.

According to DCI.DEAFNET business manager Mary Robinson, anyone wishing to connect to the system pays a small monthly fee and a straight connect charge.

DCI.DEAFNET is offered by the Deaf Communications Institute. For more information contact Ms. Robinson at 95 Bethany Rd., Framingham, MA 01701, (617) 875-3617.

THE SOURCE Increases Services

THE SOURCE has added two features to its services: MAILGRAM messages and a WATS system.

MAILGRAM messages composed by SOURCE subscribers may be sent from their computer to anywhere in the United States. If the messages are entered by 4 p.m., EST, they are virtually guaranteed next business day delivery.

Western Union's largest MAILGRAM processing center receives the SOURCE messages, then routes them by zip code to the appropriate post office. There they are printed and delivered.

SOURCE subscribers can access this service by signing on, then typing MGRAM. Further instructions prompt them through the necessary steps to complete the message.

Messages up to 100 words cost $5.15; each additional 100 words cost an extra $1.00. Discounted prices for identical text are available.

The new SOURCE WATS system enables subscribers to dial direct to SOURCE computers at a reduced cost. For a .25 cent surcharge, along with the standard usage fee, subscribers can dial a specified 800 number and be connected with their host computer.

If you have any questions on these, or any other SOURCE services, write to Source Telecomputing Corp., 1616 Anderson Rd., McLean, VA 22102, (703) 734-7500.

Stock Market Information for Micro Users

Investors with personal computers now have free access to a computerized stock market bulletin board service, called Tickerscreen. Information provided on Tickerscreen includes closing New York Stock Exchange prices, closing market indexes, commission computation on any stock or option transaction, and a demonstration of Tickertec, the personal stock market monitor.

Tickerscreen is available from 5:00 p.m. to 9:00 a.m. weekdays, and 24 hours on weekends. To enter the system, dial (212) 986-1660, and connect your computer to your phone. For more information contact Max Ule and Co., 6 E. 43rd St., 27th Fl., New York, NY 10017, (212) 687-0705.

Electronic Newspaper for Texas TRS-80 Owners

Radio Shack and the Fort Worth Star-Telegram now offer STAR-Text, an electronic newspaper home information service for the local Fort Worth, Texas, area.

STAR-Text will provide subscribers with hard news, entertainment information, and shopping information. Hard news includes local, state, national, and international stories, sports, and weather. Subscribers will have access to movie reviews, concert listings, and live theater and ticket agencies. To help with shopping, STAR-Text will offer consumer advice information, classified advertising, and product information.

For more information, contact the Star-Telegram at (817) 390-7455.

MICRO

Letters/Updates

6502 vs. 6809 Questions

Dear Editor:

I would like to discuss the article by Gregory Walker and Tom Whiteside, "Multiprecision Addition — A Comparison of 6809 and 6502 Programming" [MICRO 47:57]. In their first example [figure 1], the time for 12 zero-page instructions for the 6502 was given as four cycles each instead of three. The corrected subroutine time should be 51 cycles for the 6502 as opposed to 50 for the 6809.

In their second example, "the restriction that these subroutines must leave all processor registers unchanged" [except flags], was not adhered to for the 6809's index registers. If we eliminate the same restriction for the 6502's index registers and correct the branch-not-taken timing, the total time becomes 100 cycles as opposed to 101 for the 6809. The correct byte count for the 6502 is 16 as opposed to 17 for the 6809.

If the 6809 has any clear advantage over the 6502 in terms of execution time or memory usage, it certainly can't be proved by these examples.

George Wells
1620 Victoria Place
La Verne, CA 91750

Mr. Wells is correct that the 6502 LDA and STA instructions in figure 1 use three cycles instead of four cycles. Thanks for pointing out our goof! Including the setup time, the 6502 needs 75 cycles versus *70 for the MC6809.*

His comment that we did not preserve the index registers in the second example is not *true. The only registers changed were the "A" and "B", which were saved with the PSHS D instruction. He is correct that we missed the "1" cycle for the 6502 branch-not-taken. If we include the setup time, the 6502 needs 111 + 36 = 147 cycles* versus *101 + 24 = 125 cycles for the MC6809. The MC6809 program is faster despite the added overhead for using position-independent code (a penalty we did not require of the 6502).*

The MC6809 was more byte efficient and faster than the 6502 examples, but we trust that perceptive readers will not limit the comparison to these factors. Consider also that the MC6809 offers extra registers, many of which are 16-bit indexes. It offers the ease of writing position-independent code, extra addressing modes and extra instructions, a reduced dependence on "zero page" RAM, and a stack that may range anywhere in memory.

Greg Walker and Tom Whiteside

COPCOP Updated for OS65D 3.3

Charles H. Ellis, Jr., of Lynn, MA, sent in this update:

Peter Kleijnan's disk copy program ["COPCOP Single Drive Copier," MICRO 47:21] is a brilliant example of OSI C4P programming. After three years of compulsively programming my C4P, I thought that I had explored every corner of the operating system. What a pleasant surprise to find that a DOS command file can be executed

Listing 1: **Corrections to run on OS65D 3.3.**

```
  5 POKE 133,79
110 DIMD$(39,8):CR$ = CHR$(13):MA = INT( (PEEK(8960)
    - 36)/8)
998 DATA 2,10,18,26,64,72,88,96,104,112,120,128,136,144
1080 DISK!"ME F000,5000":PRINT#5,"EXIT";CR$;:PRINT#9
1164 PRINT#5,"CA  0200 = 06,4";CR$;CR$;"Insert
    MASTER disk";CR$;
1280 DISK!"ME 5000,F000":DISK!"IO 10,02":RETURN
```

Listing 2: **Directory Cleanup Modifications**

```
  90 DB = 11897:DEFFNA(X) = 10*INT(X/16) + X - 16
     *INT(X/16)
1091 PRINT:INPUT"Directory Cleanup";DC$
1093 IF DC$ = "Y"THEN DISK!"CA 2E79 = 12,1"
2010 IF  D$(I,1) = " "ORD$(I,1) = "N"THENGOSUB8000:
     I = I + 1:GOTO2005
5035 IF DC$ = "Y"ANDJ = 12THEND$(12,1) = "2E79/1"
8000 TT = I:IF DC$ < > "Y"THEN8090 : REM — Directory
     Entry Delete
8010 FOR II = DB + 6 TO DB + 254 STEP 8
8020 IF TT < FNA(PEEK(II) ) OR TT  > FNA(PEEK(II + 1) )
     THEN 8060
8040 FOR JJ = II - 1 TO II - 6 STEP - 1:POKEJJ,ASC("#"):
     NEXTJJ
8050 POKE II,0:POKE II + 1,0
8060 NEXT II
8090 RETURN
```

Letters/Updates *(continued)*

directly from memory (another example of the power of OSI's system architecture).

I wish to offer some corrections to this excellent program so that it will run on the new OS65D 3.3, and present a directory clean-up subroutine to enhance the disk copier.

Corrections

Under the new V3.3 of OS65D, "COPCOP" will not run without the revisions shown in listing 1. Since V3.3 BASIC and the keyboard enhancement occupy 2K of memory not used in OS65D 3.2, the published program produces an 'OM' error when it tries to dimension the 39 × 8 array. Also, the track 0 copy utility has been relocated on the V3.3 system disk from track 13 to track 6.

These problems are simple to solve. Moving up the top of the BASIC workspace by 2K leaves ample room for the array. Changing the address in lines 1080 and 1280 from $4800 to $5000 moves the memory output and input, and replacing 80 with 72 in the DATA statement (line 998) protects the command file block during copying. Changing the call in line 1164 from 13,1 to 06,4 correctly locates the track 0 utility.

The added 2K of DOS permits copying of one less track in each pass, and requires the change in line 110 (from PEEK(8960) – 29 to PEEK(8960) – 36). This still allows copying of 19 tracks per pass on a 48K machine, and 7 tracks on my 24K system.

Update —
Directory Cleanup Subroutine

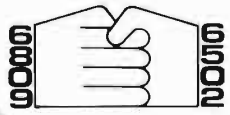
I have always been frustrated that most disk copiers simply copy the original disk directory to the copy disk, even when some tracks have not been copied to the new disk. COPCOP inspired me finally to deal with this problem. If the new lines of listing 2 are added to the program, the new disk can, optionally, be provided with a directory which includes only those files which were fully copied.

When a track is not to be copied, or has a missing header, the subroutine searches the directory buffer ($2E79-$2F78) for an entry including that track number. If the search succeeds, the buffer is edited to delete the entry. If this option has been selected, the edited directory buffer will be saved to disk during copying, rather than a copy of the original track 12,1.

A warning: this routine works only for a directory complete in the first page of track 12. I have never yet had a disk with more than one page needed for its directory. Only on a data disk might you have more than 32 one-track, named files. If this large a directory exists, the disk should be copied without the cleanup option. Then the directory can be cleaned up in the old way, by hand, file-by-file, using a delete program. I don't expect to have to do this.

MICRO™

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.

SDS GIVES YOU ALL YOU NEED!

APPLE-DOC

We are proud to announce an updated version of our best seller, **Apple-Doc** in DOS 3.3. This versatile package consists of three powerful reference utilities; **Variable Cross-Reference** which creates a table of every variable and the line number on which it occurs, **Line Number Cross-Reference** creates a table of every program lined called by a GOTO, GOSUB, etc., and all lines called from, and finally, **Constant Cross-Reference** creates a table of all numbers used in the program not listed by the above tables. Also included are **Replace**, a global replacement editor and **Lister** which produces a completely professional program listing. Additionally, the new 30 page tutorial manual makes **Apple-Doc** quick and easy to use. Whether novice or professional **Apple-Doc** more than pays for itself in time savings alone!

Price: $49.95 on disk

a.c.e.

Applesoft* Command Editor, or **A.C.E.**, combines a large number of the most useful programming utilities into one co-resident package. Co-resident means these routines become a working part of the Apple II*, thereby making **A.C.E.'s** various commands and utilities immediately available WHILE you're entering or editing existing program lines. It's functions include cross-referencing individual variable names, displaying current values of all variables, and information on memory and diskette status. For machine language programming there is a hex-decimal converter and all monitor commands can be executed directly from Applesoft. User defined macros allow strings to be input with a single keystroke and a powerful line editor makes editing easy and greatly reduces program development time. **A.C.E.** is a winner!

Price: $39.95 on disk

List Master

List Master, our latest addition, provides a number of utilities for performing large scale changes to your program listings in a high-speed and automatic way. **List Master** includes **Applespeed** which can remove REMarks, shorten variable names, combine lines, and renumbers by 1's in any Applesoft program. **Smart Renumber** not only renumbers but also gives you the option of preserving logical blocks of line numbers you have established for various routines thereby retaining the original logic and workability in the renumbered program. Also included are merge routines that allow you to quote blocks of lines from one part of the program to another and automatically renumber appropriate line references. **Comp-List** compares any two Applesoft or Integer programs and lists any lines added, deleted or changed.

Price: $39.95 on disk

All programs require 48K and Applesoft in ROM or language card.
Calif. residents add 6% sales tax.

*Apple II and Applesoft are registered trademarks of Apple Computer Co.

SDS

southwestern data systems

P.O. Box 582 • Santee, California 92071 • (714) 562-3670

Data Transfer from AIM to PET

by Alan K. Christensen

Software is provided to transfer data, such as machine-language programs, from the AIM to the PET.

AIM to PET
requires:
PET
AIM 65 or SYM
AIM Assembler ROM
Cable (see page 12)

The owner of any popular computer soon realizes that it is easy to get locked into using only the programs and peripherals offered by that manufacturer. This article grows from a belief that almost any computer can be expanded by adding a different computer in a network. This allows the user to choose between peripherals, tape formats, and programs offered for either machine. This article explains how to make a logical connection between a Commodore PET and a Rockwell AIM 65.

I purchased my AIM after a lengthy search for a fast PET assembler that would run in my 8K PET. Throughout this search, I kept admiring the small ROM-based editor assemblers found on the SYM and AIM. I decided that if I waited long enough a similar product would be available for the PET. After waiting, and adding up the cost of commercial software and the required memory expansion and additional peripherals, the cost of an AIM no longer seemed too high, so I bought one.

(Editor's note: ASM-TED is a cassette-based assembler written in machine language for the 16K PET and other 6502 computers. It is available for $49.95 from Eastern House Software, 3239 Linda Drive, Winston-Salem, NC. Two small, ROM-based assemblers have recently become available: Mikro from Skyles Electric Works, 231E South Whisman Road, Mountain View, CA 94041, and EZ-ASM from Data Cap, 73 Rue du Village, 4545 Feneur, Belgium.)

The AIM editor and assembler are not the fanciest programs, but they do have features that I required most. The assembler accepts standard 6502 mnemonics with symbolic names. It is not, however, a macro assembler. The editor is line-oriented with search and replace capabilities as well as insert and delete. With the code in ROM, they leave most of the RAM available for text or symbol tables. The combination makes it reasonable to develop serious programs in assembly language. My only problem was to get the programs from the AIM to the PET.

There appeared to be three ways to transfer information between the two computers: manually, through a storage medium, or with a direct connection. The direct connection approach turned out to be the easiest. For details on how to construct a cable, see page

Once the two machines are connected, there are many ways to use them in tandem. For example, I use the PET to buffer large amounts of output and send it to a Commodore printer at its leisure. I also use the AIM keyboard as an input device to the PET (my PET has the old, tiny keyboard). All of these applications work best in assembly language, so the first goal is to transfer assembly-language programs from the AIM to the PET. The AIM monitor supports user-defined devices. Listing 1 shows a simple software interface to get the assembly object out of the machine by sending it to the user (U) device. There is more information on the data transfer in the AIM manuals.

The interface as described supports only half-duplex transmissions. This means that the PET has to be ready to accept the information when the AIM sends it. The program for the PET is written in BASIC because it is easier to enter and debug a BASIC program on the machine. This program (listing 2) will pick apart the AIM format object code and POKE it into the PET memory. With this program, the parity of the data is not checked, and the PET relies entirely on the checksums in the object format for data integrity.

The program has been optimized for speed. I will briefly describe it from back to front. Lines 800-830 are a trap location and are only entered when an error is detected. The program variables and the 6522 registers are initialized in lines 600-735. This module is written as a subroutine because I occasionally add short, special-purpose routines on the back of the program and they need initialization too. Lines 500-550 process the last line of object format; all other lines are processed in lines 400-480. The bytes are read into an array; if no checksum errors occur, they are POKEd into memory. Subroutine 300-310 returns a 16-bit value based on four hex characters. Subroutine 200-220 returns an 8-bit value based on two hex characters. Subroutine 100-120 gets one character from the interface. The PET user port does not provide a complete A port, so this subroutine has to handshake the values using the CB2 line. The program uses no string variables and will not be affected if another program is loaded into high memory.

By typing GOSUB 600 : GOTO 800 the cable and interface subroutine can be tested. To test the entire routine, dump the contents of $8000-$83FF from the AIM with U as the output device (after assembling the AIM user interface). The PET screen should fill

entirely with various characters, and the PET should not crash. If this operation is successful, the interface has a good chance of being correct.

When the BASIC program is working, the program from listing 3 can be entered into the AIM, assembled, and transferred to the PET. This program is virtually the same as the BASIC program, but written in assembly language. There are two addresses which are valid only for the early versions of the PET. The value of ADDR must be a zero-page location that will not interfere with the operation of the PET. I used $58 at the end of the input buffer in old ROMs. (*Editor's note:* These addresses should work most of the time for upgrade and 4.0 ROMs.) This program places a jump to INIT in memory location 1 and 2 so that the program can be started by typing SYS 0. These locations could be used as the value of ADDR for any PET.

The label CKSTOP is set to the address of a routine which will check to see if the stop key has been pressed and thereby exit the program. This should be changed to the appropriate routine for later ROMs (as indicated in the comments). Any PET can use the get routine address of 65508 for CKSTOP by changing the BEQ EXITI after the label FLAGCK to a BNE EXITI. This will cause the program to halt if any key is pressed on the PET keyboard.

There are additional comments in the listings. The assembler listing has as comments the line numbers that correspond to the BASIC program. These can be used as a guide to the conversion of the program from BASIC. Note that the two programs use different methods to convert hex characters into binary values. To fit the entire assembly-language source into a 4K AIM, it should be entered without comments. The AIM can be expanded to have more memory and additional 6522 ports, so that one AIM could be connected to several PETs.

This program, with some modification, could be used to send object code from an AIM into nearly any 6502-based computer that has I/O lines accessible to the programmer. Once that is accomplished, the way is open for several network programs on the machines. One of the simplest multiprocessing configurations is to have one machine process data while the other controls input or output, including formatting or real-time control. Another possibility: one machine can serve as a memory extension of the other, perhaps maintaining sorted lists or multi-dimensional arrays, and provide data at the command of the master device. In many of these cases, a network approach will provide great improvements in speed and flexibility. The largest pitfall occurs when one machine is running while the other waits, only to have the other machine begin processing while the first waits. As a final caution, if the communications protocol becomes too complex, both machines may spend more time handshaking than working.

Making a Cable

	PET	AIM (SYM)		
Pin	Signal	Signal	AIM Application Connector Pin	SYM AA Connector Pin
A	ground	ground	1	1
B	CA1	CA2	21	4
C	PA0	PA0	14	D
D	PA1	PA1	4	3
E	PA2	PA2	3	C
F	PA3	PA3	2	12
H	PA4	PA4	5	N
J	PA5	PA5	6	11
K	PA6	PA6	7	M
L	PA7	PA7	8	10
M	CB2	CA1	20	E

Because the PET's CB2 line is used here, other uses, such as sound, will not be possible when transferring data.

Connectors

The connector for the AIM or SYM is a standard 44-pin edge card connector available from Radio Shack and other electronics parts stores. The PET's parallel user port connector may be more difficult to obtain. The following manufacturer's part numbers can be used:

Cinch	251-12-90-160
Sylvania	6AG01-12-1A1-01
Amp	530657-3
Amp	530658-3
Amp	530654-3

SYM VIA Addresses

For the SYM, the following addresses apply for its VIA #2:

Output register A	$A801
Data direction register A	$A803
Peripheral control register	$A80C
Interrupt flag register	$A80D

Alan Christensen received a BSEE from the University of Texas and has programmed both large and small computers for five years. His systems include a PET with a graphics screen display from Micro Technologys Inc., and an AIM with MTU graphics and floppy disks. Mr. Christensen may be contacted at 1303 Suffolk, Austin, Texas 78723.

Listing 1

```
;AIM/PET OUTPUT   OCT 12, 1981     AKC
; THIS PROGRAM CONNECTS THE PET AND AIM
; WITH THE PET BEING DEFINED AS THE
; USER DEVICE OF THE AIM.
;
ENTRY = $2000 ;ENTRY AND LOWEST ADDRESS
;
;
* = $010A ;OUTPUT VECTOR FOR AIM   (SYM - $A663)
UOUT .WORD OUTPUT
;
* = ENTRY
;
OUTPUT = * ;OUTPUT FUNCTION
BCS OUTCHR
LDA #$FF ;OUTPUT STATUS FOR A PORT
STA DDRA
;
;
INITA = * ; 6522 INITIALIZATION
; THIS INITIALIZATION ASSUMES THAT THE A PORT
; HAS NOT BEEN PREVIOUSLY SET TO GENERATE
; INTERRUPTS.
ORAR    = $A001 ;PORT A DATA REGISTER (WITH HANDSHAKE)
DDRA    = $A003 ;PORT A DATA DIRECTION REG
PCR     = $A00C ;PERIPHERAL CONTROL REG
IFR     = $A00D ;INTERRUPT FLAG REG
;
LDA PCR ;SAVE PORT B SETTINGS ON PCR
AND #$F0
ORA #9 ;SET HANDSHAKE FOR PORT A
STA PCR
;
RTS
;
OUTCHR = * ;OUTPUT THE CHARACTER ON STACK
LOOP LDA IFR ;WAIT UNTIL PET IS READY
AND #2 ;THIS ROUTINE WILL HANG THE AIM UNLESS
BEQ  LOOP ;THE PET RESPONDS
;
PLA ;AFTER PET RESPONSE SEND THE CHARACTER
STA ORAR
RTS
;
.END
```

Listing 2

```
10 GOSUB 600 :GOTO 400
100 POKE PCR,P0%:POKE PCR,P1%
105 IF N0 =(PEEK(ITFR) AND CA1) THEN 105
110 A=PEEK(REGA)
115 POKE ITFR,CA1
120 RETURN
200 GOSUB 100:C1=C(A-C0)
205 GOSUB 100:C2=C(A-C0)
210 A=C1*16 + C2
215 KK=KK+A
220 RETURN
300 GOSUB 200:AA=A
305 GOSUB 200:AA=256*AA + A
310 RETURN
400 GOSUB 100:IF A<>SEMI THEN 400
405 KK=N0:NR=NR+N1
410 GOSUB 200:CNT=A
415 IF CNT=N0 THEN 500
420 GOSUB 300 :AD=AA
425 FOR I=N1 TO CNT
430 GOSUB 200 :A(I)=A
435 NEXT
440 CK=KK
445 GOSUB 300
450 IF AA<>CK THEN PRINT"CHECK SUM ERROR":STOP:GOTO 800
455 FOR I=1 TO CNT
460 POKE AD,A(I)
465 AD=AD+N1
470 NEXT
480 GOTO 400
500 GOSUB 300
505 IF AA<>NR THEN PRINT"RECORD COUNT MISMATCH"
    :STOP:GOTO 800
510 CK=KK
520 GOSUB 300
525 IF AA<>CK THEN PRINT"CHECKSUM ERROR":STOP:GOTO 800
530 GOSUB 100
535 GOSUB 100
545 PRINT"DONE":STOP
550 GOTO 800
```

Listing 2 *(Continued)*

```
600 A=0:AA=0:I=0
605 AD=0:REM MEMORY ADDRESS
610 CNT=0:REM BYTE COUNT
615 N0=0:N1=1:REM CONSTANTS
620 CK=0:REM CHECK SUM
625 KK=0:REM BYTE VALUE COUNT
630 C1=0:REM HIGH NIBBLE
635 C2=0:REM LOW  NIBBLE
640 NR=0:REM RECORD COUNT
645 SEMI=ASC(";")
650 C0=ASC("0")
655 PCR=59468:REM PERIFERAL CONTROL REGISTER
660 REGA=59471:REM A DATA (WITHOUT HANDSHAKE)
665 ITFR=59469:REM INTERRUPT FLAG REGISTER
670 CA1=2:REM CA1 INTERRUPT FLAG
675 I=PEEK(PCR) AND (16+8+4+2):REM SYSTEM BITS
680 P0%=I OR ((8+4)*16):REM CB2 LOW
685 P1%=I OR ((8+4+2)*16):REM CB2 HIGH
690 DIM C(15+7):REM CHARACTERS "0"-"F"
695 DIM A(24):REM UP TO 24 BYTES/RECORD
700 DATA 0,1,2,3,4,5,6,7,8,9
705 DATA -1,-1,-1,-1,-1,-1,-1
710 DATA 10,11,12,13,14,15
715 FOR I=0 TO 15+7
720 READ C(I)
725 NEXT
730 POKE 59459,0:REM DATA DIRECTION REGISTER
735 RETURN
800 GOSUB 100
805 PRINT CHR$(A);
810 GOTO 800
820 REM PET AIM CONNECTION
825 REM ALAN K. CHRISTENSEN
830 REM JULY 20, 1981
```

Listing 3

```
;PET$AIM$ASM.C
;AIM TO PET LOADER INTERFACE
;BY ALAN K. CHRISTENSEN
;AUSTIN, TEXAS
;JULY 20, 1981
;
;PET DEPENDANT ADDRESSES
;PET 1.0 ADDRESSES GIVEN
ADDR   = $58 ;USE END OF BASIC INPUT BUFFER
             ;TO STORE ADDRESS. THE BYTES
             ;WILL BE RESTORED AT EXIT
CKSTOP = 62250 ;ROUTINE TO CHECK FOR STOP KEY
          (Upgrade - 62209, 4.0 - 62261)
;
;
        JMP INIT ;USR(0) ENTRY
;
        *=$2200
MSG     = *
FCHMSG .BYTE 13,'FINAL '
CHKMSG .BYTE 'CHECK SUM '
        .BYTE 'ERROR',13
CHKLEN = *-CHKMSG
FCHLEN = *-FCHMSG
;
RECMSG .BYTE 13,'RECORD NUMBER '
        .BYTE 'MISMATCH ERROR',13
RECLEN  = *-RECMSG
;
DONE    .BYTE 13,'DONE',13
DONEL   = *-DONE
;
;LINE 100
GETAIM = * ;GET 1 BYTE FROM THE AIM
        LDA P0      ;PULSE CB2
        STA PCR     ;FIRST LOW
        LDA P1      ;THEN HIGH
        STA PCR     ;TO MAKE HANDSHAKE
;
FLAGCK JSR CKSTOP  ;CHECK FOR STOP KEY
        BEQ EXITI    ;TO  LEAVE LOOP
;
;LINE 105
        LDA ITFR    ;CHECK THE INTERRUPT FLAG
        AND #CA1    ;REGISTER FOR READY
        BEQ FLAGCK ;SIGNAL FROM AIM
```

(Continued)

Listing 3 *(Continued)*

```
;
;LINE 110
       LDA REGA    ;GET 1 BYTE FROM INTERFACE
       PHA


;
;LINE 115
       LDA #CA1    ;CLEAR THE INTERRUPT FLAG
       STA ITFR
;
;LINE 120
;
      PLA          ;RETURN
      RTS
;
;LINE 300-310
;
G2BYTS JSR GETBYT ;GET 2 ASCII BYTES
       TAX         ;THE 1ST IN X
;                  ;AND THE SECOND IN A
;LINE 200
;
GETBYT JSR GETAIM ;GET 1 ASCII BYTE FROM AIM
       SEC         ;AND CONVERT THE ASCII
       SBC #'0'      ;HEX INTO BINARY
       CMP #10     ;
       BCC HEX1    ;RETURN BYTE IN A
       SBC #7
HEX1   ASL A
       ASL A       ;MULTIPLY 1ST DIGIT
       ASL A       ;BY 16
       ASL A
       STA DIGIT1
;
;LINE 205
;
       JSR GETAIM ;GO FOR THE
       SEC        ;SECOND DIGIT
       SBC #'0'
       CMP #10
       BCC HEX2
       SBC #7
HEX2   =   *
;
;LINE 210
;
       ORA DIGIT1 ;COMBINE THE 1ST AND 2ND DIGIT
;
;LINE 215
;
       PHA        ;ADD THIS BYTE TO THE
       CLC        ;CHARACTER COUNT
       ADC KKSUM+1
       BCC ADD1
       INC KKSUM
ADD1   STA KKSUM+1
       PLA

;
;LINE 220
;
       RTS        ;RETURN
;
EXITI  LDX SAVSTK ;RESET STACK POINTER
       TXS        ;TO ALLOW EXIT FROM
       JMP EXITT  ;WITHIN SUBROUTINE
;
;LINE 400
;
LOADER = *        ;LOAD A RECORD IN AIM FORMAT
       JSR GETAIM ;LOOK FOR THE SEMI-
       CMP #';'   ;COLON THAT STARTS
       BNE LOADER ;THE RECORD
;
;LINE 405
;
       LDY #0     ;RESET THE CHARACTER COUNTER
       STY KKSUM  ;TO COUNT ALL CHARACTERS
       STY KKSUM+1 ;AFTER THE ';'
       INC NREC+1
       BNE ADD2    ;ADD 1 TO NUMBER OF RECORDS
       INC NREC
ADD2   =   *
;
```

Listing 3 *(Continued)*

```
;LINE 410
;
       JSR GETBYT ;FIRST BYTE OF RECORD IS
       AND #$1F   ;THE DATA BYTE COUNT
       STA COUNT  ;AND SHOULD BE <= 24
;
;LINE 415
;
       BEQ LAST   ;CHECK FOR LAST RECORD
;
;LINE 420
;
       JSR G2BYTS ;GET THE BASE ADDRESS
       STX ADDR+1 ;FOR THE DATA BYTES
       STA ADDR   ;AND SAVE IN ZERO PAGE

;
;LINE 425
;
FLOOP1 =   *      ;Y=0 FROM LINE 405
;
;LINE 430
;
       JSR GETBYT ;GET THE DATA BYTES
       STA ARRAY,Y ;AND SAVE THEM IN ARRAY


;
;LINE 435
;
       INY        ;LOOP BACK FOR ALL BYTES
       CPY COUNT  ;
       BNE FLOOP1 ;
;
;LINE 440
;
       LDA KKSUM  ;THE ACCUMULATED SUM
       STA CHKSUM  ;OF THE BYTES SO FAR
       LDA KKSUM+1 ;ARE EQUIVALENT TO
       STA CHKSUM+1 ;THE CHECK SUM
;
;LINE 445
;
       JSR G2BYTS ;GET THE CHECKSUM
;                 ;FROM THE RECORD
;LINE 450
;
       CMP CHKSUM+1 ;CHECK CALCULATED
       BNE CHKERR   ;CHECKSUM WITH
       CPX CHKSUM   ;RECORD CHECK
       BNE CHKERR   ;SUM
;
;LINE 455
;
       LDY #0     ;SET UP DATA-
FLOOP2 =   *      ;BYTE STORE LOOP
;
;LINE 460
;
       LDA ARRAY,Y ;STORE ALL DATA BYTES
       STA (ADDR),Y
;
;LINE 465
;
       INY        ;INCREASE ADDRESS OF STORE
;
;LINE 470
;
       CPY COUNT  ;CHECK FOR FINISHED
       BNE FLOOP2
;
;LINE 480
;
       JMP LOADER ;GO BACK FOR NEXT RECORD
;
;LINE 500
;
LAST   JSR G2BYTS ;GET NUMBER OF RECORDS SENT
;
;LINE 505
;
       CMP NREC+1 ;COMPARE TO NUMBER OF
       BNE RECERR ;RECORDS RECEIVED
       CPX NREC   ;A MISMATCH INDICATES
       BNE RECERR ;THAT DATA WAS LOST
```

Listing 3 *(Continued)*

```
;
;LINE 510
;
       LDA KKSUM+1  ;MOVE THE CHARACTER
       STA CHKSUM+1 ;COUNT INTO CHECKSUM
       LDX KKSUM    ;
       STX CHKSUM   ;
;
;LINE 520
;
       JSR G2BYTS ;GET FINAL CHECKSUM
;
;LINE 525
;
       CMP CHKSUM+1 ;CHECKSUM SHOULD BE
       BNE FCHERR ;A GUARD FOR THE
       CPX CHKSUM   ;RECORD COUNT
       BNE FCHERR
;
;LINE 530
;
       JSR GETAIM ;CLEAR REMAINING CARRIAGE
;
;LINE 535
;
       JSR GETAIM ;RETURNS TO FREE AIM
;
;LINE 540
;
EXIT   LDX #DONE-MSG ;PRINT 'DONE'
       LDY #DONEL
       JSR PRTMSG
;
EXITT  LDA SAVEZP ;RESTORE ZERO PAGE
       STA ADDR   ;USED BY ADDRESS
       LDA SAVEZP+1
       STA ADDR+1
       RTS        ;RETURN TO CALLING PROGRAM
;
CHKERR LDX #CHKMSG-MSG ;CHECKSUM ERROR
       LDY #CHKLEN     ;PRINT MESSAGE
MSGXIT JSR PRTMSG      ;AND RETURN
       JMP EXITT
;
FCHERR LDX #FCHMSG-MSG ;FINAL CHECK-
       LDY #FCHLEN     ;SUM ERROR
       BNE MSGXIT
;
RECERR LDX #RECMSG-MSG ;RECORD COUNT
       LDY #RECLEN     ;ERROR
       BNE MSGXIT
;
;LINES 600-650
;
ARRAY  *=*+24   ;VARIABLE STORAGE
DIGIT1 *=*+1
CHKSUM *=*+2
KKSUM  *=*+2
NREC   *=*+2
;
COUNT  *=*+1
P0     *=*+1
P1     *=*+1
SAVEZP *=*+2
SAVSTK *=*+1
;
;LINES 655-670
;
PCR    = 59468 ;PERIFERAL CONTROL REGISTER
REGA   = 59471 ;A PORT DATA REGISTER
ITFR   = 59469 ;INTERRUPT FLAG REGISTER
DDRA   = 59459 ;A PORT DATA DIRECTION REGISTER
CA1    = 2     ;MASK FOR CA1 INTERRUPT FLAG
;
INIT   LDA ADDR   ;INITIALIZATION
       STA SAVEZP
       LDA ADDR+1 ;SAVE ZERO PAGE LOCATION
       STA SAVEZP+1 ;USED BY ADDR
;
       TSX        ;SAVE THE STACK POINTER
       STX SAVSTK ;SO STOP KEY CAN EXIT
;
;LINE 675
;
       LDA PCR    ;SAVE THE BITS FROM PCR
       AND #$1E   ;THAT ARE NOT USED IN INTERFACE
```

Listing 3 *(Continued)*

```
;
;LINE 680
;
       ORA #$C0    ;CB2 LOW
       STA P0
;
;LINE 685
;
       ORA #$E0     ;CB2 HIGH
       STA P1
;
;LINE 730
;
       LDA #0      ;ZEROS MAKE DATA DIRECTION
       STA DDRA    ;INTO INPUTS
;
       STA NREC    ;INITIALIZE RECORD
       STA NREC+1 ;COUNTER
;
;LINE 735
;
       JMP LOADER
;
;
; SUBROUTINE TO PRINT A MESSAGE TO THE PET SCREEN
;X POINTS TO MESSAGE WITHIN PAGE LENGTH OF MSG
;Y IS THE LENGTH OF THE MESSAGE
PRINT=$FFD2 ;PET 2001 PRINT CHARACTER
;
PRTMSG = *
;
;
LDA MSG,X
JSR PRINT
INX
DEY
BNE PRTMSG
RTS
;
.END
```

MICRO

RPL

RPL is a fast, space-efficient language, designed for the PET/CBM user who wants to develop high-speed, high-quality software with a minimum of effort. While ideal for programming games and other personal applications, it is primarily oriented toward real-time process control, utility programming, and similar demanding business and industrial uses.

R. Vanderbilt Foster, of Video Research Corporation, says he thinks that **"RPL** is one HELL of a system!" (capitals his). Ralph Bressler, reviewing the package in The Paper, says "I know of few language systems this complete, this well documented, for this kind of price." For more information, see the following:

MICRO, Dec. '81, p. 35
MICROCOMPUTING, Feb. '82, p. 10
MICRO, Mar. '82, p. 29
BYTE, Mar. '82, p. 476
COMPUTE!, Mar. '82, pp. 45, 120.

See also the article **"Basic, Forth and RPL"** in the June '82 issue of MICRO, and Mr. Bressler's review in the Jan./Feb. '82 issue of The Paper. Don't let our prices deceive you: **RPL** is a first-class, high performance language in every respect. We are keeping its price so low in order to make it accessible to the widest possible number of users. Only **$80.91**, postpaid, for both the **RPL** compiler and its associated symbolic debugger, complete with full documentation (overseas purchasers please add $5.00 for air mail shipping). Versions available for PET-2001 (Original, Upgrade or V4.0 ROM's), CBM 4032, and CBM 8032/8096, on cassette, 2040/4040, and 8050 disk.

Order Anytime, Day or Night 7 Days A Week
800-327-8965
(ask for extension 2)
VISA
Master Charge
American Express

Samurai Software
P.O. Box 2902
Pompano Beach, Florida 33062
(305) 782-9985

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

PET to AIM Download

by George Watson

A fast, reliable method is presented for transfer of programs from the PET microcomputer to the AIM 65 (or other micro with an accessible 6522). Macroassemblers available for the PET may thus be easily used for developing AIM, SYM, and KIM programs.

PET to AIM
requires:

PET
AIM 65 or SYM
Interface cable (see page 12)

During the past several years, a wide range of computers has been developed based on the 6502 microprocessor. With such a wide spectrum it is possible to choose a system tailored specifically for a given job. A microcomputer with high-level languages, disk storage, and a plethora of peripheral equipment can be chosen for an application requiring extensive data analysis or file management. For simply controlling a piece of equipment though, a more reasonable and economical choice may be a single board computer. When both types of systems are present the thought naturally occurs that any type of program development for the bare bones system should be implemented using the full-blown system. Assembler source files and object code may be easily created with powerful macro-assemblers or compilers. Then the object code needs only to be transferred to the less-expanded computer. Specifically, I would like to consider the downloading of machine code from the PET/CBM to the AIM 65.

The hardware requirements for transfer of data between two computers are: 1) an output port, 2) an input port, and 3) wires connecting the two ports. Many 6502 computers make available to the user a parallel port consisting of a 6522, the Versatile Interface Adapter. The AIM 65 has two 8-bit ports with

Listing 1: **AIM Initialization Routine. Runs in AIM.**

```
                    0010 ;************************
                    0020 ;* AIM-FROM-PET DOWNLOAD *
                    0030 ;*      GEORGE WATSON     *
                    0040 ;*      FEB. 8, 1982      *
                    0050 ;************************
                    0060 ;
                    0070 ;***********************
                    0080 ;* ASSEMBLY DIRECTIVES *
                    0090 ;***********************
                    0100 ;
                    0110            .BA $0200
                    0120            .CE
                    0130            .LS
                    0140 ;
                    0150 ;*************************
                    0160 ;* AIM J1 CONNECTION PORT *
                    0170 ;*************************
                    0180 ;
                    0190 VIA        .DE $A000           ;BASE ADDRESS OF AIM 6522
                                                        ;SYM VIA #2: $A800
                    0200 ORAH       .DE VIA+$01         ;PORT A WITH HANDSHAKING
                    0210 DDRA       .DE VIA+$03         ;DATA DIRECTION REGISTER A
                    0220 PCR        .DE VIA+$0C         ;PERIPHERAL CONTROL REG.
                    0230 IFR        .DE VIA+$0D         ; INTERRUPT FLAG REGISTER
                    0240 ;
                    0250 ;*********************
                    0260 ;* PROGRAM VARIABLES *
                    0270 ;*********************
                    0280 ;
                    0290 START      .DE $00             ;START ADDRESS OF PROGRAM
                    0300 LENGTH     .DE $02             ;LENGTH OF PROGRAM
                    0310 ;
                    0320 ;********************
                    0330 ;* INITIALIZE PORT *
                    0340 ;********************
                    0350 ;
0200- A9 00         0360 PORTINIT   LDA #0               ;SET PORT A AS INPUT
0202- 8D 03 A0      0370            STA DDRA
0205- A9 09         0380            LDA #9              ;SET CA2 OUTPUT HANDSHAKE
0207- 8D 0C A0      0390            STA PCR             ; MODE WITH CA1 INT FLAG
                    0400 ;                                SET ON RISING EDGE
020A- AD 01 A0      0410            LDA ORAH            ;CLEAR PORT
                    0420 ;
                    0430 ;*********************************
                    0440 ;* GET START ADDRESS AND LENGTH *
                    0450 ;*********************************
                    0460 ;
020D- 20 39 02      0470 ADDRESS    JSR PORTGET         ;LOAD LOW THEN HIGH BYTE
0210- 85 00         0480            STA *START
0212- 20 39 02      0490            JSR PORTGET
0215- 85 01         0500            STA *START+1
0217- 20 39 02      0510            JSR PORTGET
021A- 85 02         0520            STA *LENGTH
021C- 20 39 02      0530            JSR PORTGET
021F- 85 03         0540            STA *LENGTH+1
                    0550 ;
                    0560 ;********************
                    0570 ;* DOWNLOAD PROGRAM *
                    0580 ;********************
                    0590 ;
0221- A2 00         0600 DOWNLOAD   LDX #0              ;TRANSFER PROGRAM
0223- A0 00         0610            LDY #0
0225- 20 39 02      0620 NEXTBYTE   JSR PORTGET         ;LOAD BYTE
0228- 91 00         0630            STA (START),Y
022A- C8            0640            INY
022B- D0 03         0650            BNE SKIP1
022D- E8            0660            INX                 ;GOTO NEXT PAGE
022E- E6 01         0670            INC *START+1
0230- E4 03         0680 SKIP1      CPX *LENGTH+1       ;END OF PROGRAM?
0232- D0 F1         0690            BNE NEXTBYTE
0234- C4 02         0700            CPY *LENGTH
0236- D0 ED         0710            BNE NEXTBYTE
0238- 00            0720 EXIT       BRK
```

complete handshaking signals available. PET/CBM has one 8-bit port with partial support for handshaking. Pinouts for construction of the cable are given on page 12.

The program required in the AIM (listing 1) is very short and consists of four sections: initialization, start and length parameters, downloading, and the port input subroutine. With the VIA address substitutions given, the program should run on the SYM, using its VIA #2. As assembled, the program resides in the second page of memory and may be entered *via* the AIM monitor. This program is run by setting the program counter < * >0200, and entering < G >. The AIM should always be initialized before the PET begins the transfer.

The PET program requests that input of the AIM start address, and the PET start and end address of the program, be transferred. (See listing 2.) All addresses should be given as 4-digit hex numbers. These addresses are input using a routine present in the PET monitor. If an error is made in entry, the user will be returned to the monitor with a "." prompt; exit with X and start again. The program has been preceded with a single line of BASIC (10 SYS1037) so that RUN causes execution of the download. Any editing or variable use at this point will destroy the program.

I usually download code to the AIM immediately after assembling it with Carl Moser's MAE macroassembler. After assembly, the end address (+1) of the stored object code is stored at $7651,$7652. Unfortunately it seems that the start address and offset address are not stored (although they are present in the source file). By replacing the section entitled * GET PET END ADDRESS * as follows we need only enter the PET and AIM start address and not be concerned with the program length.

```
MAEEND >   DE $7651
HEXOUT     DE $D722
;
PETEND     LDY #H,STR3
           LDA #L,STR3
           JSR STROUT
           LDA MAEEND + 1
           STA END + 1
           JSR HEXOUT
           LDA MAEEND
           STA END
           JSR HEXOUT
```

Transferring programs between microcomputers equipped with 6522

Listing 1 *(Continued)*

```
                    0730 ;
                    0740 ;*****************
                    0750 ;* VIA PORT GET *
                    0760 ;*****************
                    0770 ;
0239- AD 0D A0      0780 PORTGET    LDA IFR             ;WAIT FOR DATA READY
023C- 29 02         0790            AND #2              ; SIGNAL
023E- F0 F9         0800            BEQ PORTGET
0240- AD 01 A0      0810            LDA ORAH            ;RECEIVE BYTE
0243- 60            0820            RTS
                    0830 ;
                    0840            .EN
```

***Listing 2:* PET to AIM Download. Runs in PET.**

```
                    0010 ;************************
                    0020 ;* PET-TO-AIM DOWNLOAD *
                    0030 ;*    GEORGE WATSON    *
                    0040 ;*    FEB. 4, 1982     *
                    0050 ;************************
                    0060 ;
                    0070 ;************************
                    0080 ;* ASSEMBLY DIRECTIVES *
                    0090 ;************************
                    0100 ;
                    0110            .BA $0400           ;NORMAL START OF BASIC
                    0120            .CE
                    0130            .OS
                    0140            .LS
                    0150 ;
                    0160 ;************************
                    0170 ;* USER PORT VARIABLES *
                    0180 ;************************
                    0190 ;
                    0200 VIA        .DE $E840           ;BASE ADDRESS OF PET 6522
                    0210 ORAH       .DE VIA+$01         ;PORT A WITH HANDSHAKING
                    0220 DDRA       .DE VIA+$03         ;DATA DIRECTION REGISTER A
                    0230 PCR        .DE VIA+$0C         ;PERIPHERAL CONTROL REG.
                    0240 IFR        .DE VIA+$0D         ;INTERRUPT FLAG REGISTER
                    0250 ;
                    0260 ;*********************
                    0270 ;* SYSTEM VARIABLES *
                    0280 ;*********************
                    0290 ;
                    0300 ;ROM VERSION    4.0             UPGRADE
                    0310 ;
                    0320 KEYBUF     .DE $9E             ;       # OF KEYSTROKES
                    0330 HEXIN      .DE $FB             ;       LOW BYTE OF ADDRESS
                    0340 HEXINPUT   .DE $D754           ;$E7A7 HEX ADDRESS INPUT
                    0350 STROUT     .DE $BB1D           ;$CA1C PRINT STRING
                    0360 RDT        .DE $FFCF           ;       INPUT CHARACTER
                    0370 ;
                    0380 ;**********************
                    0390 ;* PROGRAM VARIABLES *
                    0400 ;**********************
                    0410 ;
                    0420 PET        .DE $00             ;PET START ADDRESS
                    0430 END        .DE HEXIN           ;PET END ADDRESS
                    0440 LEN        .DE HEXIN           ;PROGRAM LENGTH
                    0450 PCRWAS     .DE STORE           ;ORIGINAL PET PCR VALUE
                    0460 ;
                    0470 SENT       .DE %00100000       ;ORA WITH PCR
                    0480 TAKEN      .DE %11011111       ;AND WITH PCR
                    0490 SETUP      .DE %11000000       ;CB2 HANDSHAKE
                    0500 ;
                    0510 ;********************
                    0520 ;* BASIC RUN START *
                    0530 ;********************
                    0540 ;
0400- 00            0550 BASIC      .BY $0              ;START OF BASIC
0401- 0B 04         0560            .BY $B $4 ;          LINE LINK
0403- 0A 00         0570            .BY $A $0 ;          LINE NUMBER 10
0405- 9E            0580            .BY $9E             ;SYS
0406- 31 30 33      0590            .BY "1037"          ;DECIMAL ADDRESS
0409- 37
040A- 00 00 00      0600            .BY $0 $0 $0 ;       END OF BASIC
                    0610 ;
                    0620 ;*****************************
                    0630 ;* INITIALIZE PET USER PORT *
                    0640 ;*****************************
                    0650 ;
                    0660 START
                    0670 ;
040D- A9 FF         0680 PORTINIT   LDA #$FF            ;SET PORT A AS OUTPUT
040F- 8D 43 E8      0690            STA DDRA
0412- AD 4C E8      0700            LDA PCR             ;SAVE CONTENTS OF PCR
0415- 8D 09 05      0710            STA PCRWAS
0418- 29 DF         0720            AND #TAKEN
041A- 09 C0         0730            ORA #SETUP          ;USE CB2 AS MANUAL HANDSHAKE
041C- 8D 4C E8      0740            STA PCR
                    0750 ;
                    0760 ;*************
                    0770 ;* SETUP AIM *
                    0780 ;*************
                    0790 ;
041F- A0 04         0800 AIMSETUP   LDY #H.STR0         ;MAKE SURE THAT AIM IS
```

Listing 2

```
0421- A9 AB     0810            LDA #L,STR0          ; WAITING FOR TRANSFER
0423- 20 1D BB  0820            JSR STROUT
0426- A9 00     0830            LDA #0
0428- 85 9E     0840            STA *KEYBUF
042A- 20 CF FF  0850            JSR RDT
                0860 ;
                0870 ;*********************************
                0880 ;* GET & SEND AIM START ADDRESS *
                0890 ;*********************************
                0900 ;
042D- A0 04     0910 AIMSTART   LDY #H,STR1          ;INPUT 4 DIGIT ADDRESS
042F- A9 CB     0920            LDA #L,STR1          ; AT WHICH PROGRAM SHOULD
0431- 20 A0 04  0930            JSR INPUT            ; BE STORED IN AIM
0434- A5 FB     0940            LDA *HEXIN
0436- 20 85 04  0950            JSR PORTSEND
0439- A5 FC     0960            LDA *HEXIN+1
043B- 20 85 04  0970            JSR PORTSEND
                0980 ;
                0990 ;************************
                1000 ;* GET PET START ADDRESS *
                1010 ;************************
                1020 ;
043E- A0 04     1030 PETSTART   LDY #H,STR2          ;INPUT ADDRESS AT WHICH
0440- A9 EF     1040            LDA #L,STR2          ; PROGRAM STARTS IN PET
0442- 20 A0 04  1050            JSR INPUT
0445- A5 FB     1060            LDA *HEXIN
0447- 85 00     1070            STA *PET
0449- A5 FC     1080            LDA *HEXIN+1
044B- 85 01     1090            STA *PET+1
                1100 ;
                1110 ;**********************
                1120 ;* GET PET END ADDRESS *
                1130 ;**********************
                1140 ;
044D- A0 04     1150 PETEND     LDY #H,STR3          ;INPUT ADDRESS OF FIRST
044F- A9 FC     1160            LDA #L,STR3          ; BYTE AFTER END OF
0451- 20 A0 04  1170            JSR INPUT            ; PROGRAM IN PET
```

(Continued on next page)

Versatile Interface Adapters is not a difficult task. Better, more efficient program development will result by using the most powerful system available, even though the program will reside in a different computer. A small revision in a program no longer requires time spent with simple assemblers and tape files, or in entering object code repeatedly by hand. I would like to acknowledge the previous efforts and gains made on the PET/AIM system by Dr. John H. Miller, III and Dr. Michael Ryschkewitsch.

George Watson is a graduate student in physics at the University of Delaware. His research involves light-scattering studies of condensed matter. Microcomputers are used heavily to control a Raman spectrometer and to collect and display spectra. He may be contacted at the Physics Department, University of Delaware, Newark, DE 19711.

SIGNALMAN MARK I DIRECT CONNECT MODEM – $89.50

Standard 300-baud, full duplex, answer/originate. Powered by long lasting 9-volt battery (not included). Cable and RS-232 connector included.

EPROMS – HIGH QUALITY, NOT JUNK

Use with PET, APPLE, ATARI, SYM, AIM, etc. 450 ns. $6.50 for 2716, $12.50 for 2532. We sell EPROM programmers for PET and ATARI

5¼ INCH SOFT SECTORED DISKETTES

Highest quality. We use them on our PETs, APPLEs, ATARIs, and other computers. $22.50/10 or $44.50/20

NEW! C. ITOH STARWRITER F-10 DAISY WHEEL PRINTER

Letter quality, flawless copy at 40 char/sec. Bidirectional printing, 15-inch carriage, uses standard Diablo ribbons and print wheels.

PARALLEL – $1495, RS-232 – $1680, TRACTORS – $210

MAE SOFTWARE DEVELOPMENT SYSTEM FOR PET, APPLE, ATARI

"The Compatible Assembler"

- Professional system for development of Machine Language Programs. 31 Characters per label.
- Macro Assembler/Text Editor for Disk-based systems.
- Includes Word Processor for preparation of Manuals, etc.
- Standard Mnemonics – Ex.: LDA (LABEL), Y
- Conditional Assembly, Interactive Assembly.
- Editor has string search/search and replace, auto line numbering, move, copy, delete, uc/lc capability.
- Relocating Loader to relocate object modules.
- Designed with Human Factors Considerations.

$169.95

FLASH!! EHS Management has decided to allow $50.00 credit to ASM/TED owners who want to upgrade to MAE. To get this credit, return ASM/TED manual with order for MAE.

SMARTERM 80 COLUMN CARD FOR APPLE – $279

Upper/lower case and 80 columns. Includes 5x7 matrix character set, full ASCII keyboard, and true shift key operation.

TYMAC PARALLEL PRINTER INTERFACE FOR APPLE – $119.95

For use with Centronics, Starwriter, Prowriter, etc.

PET BASIC SCROLL PROGRAM

Scroll thru basic program using Cursor up/down keys. Specify computer. $6.00 on cassette, $9.00 on disk.

Flip 'N' File diskette storage case (50-60 disks)- $21.95

Memory Test for Apple on Disk = $9.95, on Tape = $6.95

System Saver for Apple – Fan, Surge Protection, 2 extra outlets, Apple power cord = $75.00

BMC Green Screen Video Monitor.
12 inch CRT, sharp, crisp 40 or 80 column display. = $90.00

DC Hayes Smart Modem = $235.00, Micro Modem II = $289.00, Chronograph = $225.00

C. Itoh Prowriter Printer. Better than MX80. We use constantly with our Apple and PET. Can be used on IBM, Atari, TRS-80, etc. 120 cps, friction and tractor feeds, hi resolution dot graphics, nice looking, high quality construction. Parallel – $499.00, with IEEE interface for commodore – $599.00, RS232 – $660.00

Eastern House

3239 Linda Dr.
Winston-Salem, N.C. 27106
(919) 924-2889 (919) 748-8446
Send for free catalog!

VISA MasterCard

A harvest of savings from Apple Tree Electronics

SOFTWARE

APPLE • ATARI • TRS80 • IBM

A full line of software for business, games and education **up to 35% off!**

MUSE	IUS
VISICORP	STONEWARE
ON LINE	SYNERGISTIC
EDU-WARE	HAYDEN
HOWARD	AND MANY MORE

HARDWARE

AMDEK • HAYES • MICROSOFT

	List	Our Price
32K RAM card	$293.00	$205.00
Video Term	$345.00	$279.00
Lazer Products	– – – 20% off – – –	

DISKS

Maxell	Box of 10, 5¼", SS-DD	$35.00
Verbatim	Box of 10, 5¼", SS-DD	$29.00

MONITORS

LE MONITORS	List	Our Price
9" Green	$189.00	$159.00
12" Green	$199.00	$169.00
ZENITH		
12" Green	$179.00	$129.00

Plus a full line of **AMDEK** Monitors

PRINTERS

PAPER TIGER	List	Our Price
460G	$1,094.00	$950.00
560G	$1,394.00	$1,250.00
EPSON		
MX 70	$449.00	$395.00
MX 80FT	$745.00	$595.00
MX 100FT	$945.00	$795.00

CALL FOR THIS MONTHS SPECIAL!

1-800-835-2246 EXT. 211

OR

702-452-5589

VISA MasterCard

5130 East Charleston Blvd.
Suite 5MI
Las Vegas, Nevada 89122

Phone orders welcome. Mail orders may send charge card number (include expiration date), cashiers check, money order or personal check (allow ten business days for personal or company checks to clear). Add $3.00 for shipping, handling and insurance. Nevada residents add 5.75% sales tax. Please include phone number. All equipment is in factory cartons with manufacturers warranty. Equipment subject to price change and availability. Call or write for price list.

Listing 2 *(Continued)*

```
                        1190 ;****************************
                        1200 ;* CALCULATE & SEND LENGTH *
                        1210 ;****************************
                        1220 ;
0454- 38                1230 LENGTH     SEC                     ;SUBTRACT END ADDRESS
0455- A5 FB             1240            LDA *END                ; FROM START ADDRESS
0457- E5 00             1250            SBC *PET                ; TO FIND PROGRAM LENGTH
0459- 85 FB             1260            STA *LEN
045B- 20 85 04          1270            JSR PORTSEND
045E- A5 FC             1280            LDA *END+1
0460- E5 01             1290            SBC *PET+1
0462- 85 FC             1300            STA *LEN+1
0464- 20 85 04          1310            JSR PORTSEND
                        1320 ;
                        1330 ;*********************
                        1340 ;* DOWNLOAD PROGRAM *
                        1350 ;*********************
                        1360 ;
0467- A2 00             1370 DOWNLOAD   LDX #0                  ;TRANSFER PROGRAM
0469- A0 00             1380            LDY #0
046B- B1 00             1390 NEXTBYTE   LDA (PET),Y             ;SEND BYTE
046D- 20 85 04          1400            JSR PORTSEND
0470- C8                1410            INY
0471- D0 03             1420            BNE SKIP1
0473- E8                1430            INX                     ;GOTO NEXT PAGE
0474- E6 01             1440            INC *PET+1
0476- E4 FC             1450 SKIP1      CPX *LEN+1              ;END OF PROGRAM?
0478- D0 F1             1460            BNE NEXTBYTE
047A- C4 FB             1470            CPY *LEN
047C- D0 ED             1480            BNE NEXTBYTE
                        1490 ;
047E- AD 09 05          1500 EXIT       LDA PCRWAS              ;RESTORE PET
0481- 8D 4C E8          1510            STA PCR
0484- 60                1520            RTS
                        1530 ;
                        1540 ;*****************
                        1550 ;* VIA PORT SEND *
                        1560 ;*****************
                        1570 ;
0485- 8D 41 E8          1580 PORTSEND   STA ORAH                ;PLACE BYTE AT PORT A
0488- AD 4C E8          1590            LDA PCR
048B- 09 20             1600            ORA #SENT               ;SET DATA SENT SIGNAL
048D- 8D 4C E8          1610            STA PCR
0490- AD 4D E8          1620 WAIT       LDA IFR                 ;WAIT FOR DATA RECEIVED
0493- 29 02             1630            AND #2                  ; SIGNAL
0495- F0 F9             1640            BEQ WAIT
0497- AD 4C E8          1650            LDA PCR                 ;RESET DATA SENT SIGNAL
049A- 29 DF             1660            AND #TAKEN
049C- 8D 4C E8          1670            STA PCR
049F- 60                1680            RTS
                        1690 ;
                        1700 ;*****************
                        1710 ;* INPUT ROUTINE *
                        1720 ;*****************
                        1730 ;
04A0- 20 1D BB          1740 INPUT      JSR STROUT              ;PRINT INPUT PROMPT
04A3- A9 00             1750            LDA #0                  ;EMPTY KEYBOARD BUFFER
04A5- 85 9E             1760            STA *KEYBUF
04A7- 20 54 D7          1770            JSR HEXINPUT            ;INPUT HEX ADDRESS
04AA- 60                1780            RTS
                        1790 ;
                        1800 ;*************************
                        1810 ;* INPUT PROMPT MESSAGES *
                        1820 ;*************************
                        1830 ;
04AB- 0D 50 52          1840 STR0       .BY $D 'PREPARE AIM, THEN HIT RETURN. ' 0
04AE- 45 50 41
04B1- 52 45 20
04B4- 41 49 4D
04B7- 2C 20 54
04BA- 48 45 4E
04BD- 20 48 49
04C0- 54 20 52
04C3- 45 54 55
04C6- 52 4E 2E
04C9- 20 00
04CB- 0D 0D 45          1850 STR1       .BY $D $D 'ENTER HEX ADDRESSES:' $D
04CE- 4E 54 45
04D1- 52 20 48
04D4- 45 58 20
04D7- 41 44 44
04DA- 52 45 53
04DD- 53 45 53
04E0- 3A 0D
04E2- 0D 41 49          1860            .BY $D 'AIM START? ' 0
04E5- 4D 20 53
04E8- 54 41 52
04EB- 54 3F 20
04EE- 00
04EF- 0D 50 45          1870 STR2       .BY $D 'PET START? ' 0
04F2- 54 20 53
04F5- 54 41 52
04F8- 54 3F 20
04FB- 00
04FC- 0D 50 45          1880 STR3       .BY $D 'PET END?   ' 0
04FF- 54 20 45
0502- 4E 44 3F
0505- 20 20 20
0508- 00
0509- 00                1890 STORE      .BY 0
                        1900 ;
                        1910            .EN
```

END OF THE PASS!

MICRO

Commodore Programs Move into the *Fast Lane* with PETSPEED

Petspeed — The Optimizing Basic Compiler that runs Commodore BASIC 40 times faster. You can dramatically reduce long processing times, tedious disk handling, and long print runs. No other compiler can offer the same speed, compatability and trouble-free compiling as Petspeed.

Compatible —Petspeed compiles any BASIC application and is available for any combination of 4000 and 8000 series Commodore equipment.

Faster The key that sets Petspeed apart from other compilers is **optimization.** Where most compilers merely translate from one language to another, Petspeed analyzes your source program to eliminate unnecessary complexities, thus speeding processing time. Your program is reduced to much smaller components and reassembled into a far more efficient form. Compare these optimizing features:

- 4-Pass compiler.
- Automatically uses faster integer arithmetic when possible.
- Automatically handles frequently occurring variables and arrays.
- Subroutines no longer need be located at the beginning of your program.
- Petspeed automatically calls all subroutines at maximum speed.
- Petspeed runs twice as fast as other compilers.
- Larger programs require far less memory when compiled.

Easy to Use Petspeed is as easy to use as these screen displays illustrate.

Directory BEFORE compilation.

Simply type in your program name.

Directory AFTER compilation

It isn't necessary to add compiler directives. Simply type in the program name. In less than 2 minutes, you'll see your program run up to 40 times faster.

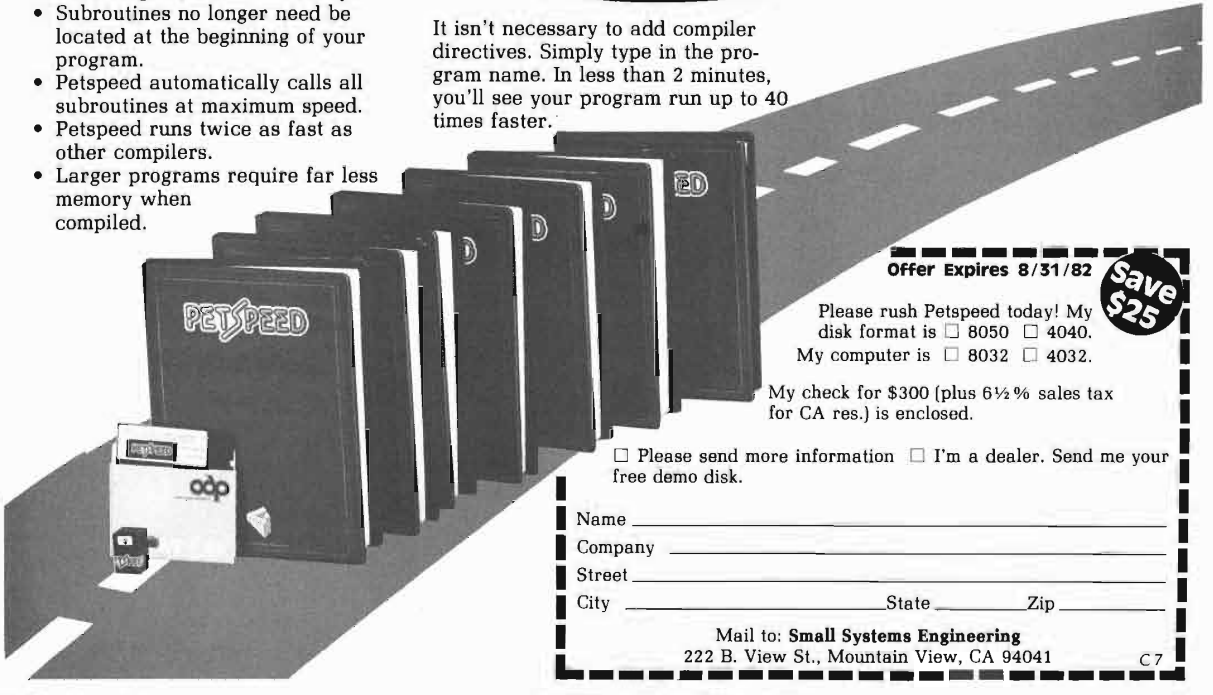
Security A security device is provided to run Petspeed, but no run-time key is necessary for compiled programs. You're free to build in your own protection. Petspeed code cannot be listed by others, so compiled programs cannot be tampered with. Your programs belong entirely to you.

Move your Commodore into the fast lane today with Petspeed. And save $25 too! Send this coupon today to

SSE

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

Dealers: Ask for our free demo disk.
Price $325



Offer Expires 8/31/82 Save $25

Please rush Petspeed today! My disk format is ☐ 8050 ☐ 4040.
My computer is ☐ 8032 ☐ 4032.

My check for $300 (plus 6½% sales tax for CA res.) is enclosed.

☐ Please send more information ☐ I'm a dealer. Send me your free demo disk.

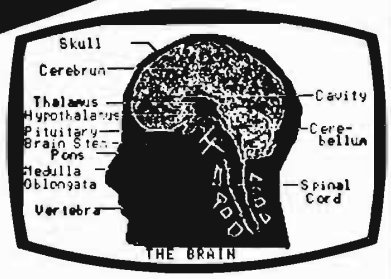
Name _____
Company _____
Street _____
City _____ State _____ Zip _____

Mail to: **Small Systems Engineering**
222 B. View St., Mountain View, CA 94041 C7

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**

Expanding File Cabinet for the Apple

by David P. Allen

The public domain "File Cabinet" is still in use by many Apple owners. Techniques to access "File Cabinet" data for use in other programs are presented.

File Cabinet
requires:

Apple II or Apple II Plus
48K
Applesoft in ROM

An axiom of executive efficiency experts is "Never handle a piece of paper more than once." When you get a paper, finish with it completely before you set it down.

There is a parallel to this idea when it comes to managing data with a computer: "Never enter the same data more than once through the keyboard." Once entered, we should be able to do just about anything we want with it — sort it, select items, make labels, edit, etc.

Several programs for sale do exactly that. They're called data managers, and most of them are very good. They are also among the most expensive software items being sold today.

One data manager has been around for almost as long as the Apple II itself. One of its best features is that it is almost free. I am talking about the reliable but often-sneered-at FILE CABINET program that originally appeared on one of the first Apple-supplied "Contributed Programs" disks. This program and its progeny have been used for many tasks by Apple users. In spite of its shortcomings (and there are a few), it continues to have a life undiminished.

One reason for this is the great volume of data that has been translated into FILE CABINET index files. Before you know it, you have spent so much time entering information into this electronic cabinet that the thought of entering all that material through the keyboard again is unthinkable. Finding myself in this position, but still wanting to move batches of information from the FILE CABINET to other application programs, I decided to investigate whether there is a way to link the FILE CABINET with other programs.

Figure 1

```
]CATALOGD1

DISK VOLUME 254

K80N
  REC#: 1
1 CITY: ALBANY NY
2 LAT.: 42.65
3 LONG.: -73.75

K80N
  REC#: 2
1 CITY: ANNAPOLIS MD
2 LAT.: 38.98
3 LONG.: -76.50

K80N
  REC#: 3
1 CITY: ATLANTA GA
2 LAT.: 33.75
3 LONG.: -84.39

K80N
  REC#: 4
1 CITY: AUGUSTA ME
2 LAT.: 44.31
3 LONG.: -69.77

K80N
  REC#: 5
1 CITY: AUSTIN TX
2 LAT.: 30.26
3 LONG.: -97.74

K80N
  REC#: 6
1 CITY: BATON ROUGE LA
2 LAT.: 30.45
3 LONG.: -91.18

K80N
  REC#: 7
1 CITY: BISMARCK ND
2 LAT.: 46.81
3 LONG.: -100.79

K80N
  REC#: 8
1 CITY: BOISE ID
2 LAT.: 43.62
3 LONG.: -116.2
```

I soon developed a simple program to do just that. Before analyzing this program, let's take a look at what we are starting with, and what we are going to wind up with. To make things clear, we'll use some real data.

The example I have chosen is data for a GREAT CIRCLE program which computes distances and bearings to various places in the country. To accomplish this task the GREAT CIRCLE program needs the identity (city and state), plus the latitude and longitude of the distant city. This information is entered into a file in the FILE CABINET program. A typical printout of the records of this file appears in figure 1. Each record contains the data on one location.

As with most Applesoft programs, the GREAT CIRCLE program works on data in the form of DATA statements. Figure 2 shows a listing of some of the data statement lines from the GREAT CIRCLE program. Our task, then, is to get the information out of the FILE CABINET records and into the data statements of GREAT CIRCLE. The program listing NA DATA WRITER EXEC does the trick.

Line 1 establishes three maxfiles; a necessity if we have been playing around with FILE CABINET which reduces maxfiles to one. Line 100 shows us what is going on. Line 300 sets up the program to wind things up after the last piece of data has been read. Lines 400 to 600 establish a file called "NA DATA WRITER", while line 700 determines the line number of the first data statement to be used in the final program. Lines 800 and 900 open up the FILE CABINET index file, in this case called "NA BEARINGS INDEXFILE". Figure 3 shows us the contents of this file as compiled by FILE CABINET. It reveals that all of the needed data is nicely arranged in the order that we need it. Depending on which version of the FILE CABINET program we are using, there is some

'housekeeping' information stored as a preamble to the meat of the file. In this case, just before the first entry, 'ALBANY NY', there is the figure 51, which represents the number of records currently existing in this indexfile. Necessary information for FILE CABINET, but a problem for GREAT CIRCLE.

Looking at NA DATA WRITER, we see that lines 1000 to 1300 read out the data from the indexfile into strings A through D. Line 1500 starts writing this information into the text file NA DATA WRITER. Lines 1600 and 1700 start the data statement with a line number and the reserved word DATA. Line 1800 increments the line number counter, line 1900 writes in the information stored in B$, C$, and D$, and line 2000 prepares for more data. Line 2100 starts us through the process over again. Note that A$ is collected once and not used. This is the housekeeping data and is collected just to get it out of the way. Consequently, we loop back to line 1100 and not line 1000. When we run out of data the program jumps to line 2400 where it prints out a final data statement containing the 'END DATA' string, which the GREAT CIRCLE program needs to complete its assignment.

Here is the way this routine is brought into play. First we enter all our data into the FILE CABINET program. We can sort, change, delete, or add records as we wish. When we have the information the way we want it, it is stored in the INDEXFILE, in this case NA BEARINGS INDEXFILE. Now we load and run our interconnecting program, NA DATA WRITER EXEC. This program must be run with the disk containing the FILE CABINET file, NA BEARINGS INDEXFILE, in the disk drive. It will create the text file, NA DATA WRITER.

We now load the program that we wish to add the data statements to, in this case GREAT CIRCLE. At this point we delete any old data statements which may exist in the block of data statement numbers we are about to add. In this example we would delete lines 3000 through 3999. We are now ready for the final step.

With the disk containing the text file NA DATA WRITER in the slot, we exec this file. This file, shown in figure 4, is added to GREAT CIRCLE and the data statements are now added. Our task is complete.

The advantage of this system is that it can be adapted to almost any program that can use data statements. We can use the FILE CABINET program to manipulate our raw data into the form we want before committing it to data statements. I'm sure you can find lots of ways to use this procedure to shuttle basic data from file cabinet files to a variety of other programs.

The author may be contacted at 19 Damon Road, Scituate, MA 02066.

Figure 2

```
3000  DATA ALBANY NY,42.65,-73.75
3003  DATA ANNAPOLIS MD,38.98,
      -76.50
3006  DATA ATLANTA GA,33.75,-84.39
3009  DATA AUGUSTA ME,44.31,-69.77
3012  DATA AUSTIN TX,30.26,-97.74
3015  DATA BATON ROUGE LA,30.45,
      -91.18
3018  DATA BISMARCK ND,46.81,
      -100.79
3021  DATA BOISE ID,43.62,-116.2
```

Figure 3:
Printout of NA Indexfile Bearings

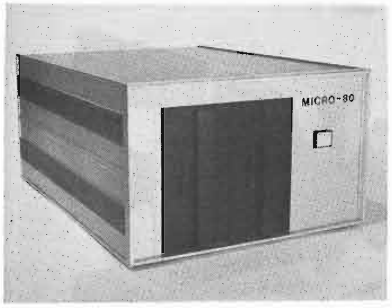
```
 51
 ALBANY NY
 42.65
 -73.75
 ANNAPOLIS MD
 38.98
 -76.50
 ATLANTA GA
 33.75
 -84.39
 AUGUSTA ME
 44.31
 -69.77
 AUSTIN TX
 30.26
 -97.74
 BATON ROUGE LA
 30.45
 -91.18
 BISMARCK ND
 46.81
 -100.79
 BOISE ID
 43.62
 -116.2
```

Figure 4: **Printout of NA Data Writer**

```
1  PRINT "MAXFILES 3": REM   CONTROL-D AFTER FIRST QUOTE
2 DEE$ =  CHR$ (4): REM   CONTROL-D
100  PRINT DEE$;"MON C,I,O"
200  REM


<<< NA DATA WRITER EXEC >>>

     by David P. Allen

       July 14, 1981


300  ONERR  GOTO 2400
400  PRINT DEE$;"OPEN NA DATA WRITER"
500  PRINT DEE$;"DELETE NA DATA WRITER"
600  PRINT DEE$;"OPEN NA DATA WRITER"
700 LINENUMBER = 3000
800  PRINT DEE$;"OPEN NA BEARINGS INDEXFILE"
900  PRINT DEE$;"READ NA BEARINGS INDEXFILE"
1000  INPUT A$
1100  INPUT B$
1200  INPUT C$
1300  INPUT D$
1500  PRINT DEE$;"WRITE NA DATA WRITER"
1600  PRINT LINENUMBER;
1700  PRINT "DATA";
1800 LINENUMBER = LINENUMBER + 3
1900  PRINT B$;",";C$;",";D$
2000  PRINT DEE$;"READ NA BEARINGS INDEXFILE"
2100  GOTO 1100
2200  PRINT DEE$;"CLOSE"
2300  END
2400 LINENUMBER = LINENUMBER + 3
2500  PRINT DEE$;"WRITE NA DATA WRITER"
2600  PRINT LINENUMBER;
2700  PRINT "DATA";
2800  PRINT "END DATA 1,1,1
2900  GOTO 2200
```

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

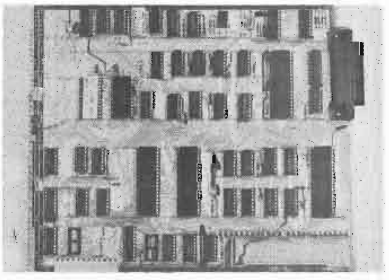
Software available in IBM single density 8″ format.

Microsoft		**Digital Research**		**Micropro**	
Basic-80	$289	PL/1-80	$459	Wordstar	$299
Basic Compiler	$329	Mac	$ 85	Mail-Merge	$109
Fortran-80	$410	Sid	$ 78	Spellstar	$175
Cobol-80	$574	Z-Sid	$ 95	Super Sort I	$195
Macro-80	$175	C Basic-2	$110	**Pascal**	
Edit-80	$105	Tex	$ 90	Pascal/MT +	$429
Mu Simp/Mu Math	$224	DeSpool	$ 50	Pascal Z	$349
Mu Lisp-80	$174	**Ashton-Tate**		Pascal M	$355
		dBase II	$595		

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		$ 80
	parallel printer and real time calendar clock	



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

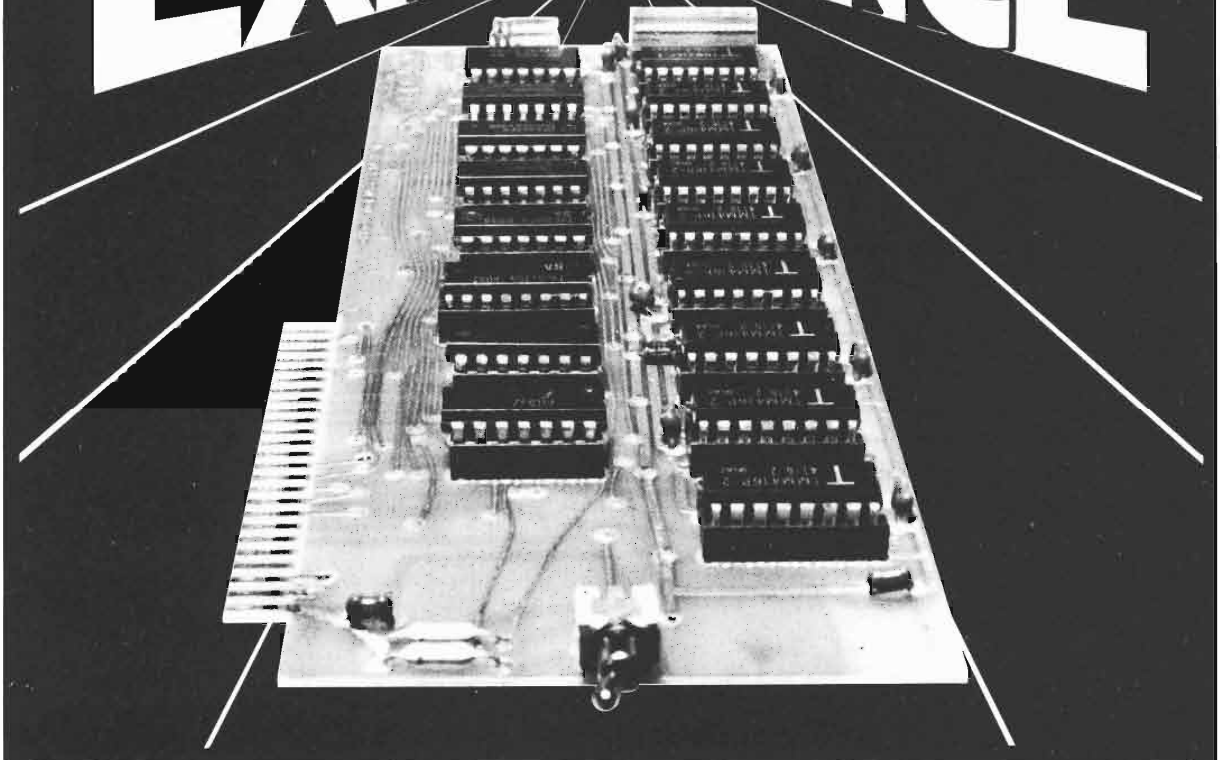
ML 82A Dot Matrix Printer **$534**
120 CPS, 80/120 columns, 9.5″ paper width, friction or pin feed

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

NEW LOW PRICE
DIRECT FROM MANUFACTURER
$120.00

EXPAND ☆ ENHANCE



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

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.

Duty Cycle Monitor for the VIC-20

by Bob Kovacs

This VIC-20 utility program determines the correct volume level when reading cassette tapes from a conventional tape recorder. Techniques are presented for using the VIC's built-in timers in conjunction with a machine-language program to measure waveform pulse widths. In addition, an interface circuit is provided.

Duty Cycle Monitor
requires:

VIC-20 with properly interfaced cassette recorder. May be modified for Apple.

Although the VIC-20 cassette tape interface was designed for use with Commodore's own data tape recorder, it can readily be adapted for use with a standard audio tape recorder. My own design (see figure 1) has proven adequate, although several variations are possible.

In general, the output of the adapter circuit will be dependent on the playback volume level. At least three conditions must be satisfied before consistent, error-free loading can be attained. First, there must be sufficient amplitude to exceed the adapter's detection threshold. Next, the level must be such that the "sync" tone at the beginning of the tape has a 50 percent duty cycle. (This tone is easily distinguished from the noisy sounds of data by actually listening to the tape. I've modified my recorder as shown in figure 2 so that I can hear these sounds during a LOAD.) Then, the variability (i.e. jitter) in both the playback frequency and duty cycle must be limited.

The ability to read tapes is a function of how they were recorded. Tapes played back on the machine on which they were recorded are generally read more easily than those recorded elsewhere. This is primarily due to variations in recording head alignment from

Figure 1: **VIC-20 Cassette Tape Recorder Adapter**

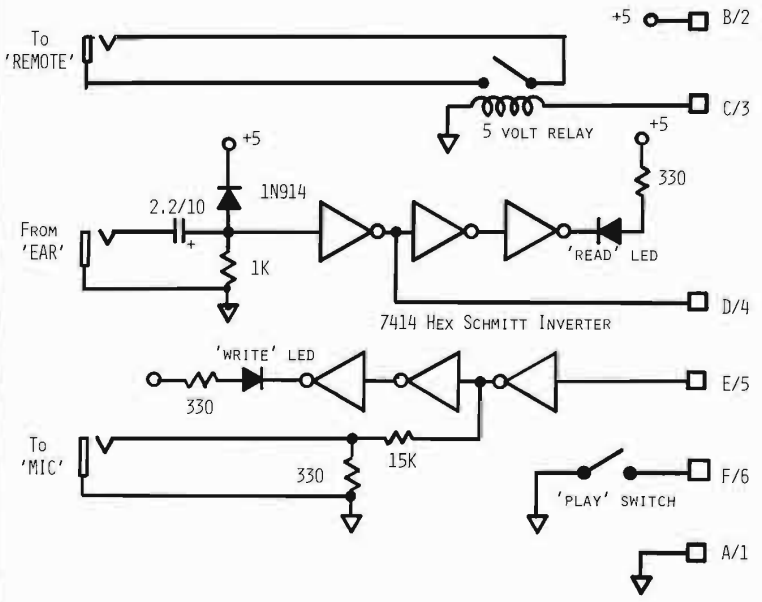
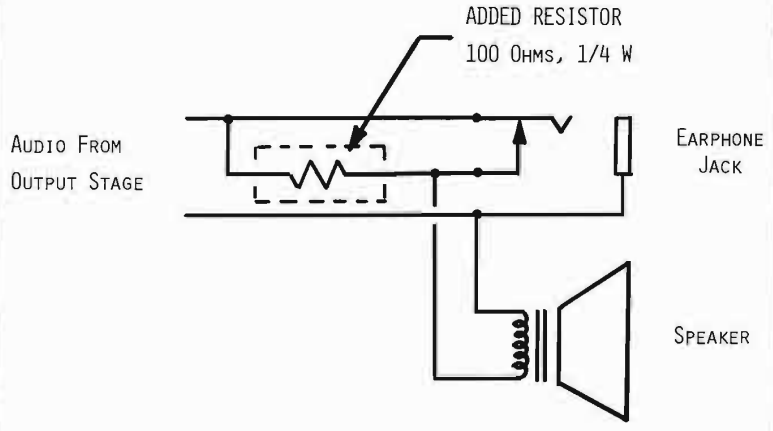


Figure 2: **Cassette Recorder Modification — Allows tape to be heard at reduced volume during a 'LOAD'**



one machine to another, which can have a significant effect on preserving the fidelity of the waveform during playback. Most recorders are provided with a small alignment screw near the record/playback heads.

Setting the volume level is usually done by trial and error. Eventually upper and lower level limits are established so that a given tape can be read consistently. Tapes that originated on other recorders can occasionally be troublesome, making the whole process frustrating. The program shown in listing 1 is an aid for establishing the proper volume level and for assessing playback jitter. It does this by separately measuring the duration of the positive and negative portions of the playback waveform. The results are displayed graphically for ease of interpretation so that a sync tone having a 50 percent duty cycle is displayed with equal positive and negative values. In addition, the stability of the display is a direct indication of phase and frequency jitter due to amplitude variations and motor speed fluctuations.

The Duty-Cycle program presented here is easy to use. Just run the program while playing the tape to be loaded. The volume level is adjusted during the sync tone until the display indicates a 50 percent duty cycle. This condition is met when the two "+" symbols line up. After the level is adjusted, the program is halted and the tape is read in using the LOAD command. (If your tape recorder also has a tone control, then it should be set for maximum treble.)

How it Works

A machine-language routine (shown in listing 2) performs the main task of measuring and displaying pulse widths. This routine is already incorporated in DATA statements of the BASIC program in listing 1. The routine was initially developed, assembled and tested on an Apple II using a John Bell dual 6522 VIA parallel board. The program was adapted to the VIC-20 by redefining the VIA and screen addresses.

The signal from the cassette tape adapter is input on pins D and 4 of the cassette I/O connector. This goes directly to the CA1 pin of VIA #2. (*Note:* VIA is short for Versatile Interface Adapter and the VIC-20 contains two of them. Each VIA contains two programmable 8-bit ports and two control lines per port. In addition, each VIA contains two 16-bit timer/counters and one shift register. See reference 2 for more details on the VIC-20 and VIA operation.) The

***Listing 1:* Duty Cycle Monitor for VIC-20**

```
5 PRINT""
10 PRINT"CASSETTE RECORDER VOL"
15 PRINT"CONTROL ADJUSTMENT PGM"
20 PRINT"ADJUST VOLUME SO THAT"
25 PRINT"BOTH '+' SYMBOLS LINE"
30 PRINT"UP DURING THE PURE"
35 PRINT"'SYNC' TONE AT THE"
40 PRINT"BEGINNING OF THE TAPE"
80 REM ---------------------
90 A=832
100 READ N:FOR M=0 TO N-1:READ H$
105 HH=ASC(LEFT$(H$,1))-48
110 IF HH>9 THEN HH=HH-7
115 HL=ASC(RIGHT$(H$,1))-48
120 IF HL>9 THEN HL=HL-7
125 H=HL+HH*16
130 POKE A+M,H:NEXT M
135 PRINT:PRINT"<HIT ANY KEY TO START>"
140 GET A$:IF A$="" THEN 140
145 L$="   [---------------]"
150 PRINT"<HIT ANY KEY TO STOP>"
155 PRINT""L$
160 FOR I=38576TO38641:POKEI,0:NEXTI
165 POKE251,0:POKE252,0
170 SYS A:GET A$:IF A$="" THEN 170
180 PRINT:PRINT:PRINT"NOW LOAD THE PROGRAM"
190 END
299 REM ---------------------
300 DATA 115
310 DATA 20,66,03,20,7D,03,20,5D
320 DATA 03,A2,00,20,88,03,20,5D
330 DATA 03,20,7D,03,20,66,03,A2
340 DATA 01,20,88,03,60,A9,01,0D
350 DATA 2C,91,20,6F,03,60,A9,FE
360 DATA 2D,2C,91,20,6F,03,60,8D
370 DATA 2C,91,AD,21,91,A9,02,2C
380 DATA 2D,91,F0,FB,60,A9,FF,8D
390 DATA 18,91,A9,00,8D,19,91,60
400 DATA A9,00,AC,19,91,D0,0A,38
410 DATA A9,FF,ED,18,91,4A,4A,4A
420 DATA 4A,48,A9,B3,CA,E8,F0,02
430 DATA A9,DF,85,D1,84,FB,A9,20
440 DATA 91,D1,68,A8,94,FB,A9,2B
450 DATA 91,D1,60
```

Listing 2

```
DUTY CYCLE MONITOR FOR VIC-20
                1000 * VIC-20 CASSETTE TAPE LEVEL ADJUST UTILITY
                1010 *
                1020 * BY BOB KOVACS - 16 MAR 82
                1030 * 41 RALPH ROAD, WEST ORANGE NJ 07052
                1040 *
                1050 *
                1060 * THIS ROUTINE DISPLAYS THE DURATION
                1070 * OF + AND - CYCLES OF THE CASSETTE
                1080 * TAPE WAVEFORM INPUT ON CA1 OF VIA2
                1090 *--------------------------------
                1100 * USE CASSETTE FILE BUFFER SPACE
                1110 *
                1120         .OR $0340     832 DECIMAL
                1130         .TA $0800
                1140 *
                1150 *       .LISTOFF
                1160 *--------------------------------
                1170 *
00D1-           1180 LINE    .EQ $D1,D2    START ADDRESS OF CURRENT
                                           SCREEN LINE
00FB-           1190 TMP     .EQ $FB       PLOT INDEX
                1200 *
1EB3-           1210 LINE9   .EQ $1EB3     START ADDRESS OF LINE #9 + 3
1EDF-           1220 LINE11  .EQ $1EDF     START ADDRESS OF LINE #11 + 3
                1230 *
9110-           1240 VIA1    .EQ $9110     USER VIA BASE ADDRESS
9120-           1250 VIA2    .EQ $9120     INTERNAL VIA BASE ADDRESS
                1260 *
                1270 * 6522 REGISTER DEFINITIONS
                1280 *
0000-           1290 INOUT.B .EQ $00       PORT B INPUT/OUTPUT REGISTER
0001-           1300 INOUTH.A .EQ $01      PORT A INPUT/OUTPUT
                                           REG W/HANDSHAKE
```

(Continued)

Listing 2 *(Continued)*

```
0002-           1310 DDIR.B .EQ $02      PORT B DATA DIRECTION REGISTER
0003-           1320 DDIR.A .EQ $03      PORT A DATA DIRECTION REGISTER
0004-           1330 T1C.LO .EQ $04      TIMER 1 COUNTER LOBYTE
0005-           1340 T2C.HI .EQ $05      TIMER 1 COUNTER HIBYTE
0006-           1350 T1.LO  .EQ $06      TIMER 1 LATCH LOBYTE
0007-           1360 T1.HI  .EQ $07      TIMER 1 LATCH HIBYTE
0008-           1370 T2.LO  .EQ $08      TIMER 2 LOBYTE
0009-           1380 T2.HI  .EQ $09      TIMER 2 HIBYTE
000A-           1390 SHIFT  .EQ $0A      SERIAL I/O SHIFT REGISTER
000B-           1400 AUXCON .EQ $0B      AUXILIARY CONTROL REGISTER
000C-           1410 PERCON .EQ $0C      PERIPHERAL CONTROL REGISTER
000D-           1420 INTFLG .EQ $0D      INTERRUPT FLAG REGISTER
000E-           1430 INTEN  .EQ $0E      INTERRUPT ENABLE REGISTER
000F-           1440 INOUT.A .EQ $0F     PORT A INPUT/OUTPUT REGISTER
                1450 *-------------------------------
                1460 *
0340- 20 66 03  1470 START  JSR NEG      WAIT FOR NEG CA1
0343- 20 7D 03  1480        JSR SETT2    START TIMER
0346- 20 5D 03  1490        JSR POS      WAIT FOR POS CA1
0349- A2 00     1500        LDX #$00     INIT LINE INDEX
034B- 20 88 03  1510        JSR PLOT     DISPLAY TIME OF NEG CYCLE
                1520 *
034E- 20 5D 03  1530        JSR POS      WAIT FOR POS CA1
0351- 20 7D 03  1540        JSR SETT2    START TIMER
0354- 20 66 03  1550        JSR NEG      WAIT FOR NEG CA1
0357- A2 01     1560        LDX #$01     INIT LINE INDEX
0359- 20 88 03  1570        JSR PLOT     DISPLAY TIME OF POS CYCLE
                1580 *
035C- 60        1590        RTS          RETURN BACK TO BASIC
                1600 *-------------------------------
                1610 * WAIT FOR POS TRANSITION ON CA1
                1620 *
035D- A9 01     1630 POS    LDA #$01
035F- 0D 2C 91  1640        ORA PERCON+VIA2 SET LOBIT
0362- 20 6F 03  1650        JSR TRAN
0365- 60        1660        RTS
                1670 *-------------------------------
                1680 * WAIT FOR NEG TRANSITION ON CA1
                1690 *
0366- A9 FE     1700 NEG    LDA #$FE
0368- 2D 2C 91  1710        AND PERCON+VIA2 CLR LOBIT
036B- 20 6F 03  1720        JSR TRAN
036E- 60        1730        RTS
                1740 *-------------------------------
                1750 * WAIT FOR ANY TRANSITION ON CA1
                1760 *
036F- 8D 2C 91  1770 TRAN   STA PERCON+VIA2 SETUP CA1 CONTROL
0372- AD 21 91  1780        LDA INOUTH.A+VIA2 CLR CA1 INTERRUPT FLAG
0375- A9 02     1790        LDA #$02     CA1 INTERUPT MASK
0377- 2C 2D 91  1800 .10    BIT INTFLG+VIA2 TEST FOR CA1 TRANSITION
037A- F0 FB     1810        BEQ .10      LOOP UNTIL FOUND
037C- 60        1820        RTS
                1830 *-------------------------------
                1840 * INTITIALIZE & START T2 TIMER
                1850 *
037D- A9 FF     1860 SETT2  LDA #$FF     INIT LOBYTE
037F- 8D 18 91  1870        STA T2.LO+VIA1
0382- A9 00     1880        LDA #$00     INIT HIBYTE &
0384- 8D 19 91  1890        STA T2.HI+VIA1  START COUNTDOWN
0387- 60        1900        RTS
                1910 *-------------------------------
                1920 * READ TIMER & DISPLAY COMPLEMENT
                1930 *
0388- A9 00     1940 PLOT   LDA #$00     CHECK TIMER FOR UNDERFLOW
038A- AC 19 91  1950        LDY T2.HI+VIA1
038D- D0 0A     1960        BNE .05      BRANCH IF NG
                1970 *
038F- 38        1980        SEC          SETUP FOR SUBTRACTION
0390- A9 FF     1990        LDA #$FF     DETERMINE PULSE DURATION
0392- ED 18 91  2000        SBC T2.LO+VIA1 IN CPU CYCLES
0395- 4A        2010        LSR          DIVIDE BY 4
0396- 4A        2020        LSR          TO LIMIT RESULT
0397- 4A        2030        LSR          BETWEEN 0 & 16
0398- 4A        2040        LSR
0399- 48        2050 .05    PHA
039A- A9 B3     2060        LDA #LINE9   LOAD SCREEN LINE OFFSET
039C- CA        2070        DEX
039D- E8        2080        INX          IS X=0 ?
039E- F0 02     2090        BEQ .10
03A0- A9 DF     2100        LDA #LINE11  ALTERNATE OFFSET
03A2- 85 D1     2110 .10    STA LINE     SET LINE POINTER
03A4- B4 FB     2120        LDY TMP,X    PREVIOUS PLOT INDEX
03A6- A9 20     2130        LDA #$20     SPACE
03A8- 91 D1     2140        STA (LINE),Y CLEAR OLD PLOT
03AA- 68        2150        PLA
03AB- A8        2160        TAY
03AC- 94 FB     2170        STY TMP,X    SAVE NEW PLOT INDEX
03AE- A9 2B     2180        LDA #$2B     '+'

DUTY CYCLE MONITOR FOR VIC-20
03B0- 91 D1     2190        STA (LINE),Y DISPLAY TO SCREEN
03B2- 60        2200        RTS
                2210 *-------------------------------
```

VIA can be programmed to detect either a positive or negative transition at the CA1 pin *via* the Peripheral Control Register. (This is just one of sixteen registers contained in each VIA.)

The approach is fairly straightforward. First set the VIA to detect a negative transition of CA1, and wait for that transition to occur. Then immediately start one of the VIA timers that counts down at the basic CPU clock rate. While the timer is counting, set the VIA to detect a positive transition of CA1, and wait for the transition. When it occurs, read the contents of the timer. Subtracting this value from the initial one results in the duration (in CPU cycle time units) of the negative portion of the input waveform. Since the ideal sync waveform is a square wave with a frequency of about 3 KHz (3000 cycles per second), its half-cycle duration is about 167 microseconds. For a CPU clock speed of 1 MHz this is equivalent to 167 timer units.

Allowing a measurement range of 0 to 255 seems reasonable and limits the arithmetic to the low byte of the 16-bit timer value. A graphic display is more desirable than a numeric one, but the VIC's display is limited to 22 columns per line. Although higher resolution could have been obtained by defining special plot characters through screen bit mapping techniques, I found that a lo-res plot was adequate. The timer measurement was divided by 16 to limit the range from 0 to 15. The "+" symbol was plotted on the screen by storing it directly in screen memory, using the time duration as an offset.

Next, this entire operation was repeated for the positive portion of the input waveform. This measurement is plotted below the previous one. When the plot symbols line up, both cycles have the same time duration.

Apparently the VIC is occasionally interrupted by the other VIA timers to perform routine housekeeping functions. No attempt was made to disable this activity and this sometimes results in a scrambled plot. A check is made on the timer high byte to help clean up this interference. Since the timer is initialized to 255 at the start of the countdown and a duration of 176 is normally expected, if the high byte is found to be nonzero, it is likely that this measurement was corrupted and the reults invalid. An invalid measurement is indicated by locating the plot symbol at the leftmost position.

The BASIC Program

The BASIC program shown in listing 1 includes the machine-language routine just described. Of the various methods considered to combine machine language with BASIC, I decided that a simplified approach was less prone to error, although less memory efficient. The first hurdle to overcome was a conflict in number bases. BASIC requires that numbers be in decimal, whereas hexadecimal is more natural for machine code. As can be seen from the listing, the machine-language routine is entered as a hexadecimal string and lines 105 through 125 perform a conversion to decimal. The resultant value is then POKEd into the RAM space allocated to the cassette file buffer. Since no file data is input during this measurement, this space remains unused and is a convenient place to put the machine-language program. (Note that it takes about four seconds to convert and store 115 bytes.)

Line 145 defines a plotting scale with time increasing toward the right with a scale factor of about 16 microseconds per column. Line 160 sets the screen color from white to black for those lines (9, 10, and 11) that will contain plot characters. Line 165 initializes two temporary page-zero locations which hold the location of the previous plot position. Before plotting the new point the old plot symbol must be erased by writing a space over it. Finally, line 170 calls the machine-language routine with the SYS command. After a pair of points are plotted, control returns to that line where the keyboard is tested for input. Hitting any key exits the program, otherwise it loops back to plot another pair of points. Remember that control returns to BASIC only if a signal is detected on the cassette input line (but you can use RESTORE to quit at any time).

Just a final word on the printout shown in listing 1. My Epson MX-80 printer does not print those special VIC-20 display control characters which show up as graphics symbols. Thus the inverse heart symbol used to indicate 'clear screen' is not printed within the quote marks on lines 5 and 150. Similarly the nine 'cursor down' inverse Q symbols did not get printed in line 155. Make sure you include them when you type in the program. Lines 140 and 170, however, are just null strings with nothing between the quote marks.

References

1. "Add a Cassette Interface to Your VIC-20," *BYTE*, March 1982.

2. "VIC-20 Programmer's Reference Guide," Commodore Business Machines, Inc.

Bob Kovacs has been captivated by microcomputers since the Apple II made its debut four years ago. He has recently been unraveling some of the mysteries of the VIC-20 and is actively involved with several microcomputer hardware and software projects. He may be contacted at 41 Ralph Road, West Orange, NJ 07052.

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

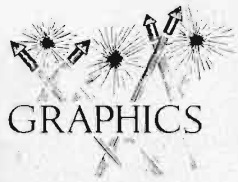
These Features Plus Many More Are Right At Your Fingertips With EDITRIX™

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.

THE ROUTINE MACHINE™



A Versatile Programming Utility for the Apple II.

Now, from the programming experts at S.D.S., an easy-to-use way of putting the POWER and SPEED of machine language routines in YOUR OWN APPLESOFT PROGRAMS!

ROUTINE MACHINE does all the work for you — no knowledge of machine language programming, whatsoever, is required. Simply choose the routine you need from an ever-growing library, and Routine Machine will effortlessly put them in your program, and all done transparently! No need to be aware of or bother with BLOAD's, HIMEM:, etc.

Best of all, with just this starter package, you'll have the routines to put High Resolution **graphics** and **sound** in your programs immediately! Also included is a versatile **print using** module to banish the "decimal point demons" forever! To round out the package, we've also included powerful **search** and **sort** routines (for single dimension arrays; Search: 1000 elements in 1 second Sort: 1000 elements in 90 seconds), and a number of other often-needed routines as well (30 routines in all).

OUR GUARANTEE

IF YOU DON'T SAVE MORE THAN THE PURCHASE PRICE OF 'ROUTINE MACHINE' IN YOUR OWN PROGRAMMING TIME IN THE FIRST 30 DAYS YOU OWN IT, SIMPLY RETURN IT FOR A COMPLETE REFUND, NO QUESTIONS ASKED!

Additional library disks titled **"Ampersoft Program Library"** are already available.

Some of the other routines in The Routine Machine (plus others not listed) are:

SWAP: Swaps two string or numeric values.

TEXT OUTPUT: Prints with no "word break" on screen.

STRING OUTPUT: Input any string, regardless of commas, etc.

ERR: Stack fix for Applesoft ONERR handling.

GOTO, GOSUB: Allows computed statements. Example: **GOTO X * 5** or **GOSUB X * 5.**

BLOAD: Load any binary file 5 times faster than normal. Hi-Res pictures load in under 2 seconds.

RESET HANDLER: Treats RESET with ONERR; or will RUN or reboot disk.

HI-RES ASCII: Character set for mixing text Hi-Res graphics.

TURTLE GRAPHICS: Versatile Hi-Res graphics routines for easy drawing of Hi-Res figures.

southwestern data systems™

P.O. BOX 582 • SANTEE, CALIFORNIA 92071 • TELEPHONE: 714/562-3670

Build performance into your system

with OS-9** software tools

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. 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. Its modular structure makes it 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

- most complete and versatile PASCAL available for the 6809
- capable of generating P-code for interpretive execution while debugging OR
- highly optimized 6809 assembly language source code output for maximum speed
- "virtual memory" P-code interpreter lets you run large PASCAL programs

CIS COBOL * Compiler**

- ideal for most demanding business applications
- features ISAM, Debug, ACCEPT/DISPLAY and Interprogram Communications modules
- retains full compatibility with CP/M software
- meets ANSI 1974 Level One COBOL standard and is GSA certified
- Also available-FORMS 2 automatic program generator for easy interative design of screen oriented applications.

BASIC09 Structured Basic Interactive Compiler**

- fastest and most comprehensive full Basic language available for the 6809
- combines standard Basic with the best features of PASCAL
- features compiler speed, interpreter friendliness and superlative debugging facilities
- option available: Run B...a ROMable run-time system for compiled Basic 09

C Language Compiler

- complete implementation of the UNIX version 7 C language
- includes 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 more information contact your computer supplier, or

MICROWARE

Microware Systems Corporation
5835 Grand Avenue, Des Moines, Iowa 50312 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.

Interfacing the Color Computer

by John Steiner

Circuits to interface the Color Computer to an RS-232 port and a motor control relay are presented. A Morse Code send/receive program is included as a demonstration.

Morse Code
requires:

TRS-80 C
Interface hardware
Code-send hardware

One of my goals was to write a program that would send machine-quality Morse code, but two good friends beat me to it. Arlin Karger and Ken Christiansen spent many hours learning how to interface the computer to other equipment. The techniques discussed here will assist you in your particular application.

Interfacing to the Outside World

The Color Computer has three output ports built into its system — the motor control relay, the RS-232 port, and the cassette port. The program listed here can use two of them. The motor control relay is used to key a transmitter. The REM statement in line 35 outlines the changes that must be made to the program to implement an RS-232 option.

The output ports on the TRS-80C are peripheral interface adapters (PIAs). These versatile ICs look like just another memory location to the processor, while allowing the programmer to use them to input or output data. Since these devices are memory mapped, a simple PEEK will read data from the outside world, while a POKE will write data out. The programmer must initialize the PIA to perform either an input or output function, upon power-up. The BASIC interpreter has done this for us, so using these locations is simple. A POKE 65313,60 will close the cassette motor control relay, allowing the cassette to run if it is in play. POKE 65313,52 turns off this relay. Another output port, the RS-232 data-out line, is controlled by memory location 65312. POKE 65312,0 and POKE 65312,2 change the logic state of the data-out pin on the RS-232 port. After the program description, several interface circuits are shown, which can be used to connect the Color Computer to a transmitter, or almost any external device.

Program Details

This program was inspired by a Morse send-receive program for the model I/III TRS-80 in *QST* magazine. After lamenting that the program would not run on the TRS-80C, Arlin attempted to duplicate its transmit functions. He ran into quite a few problems trying to adapt the program to the Color Computer. This listing barely resembles the original version, and many additions and features have been added that make it a real ham operator's program. However, credit for the algorithm belongs to J.C. Sprott, W9AV, and is gratefully acknowledged.

After Arlin finished work on the program, he found that there were timing and processing problems. The code it sent had a real "banana boat swing." Programming to precision timing in BASIC is a challenge in itself, so Arlin called upon Ken to assist in fine-tuning the program. The program now only lacked a receive function. This was not considered to be a serious handicap, since most Morse code programs have difficulty decoding all but the most precisely sent text. Ken thought it would be nice, however, to add a keyboard routine the operator could use to copy the message as it is being sent. The operator would be able to review the text of the contact at his leisure.

Meanwhile, Arlin was not content to leave the transmit section alone, so he went back to work. First he added a cursor that followed along, sending text behind the type-ahead buffer, then adding three message buffers to be called at the operator's request. These buffers are probably the most useful accessory a code operator can have. Each buffer can store up to 254 characters, so you could store nearly an entire contact in the buffer.

Program Initialization

The program REMarks indicate changes that can be made if you have a 32K machine. Essentially you are able to store a larger receive buffer. You also have two options for interfacing your computer with the transmitter. You may use either the cassette motor relay or the RS-232 interface. Included are a few inexpensive, easily built circuits that perform the interface function. More details on these later.

The program execution begins with a POKE command, which switches the microprocessor into the high speed mode. As a result, internal timing of the machine is increased by a factor of two. One caution is in order here: do not exit this program with the BREAK key. If you do, you will still be in high speed, and tape saves and loads will crash with an error. Press the <$> key to end the program. This will return the computer to normal speed. Or press the RESET button on the right rear of the computer. Another caution: if you have a Color Computer disk system, you will have to unplug it and load this program from cassette or rewrite the timing loops to work in low speed, as the disk system will not support the high-speed POKE. If you attempt to execute the POKE with a disk in the drive, you may lose all information on the disk. Some non-disk Color Computers will not support the high-speed POKE because the PIAs that handle keyboard and other interfacing duties are not fast enough. If, after running this program, the computer locks up, or operates erratically, you will have to remove the POKE commands and rewrite the timing routine. Maximum reliable sending speed will drop to about 20 WPM.

After reserving memory, program control branches to line 760, where memory is reserved, and the code look-up table is entered into memory. Though DIM statements are usually put at the beginning of a program, timing of the sending was erratic until the two array statements were moved closer to the lines associated with the send routine. We're still trying to figure out why. You will be asked to input the speed in words per minute, after which the instructions are printed. Then the dot/dash ratio is optimized in the routine starting at line 450, and the main program routine begins in the receive mode with line 200.

Operation of the Receive Section

A control loop, initiated in lines 210 to 300, gets characters from the keyboard using the INKEY$ function. Lines 230 to 250 check each key entered for the command symbols that allow branching away from the receive section. To switch between transmit and receive, just press the up arrow. To end the program, you must press the dollar sign key. Pressing the <@> key will display the transmit menu and allow you to change speed; transmit; or review, load, and/or send the three transmit buffers.

If the ampersand key (&) is found, control will branch to the display receive buffer routine, so you may review the receive text as desired. Upon entering the transmit mode, the latest receive text is added to the end of the receive buffer. As a result, this breaks the buffer into blocks of text that can be reviewed one at a time. Line 440 contains the subroutine that increments the text block counter as each group of text is being stored, and also increments if the string length reaches 200 characters. You may look at any section of the receive buffer or, by entering < 1000>, return to receive.

Line 260 helps to keep words from being broken up at the right edge of the screen. For the user, the computer becomes an electric typewriter with CRT readout.

The Transmit Section

When you are ready to transmit, you have two choices: you can press the up arrow to go directly to transmit, or press the <@> key to display the transmit menu. The transmit section begins at line 1100, where the screen clears and TX appears in the upper left-hand corner. You may begin typing, and the transmit cursor will follow

***Listing 1:* Morse Code Send-Receive**

```
5 REM THIS CW SENDING PROGRAM  WAS WRITTEN FOR THE TRS80 COLOR COMPUTER BY ARLIN
 KARGER,WDOHXQ AND THE RECEIVE,TYPING FUNCTION WAS ADDED BY KENNETH CHRISTIAN- S
EN,WOCZ.THE IDEA WAS INSPIRED  BY AN ARTICLE IN JULY 81 QST BY W9AV.
15 REM QUESTIONS MAY BE DIRECT- ED TO WDOHXQ MOORHEAD,MINN OR   WOCZ FARGO, N.D.
25 REM FOR 32K CHANGE LINE 60 TO CLEAR 18000; CHANGE LINE 760 TO DIM RA$(800)
35 REM FOR RS232 OUTPUT CHANGE POKES IN LINES 990 AND 1210 TO POKE 65312,0 AND L
INES 1020 AND 1240 TO POKE 65312,2
40 POKE65495,0
50 PCLEAR1
60 CLEAR5000
70 GOTO750
80 RX=1
90 CLS:PRINT@198,"<ENTER> SPEED(WPM)";:INPUT S
100 GOSUB630
110 SS=800/S-19
120 CLS:PRINT@10,"INSTRUCTIONS"
130 PRINT"PRESS <^> TO TRANSMIT":PRINT
140 PRINT"PRESS <^> TO RETURN TO RECEIVE":PRINT
150 PRINT"PRESS <SHIFT AND &> TO REVIEW   RECEIVE PART OF QSO":PRINT
160 PRINT"PRESS <@> FOR SPECIAL TRANSMIT  FUNCTIONS":PRINT
170 PRINT"PRESS <SHIFT AND $> TO END      PROGRAM":PRINT:PRINT
180 INPUT"PRESS <ENTER> TO CONTINUE";RR
190 GOSUB450
200 CLS:PRINT"RECEIVE"
210 RA$=INKEY$
220 IF RA$=""THEN210
230 IF RA$="^"GOSUB440:GOTO1100
240 IF RA$="$"GOTO1330
250 IF RA$="@"THEN800
260 IF POS(0)>22THEN IF RA$=CHR$(32)THEN RA$=CHR$(13)
270 PRINT RA$;
280 IF RA$="&"THEN310
290 IF LEN(RA$(RX))<200THEN RA$(RX)=RA$(RX)+RA$ ELSE GOSUB440
300 GOTO210
310 FOR RY=1TORX-1
320 PRINT RY"~~"RA$(RY)
330 NEXT RY
340 INPUT"PRESS<ENTER> TO CONTINUE";RR
350 CLS:PRINT@67,"NUMBER OF SEGMENT OF QSO"
360 PRINT@168,"TO BE REVIEWED"
370 PRINT@257,"1000 TO RETURN TO MAIN PROGRAM"
380 INPUT RS
390 IF RS=1000GOTO200
400 IF RS>RX-1 OR RS<1THEN380
410 CLS:PRINT RA$(RS)
420 INPUT"PRESS <ENTER> TO CONTINUE";RR
430 GOTO350
440 RX=RX+1:RETURN
450 D=-30
460 F=.6
470 IF S>7THEND=-20:F=.6
480 IF S>8THEND=-10:F=.6
490 IF S>11THEND=-1:F=.7
500 IF S>14THEND=1:F=.8
510 IF S>16THEND=2:F=.9
520 IF S>18THEND=0:F=1
530 IF S>22THEND=0:F=1.2
540 IF S>24THEND=10:F=1.4
550 IF S>25THEND=5
560 IF S>26THEND=5:F=1.7
570 IF S>28THEND=20:F=2
580 IF S>29THEND=5
590 IF S>30THENF=2.4
600 IF S>32THEND=5:F=2.8
610 IF S>34THEND=5:F=3.5
620 RETURN
630 IF S>30THEN90
640 IF S=30THENS=35
650 IF S=29THENS=33
660 IF S=28THENS=30
670 IF S=27THENS=28
680 IF S=26THENS=27
690 IF S=25THENS=26
700 IF S=24THENS=25
710 IF S<24THENS=S
720 IF S<5THENS=5
730 RETURN
740 GOTO 750
750 CLS:PRINT@200,"PLEASE STAND BY"
760 DIM RA$(200)
770 DIM P(47,6)
780 FOR Q=1TO47:FOR U=1TO6:READ P(Q,U):NEXT U,Q
790 GOTO80
800 PRINT:PRINT:PRINT"0= SEND FROM KEYBOARD","1= SEND BUFFER #1","2= SEND BUFFER
 #2","3= SEND BUFFER #3","4= STORE BUFFERS","5= CHANGE SPEED",,"6= REVIEW BUFFER
S","7= RECEIVE":PRINT
810 INPUT"WHICH ACTION?(1-7)";WA:CLS
820 IF WA=0THEN1100
830 IF WA=1THEN B$=MA$:IF MA$=""THENB00:GOTO940
840 IF WA=2THEN B$=MB$:IF MB$=""THENB00:GOTO940
850 IF WA=3THEN B$=MC$:IF MC$=""THENB00:GOTO940
860 IF WA=4THEN1350
870 IF WA=5THEN90
880 IF WA=6THEN1530
890 IF WA=7THEN200
900 IF WA>7THENB00
910 DATA 5,5,1,1,5,5,1,5,1,5,1,0,1,5,1,5,1,5,5,1,1,5,1,0,5,5,5,5,5,0,1,5,5,5,5,0
,1,1,5,5,5,0,1,1,1,5,5,0,1,1,1,1,5,0,1,1,1,1,1,0,5,1,1,1,1,0,5,5,1,1,1,0,5,5,5,1
,1,0,5,5,5,5,1,0,5,5,5,1,1,1,5,1,5,1,5,1,1,5,1,1,1,0,5,1,1,1,5,0
```

(Continued)

Listing 1 *(Continued)*

```
920 DATA 1,1,1,5,1,5,1,1,5,5,1,1,0,0,0,0,0,0,1,5,0,0,0,0,5,1,1,1,0,0,5,1,5,1,0,0
,5,1,1,0,0,0,1,0,0,0,0,0,1,1,5,1,0,0,5,5,1,0,0,0,1,1,1,1,0,0,1,1,0,0,0,0,1,5,5,5
,0,0,5,1,5,0,0,0,0,1,5,1,1,0,0,5,5,5,0,0,0,0,5,1,0,0,0,0,0,5,5,5,5,0,0,0,1,5,5,1,0,0
930 DATA 5,5,1,5,5,0,0,1,5,1,0,0,0,1,1,1,0,0,0,5,5,0,0,0,0,0,0,1,1,5,0,0,0,0,1,1,1,5,5,0,0
,1,5,5,0,0,0,5,1,1,5,0,0,5,1,5,5,0,0,5,5,1,1,0,0
940 X=0
950 X=X+1:V$=MID$(B$,X,1):IF V$="@"THEN1100
960 Q=ASC(V$)-43
970 IF Q<1 OR Q>47 THEN PRINT " ";:FOR U=1 TO 16*SS/5:NEXT:GOTO 1060 ELSE PRINT
V$;
980 FOR T=1TO10:NEXT
990 FOR U=1TO 6:IF P(Q,U)=0 THEN GOTO 1060 ELSE POKE 65313,60
1000 FOR Y=0 TO F*(SS*P(Q,U)):NEXT
1010 FOR T=1TO10:NEXT
1020 POKE 65313, 52
1030 FOR Y=2 TO SS:NEXT
1040 FOR T=1TO10:NEXT
1050 NEXT U
1060 FOR U=6 TO 2*SS:NEXT
1070 FOR T=1TO10:NEXT
1080 G$=INKEY$:IF G$="@"THEN1100
1090 GOTO 950
1100 CLS:PRINT "TX":B$="":X=0:V$=""
1110 CS$=CHR$(143+112)
1120 B$=MID$(B$,2)
1130 Z$=V$+CS$+MID$(B$,2)
1140 PRINT@X,Z$;
1150 E$=INKEY$:B$=B$+E$:PRINT E$;
1160 V$=MID$(B$,1,1):IF V$="" THEN 1150
1170 IF V$="@" THEN 800
1180 IF V$="^" THEN 200
1190 Q= ASC(V$)-43
1200 IF Q<1OR Q>47 THEN PRINT"";: FOR U=1 TO 4*SS/5:E$=INKEY$:B$=B$+E$:PRINTE$;:
NEXT:GOTO 1290
1210 FOR U=1 TO 6: IF P(Q,U)=0 THEN GOTO 1290 ELSE POKE 65313, 60
1220 FOR Y=0 TO F*(SS*P(Q,U)):NEXT
1230 E$=INKEY$:B$=B$+E$:PRINT E$;
1240 POKE 65313, 52
1250 E$=INKEY$:B$=B$+E$:PRINT E$;
1260 FOR Y=2 TO SS:NEXT
1270 E$=INKEY$:B$=B$+E$:PRINT E$;
1280 NEXT U
1290 FOR U= 6 TO 2*SS:NEXT
1300 E$=INKEY$:B$=B$+E$:PRINT E$;
1310 X=X+1:IF X=415 THEN X=0:CLS
1320 GOTO 1120
1330 POKE 65494,0
1340 END
1350 PRINT"ENTER BUFFER #1 (Y/N)"
1360 YN$=INKEY$:IF YN$=""THEN1360
1370 IF YN$="Y" THEN 1390
1380 IF YN$<>"Y"THEN1400
1390 CLS:LINE INPUT"ENTER BUFFER #1  ";MA$:MA$=MA$+"@"
1400 PRINT"ENTER BUFFER #2 (Y/N)"
1410 YN$=INKEY$
1420 IF YN$=""THEN1410
1430 IF YN$="Y"THEN1450
1440 IF YN$<>"Y"THEN1460
1450 LINE INPUT"ENTER BUFFER #2  ";MB$:MB$=MB$+"@"
1460 PRINT"ENTER BUFFER #3 (Y/N)"
1470 YN$=INKEY$
1480 IF YN$=""THEN1470
1490 IF YN$="Y"THEN1510
1500 IF YN$<>"Y"THEN1520
1510 LINE INPUT"ENTER BUFFER #3  ";MC$:MC$=MC$+"@"
1520 GOTO 800
1530 PRINT "BUFFER #1 =",,MA$:INPUT"PRESS <ENTER>";RR:PRINT:PRINT "BUFFER #2 =",
,MB$:INPUT"PRESS <ENTER>";RR:PRINT:PRINT "BUFFER #3 =",,MC$:INPUT "PRESS <ENTER>
";RR:GOTO800
```

***Table 1:* Parts List**

Component	Radio Shack Part Number
Q1 NPN	276-2016
Q1 PNP	276-2023
D1, D2	275-1101
RL1	275-004
R1	1 K ½ watt
R2	10 K ½ watt
RS-232 Plug	26-3020 four-pin DIN cable that Radio Shack provides for the Color Computer. Cut it in two pieces, and save the remainder for another project.

Mini Phone Jack — 2 conductor 1/8 inch to connect to motor control plug on cassette cable.

Miscellaneous hardware, etc., to connect to your transmitter.

along behind you sending what has been entered from the keyboard. Again, the INKEY$ function is used to get keys from the keyboard. The keyboard is checked between each element of each character being sent. This gives the operator plenty of access to the keyboard, but you must develop a consistent typing stroke. Typing speed will be affected by code speed. You will have to slow your typing, and use a consistent entry speed, or you may lose characters.

Each character is converted to its ASCII value, minus 43, in line 1190, where it is compared to the look-up table array. Note that the code is actually stored in this array. Initial values are one for a "dit," and five for a "dah," with correction applied after the speed is selected.

Don't use the backspace key or try to type too far ahead of the text being sent. Either of these actions will cause sending problems. Try to keep only a line or so ahead, although no problems should begin until you get more than 255 characters ahead. As characters are being sent directly from the buffer string (B$), a backspace could branch to a subroutine that eliminates the last character of B$. This, however, will affect the timing, so modifications in the send routine will result in modifications to the statements in lines 450 to 730.

To return to receive, just press the up arrow. If you wish to use the transmit buffers, press the < @ > key, which will transfer control to line 800, where the Transmit menu will be displayed. The menu allows the following options: send from the keyboard; send message 1, 2, or 3; store messages; change speed; review messages; or, return to receive.

Choosing the store message option will send program control to line 1350. The "ENTER MESSAGE #1 (Y/N)" prompt will appear. Enter a Y and you may enter a message, otherwise you will be asked if you want to enter the second message. After stepping through each message, control returns to the transmit menu. The messages are stored with an < @ > symbol at the end, which will send control immediately to the menu so you may choose another message or return to transmit or receive.

Interface Circuits

This section describes five circuits that allow you to interface the computer to the transmitter, or any device that needs a logic state change to perform a desired function. Do not key a

Figure 1

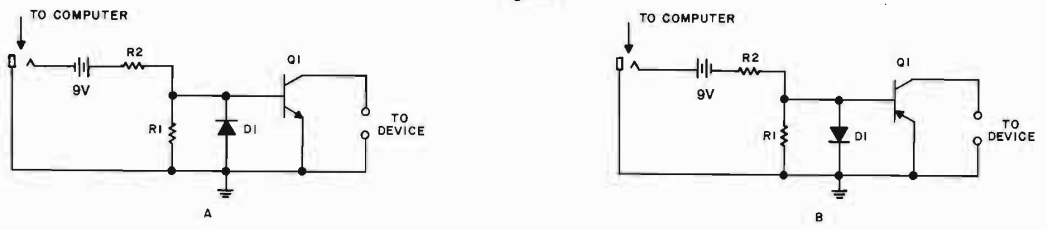
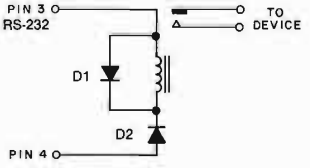


Figure 2

Figure 3



transmitter, or other high current device, directly from the motor control relay. It is not built for that kind of demand.

Figure 1 contains two circuits — either can be used to interface the computer. Figure 1A is for positive-going logic, while 1B is for transmitters with grid-block keying or other negative-going bias circuits. Figure 2 contains two circuits that can be used to connect to the RS-232 interface. Again, figure 2A is for positive logic, and 2B is for negative bias circuits. If you have grid-block keying, and you intend to use the RS-232 jack, you must reverse the data in the POKE statements from what is stated in the REM in line 35. In other words, lines 990 and 1210 must contain POKE 65312,2 and 1020, and line 1240 uses POKE 65312,0. Also, if you have grid-block keying, you must be careful not to turn on the transmitter until the program is running and ready to send. This is because the normal logic level of the RS-232 will key the transmitter on until the program initiates and shuts it off. Figure 3 contains another RS-232 circuit that uses only a relay and a couple of diodes. The advantage of using the RS-232 is that no external voltage source is required.

All parts are available from Radio Shack, and their part numbers are listed in table 1. All circuits could be easily wired onto perfboard, or mounted on a PC board.

The program listing is quite long, and if you decide not to type it in, a cassette version is available. Just send $5.00 to:

Arlin Karger
2214 South Eleventh St.
Moorhead, MN 56560

We hope this article will inspire you to develop techniques that let you interface your computer to the world around it. The applications are limited only by the imagination.

John Steiner is an electronics instructor in the Fargo, ND school system. His hobbies include programming, amateur radio, and writing. He has written articles for several publications, and is at present completing a book on electronics. John's computer system includes a 32K TRS-80C with Radio Shack disk system, and an Epson MX-80 printer.

MICRO™

OSI-C4PMF

GOBBLER MANIA

Attempt the maze eating all you can but don't get caught
Full color & sound!
5¼ in. disk $12.95

SENTINEL

Is there any enemy around the next hall? Better ready your crossbow for battle.
5¼ in. disk $19.95

We also have: ALPHA BASE, RADAR TANK, HYPER ATTACK, and many utilities too!
All in 100% machine code.
Send $1 for complete catalog to:

INTERESTING SOFTWARE
21101 S. Harvard Blv.
Torrance, Cal. 90501

ATARI

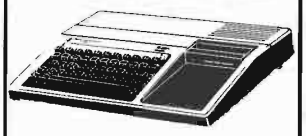
ATARI 800
16K . . . $649
32K . . . $729
48K . . . $769

410 Recorder	$76.00
810 Disc Drive	$449.00
822 Printer	$269.00
825 Printer	$589.00
830 Modem	$159.00
820 Printer	$259.00
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	$74.95
Microtek 32K RAM	$119.95
Ramdisk (128K)	$429.95
INTEC 32K Ram	$119.95
INTEC 48K Ram	$219.95
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 . . . $269
32K . . . $389
48K . . . $489

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	$79.95
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
32K Expansion	$329.00
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

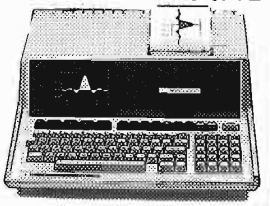
Televideo

910	$579.00
912C	$699.00
920C	$749.00
925C	$749.00
950	$939.00
802	$Call
802H	$Call
806	$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•125	$1999.00
HP•83	$1699.00
HP•85 16K Memory Module	$169.00
5¼" Dual Master Disc Drive	$1769.00
HP•87	$1769.00
Hard Disk w/Floppy	$4349.00
Hard Disk	$3440.00
"Sweet Lips" Plotter	$1149.00
80 Column Printer	$799.00
87 CP/M	$399.00
87 128K Memory	$610.00
87 Visicalc	$205.00
125 & 85 Visicalc	$179.00

HP41CV Calculator $239

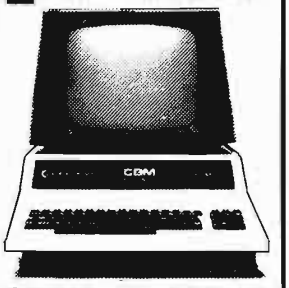
41C	$189.00
11C New Low Price	$79.00
12C	$114.00
34C	$114.00
38C	$114.00
HP•41 Printer	$340.00
HPIL CALCULATOR PERIPHERALS	
IL Modual	$104.00
Digital Cassette	$449.00
Printer/Plotter	$419.00
Card Reader	$164.00
Optical Wand	$99.00

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
7710/7730	$2399.00
3510/3530	$1789.00
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	$1629.00
IDS	
Prism	$Call
MPC Apple Parallel	
Board & Cable	$69.00
2 Meter RS232-RS232	$29.95
Cables Available For Most Interfacing Purposes	

commodore

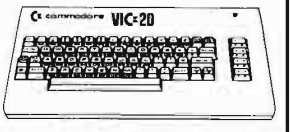


8032 $1039

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
D9060 5 Megabyte Hard Disk	$2399.00
D9090 7.5 Megabyte Hard Disk	$2699.00
8050	$1299.00
4040	$969.00
8300 (Letter Quality)	$1799.00
8023	$769.00
4022	$499.00
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

16K VIC Expansion	$99.00
Commodore Catassette	$69.00
Disk Drive	$499.00
VIC Graphic Printer	$339.00
3K Memory Expander	$32.00
8K Memory Expander	$53.00
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
VIC 1910 Radar Rat Race	$23.00

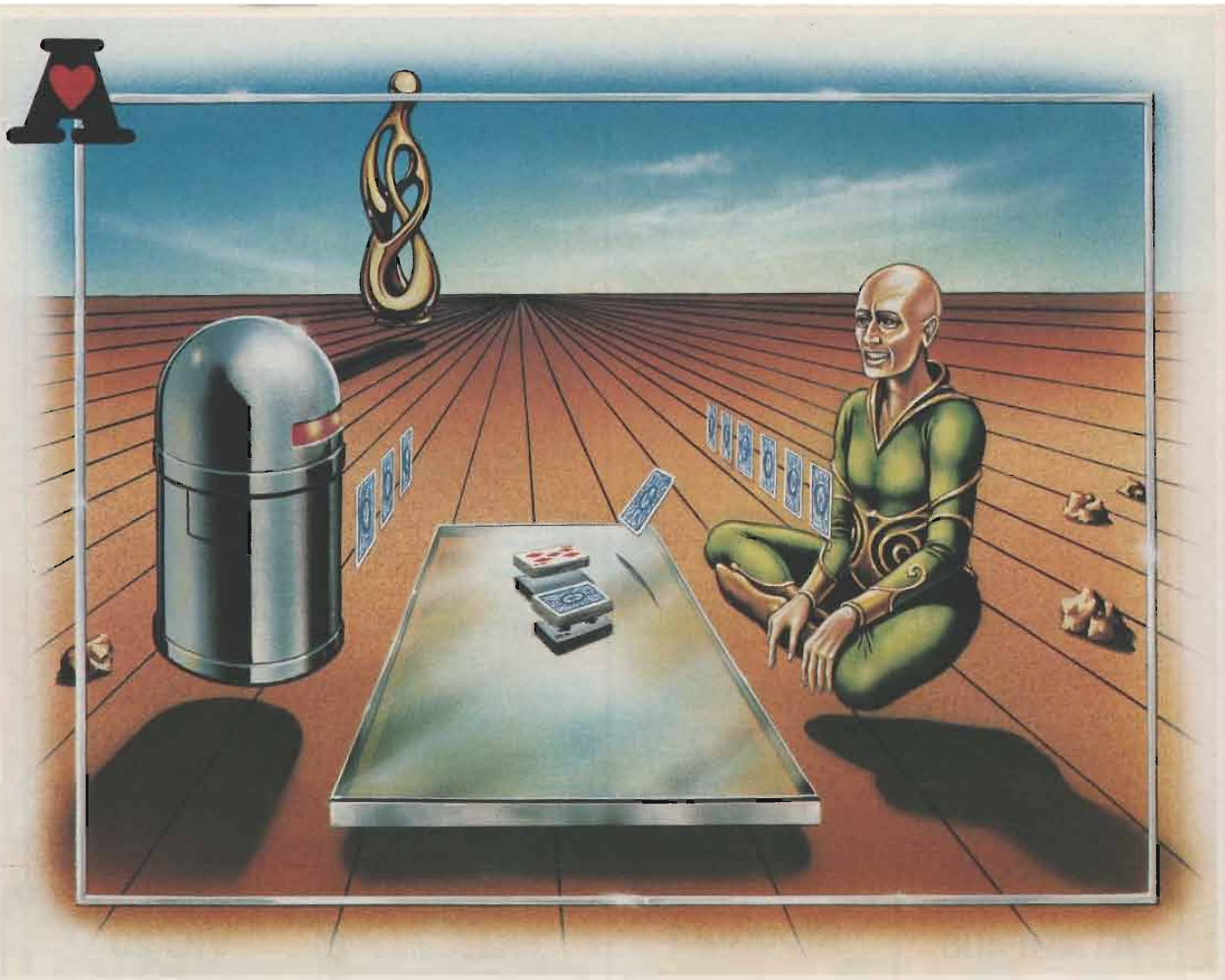
east computer mail order west

800-233-8950 CALL TOLL FREE **800-648-3311**

477 East Third Street
Williamsport, PA 17701
(717) 327-9575
Patricio Habla Espanol

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.

P.O. Box 6689
Stateline, Nevada 89449
(702) 588-5654
Franco Habla Espanol



List off your favorite card games and you'll agree not one has the fun and enjoyment offered by Gin Rummy . . . provided you're up against a tough opponent.

Good Gin players have always been agonizingly hard to find, great ones almost impossible . . . especially at the times you're in the mood. Until Now. **Now** you have **Computer Gin Rummy!** – the perfect opponent who's ready to play whenever you are, for as long as you want!

Computer Gin Rummy is no simple "Go Fish" type of competitor . . . but a master tactician who plays like a pro, who has all the strategy, knows all the moves to give you a terrific battle every time! He knocks when you least expect it, holds out to undercut you – he discards the right cards (which are wrong for you!), and suddenly lays down his hand, as if to say, "name of the game," and gins! This is real Gin Rummy where you can change your lay-offs, rearrange your cards at will, even take back a bad discard before it's fully in play.

Once you start playing **Computer Gin Rummy** you'll find all other card games seem dull and boring, because none give you such a smart competitor. Now for some extra good news. Along with GIN RUMMY, you also get KNOCK RUMMY and ONE-MELD RUMMY so you can relax with a change of pace.

Get your copy now and let's DEAL! Only $29.95 for the apple II,* at your computer store, or from:

DATAMOST INC
9748 Cozycroft Avenue
Chatsworth, California 91311
(213) 709-1202

VISA/MASTERCHARGE accepted. $1.00 shipping/handling charge. (California residents add 6% sales tax.)

*Apple II as a trademark of Apple Computer, Inc.

Makes all other card games seem boring . . .

COMPUTER GIN RUMMY

Reviews in Brief

Product Name: **Disassembler 6809**
Equip. req'd: TRS-80C Color Computer with 16K memory
Price: $19.95
Manufacturer: Soft Sector Marketing
6250 Middlebelt
Garden City, MI 48135

Description: This product is a disassembler written in BASIC and distributed in Color Computer cassette format. Besides the intended disassembly function, which basically goes to the screen, it has a printer output and a rather nice subroutine trace function. It also has a "Zap" display mode which zips through code and displays three columns of two-byte values, without regard for opcodes, etc; in other words, a screen dump. The subroutine mode allows the user to stop the current disassembly pass and trace down a subroutine, with nesting to 20 levels. At the end of a side excursion, the program will return to the same page of disassembly it was on when diverted.

Pluses: Low cost and reasonable speed of operation, thus greatly expanding the capability of the computer's BASIC to include assembly-language experimentation. Easy to use, with short learning curve. Although the program is not able to directly handle data tables, it flags inappropriate values with "bad opcode" and this should alert the user to the possibility of the presence of a data table at the address being disassembled. The experienced 6809 programmer will be able to define quickly the data table, and the disassembler quickly "syncs up" with valid code following the data.

Minuses: No problems noted except that for instructions of the form LEAX LABEL,PCR, the output fails to specify the ,PCR mode for instructions using 16-bit offset (8-bit offset instructions are correctly specified). This is a relatively minor bug, and does no harm as long as the user is aware of it.

Skill level required: In general, a user would need to be quite familiar with the computer's assembly language before being able to derive a significant amount of information from any disassembler.

Reviewer: Ralph Tenny

Product Name: **Compuvoice Synthesizer**
Equip. req'd: TRS-80 Color Computer, 16K or 32K with Extended BASIC
Price: $44.95
Manufacturer: Spectral Associates
141 Harvard Avenue
Tacoma, WA 98466

Description: The *Compuvoice Synthesizer* is a completely software-based phoneme speech generator for the TRS-80C. It is a machine-language routine that resides in high memory. Access to the synthesizer is *via* USR calls. The phonemes are generated in software and are sent to the audio circuit in the monitor. Using the synthesizer is not difficult. Strings are defined that contain special phoneme symbols. These strings are the arguments in the USR call. For example, X$="/AAYT/":A$=USR0(X$) is all the code that is required for the computer to pronouce the word "eight". You can redefine the strings and pronounce new words and phrases. String length is limited only by BASIC's 255-character limit.

Pluses: The routine is easy to load and execute. The phonemes are easily defined and generated. The process of learning to operate the synthesizer teaches the programmer the techniques of artificial speech generation. Though based on international language, the phonemes have been changed to single key characters for easy entry. The synthesizer needs no hardware modifications and uses only 8336 bytes.

Minuses: The actual speech is difficult to understand at best. I spent a lot of time trying to make it say certain phrases and was not often successful in having an interested observer understand what it was saying. Certain phrases are easier to understand than others. Even the demonstration program provided, which speaks the numbers from one to nine, had numbers that were difficult to understand.

Documentation: An excellent five-page manual explains the operation of the program, and loading and use in BASIC programs. Also included are instructions on creating speech using phoneme synthesis. Within only a few minutes, I had created a program that allowed me to enter and edit strings to be spoken.

Skill level required: Knowledge of BASIC programming, especially string handling techniques. No previous knowledge of speech synthesis is required.

Reviewer: John Steiner

Product Name: **Dot Matrix Serial Impact Printer Model 8510 Prowriter**
Equip. req'd: Any computer with either parallel or serial interface
Price: $740.00
Manufacturer: C. Itoh Electronics, Inc.
5301 Beethoven Street
Los Angeles, CA 90066

Description: A dot matrix printer using a 7×9 matrix for alphabetic characters; an 8×8 matrix for graphics. Both parallel and serial (to 9600 baud) interfaces are standard. Input buffer holds 1.5K bytes. Printing speed is 100 CPS or 44 LPM (80-character lines). Both sprocket and friction feed are standard. Maximum paper width 9.5", print line 8". Four type faces are standard: pica, elite, compressed, and proportional (10, 12, 17, and 14 cpi). All are available in boldface; all except proportional in elongated

(double-width). Bit-image graphics standard: 8 dots in matrix addressable by bit.

Pluses: Well-formed characters, true lower-case descenders, German, Swedish, and some Japanese characters. Extensive choice of options — both hardware and software — including 16 horizontal tabs, vertical tabs to any line, line-feed selectable in 1/144-inch increments.

Minuses: None serious. Print quality not quite equal to a daisy wheel printer although excellent for dot matrix. A wider carriage and higher printing speeds would be convenient.

Documentation: "Preliminary" 60-page manual. Complete and well-written. The few typographical errors do not appear at critical points.

Skill level required: Minimal, if the computer has a suitable output port. Considerable programming ability needed to take full advantage of all options.

Reviewer: Rolf B. Johannesen

Product Name:	**"SLIM" Single Board Computer p/n 81-260**
Equip. req'd:	Normal assembly tools for kit; depends upon application with assembled and tested version.
Price:	$199.95 - assembled and tested $169.95 - kit $ 39.95 - bare board

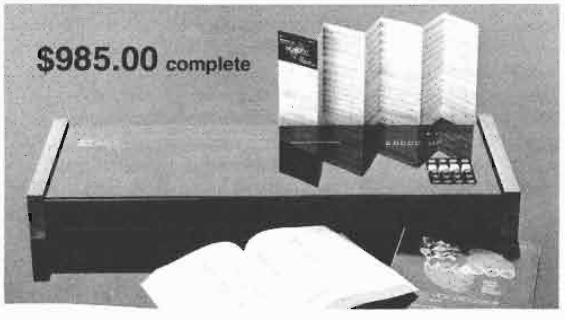
ENGINEERS/TECHNICIANS

THE MICRO 68000 IS DESIGNED FOR YOU!

COMPLETE, READY-TO-GO SYSTEM INCLUDES:

☐ 6 amp switching power supply ☐ Keyboard ☐ Display - Hex & Binary ☐ Pete Bug keyboard monitor ☐ Optional Macs Bug CRT monitor ☐ Attractive cabinet ☐ Dual RS232 interface ☐ 32 bit parallel I/O ☐ Versabus compatibility ☐ The only system that provides for direct entry of 68000 machine code.

CSA *For information call (714) 566-3911*
Computer System Associates
7562 Trade Street, San Diego, CA 92121



Manufacturer: JOHN BELL Engineering, Inc.
P.O. Box 338
Redwood City, CA 94064

Description: *"SLIM"* is an acronym for Single-board Large-scale Integration Microcomputer. It gives more power than a KIM-1 on a 4.5" × 6.5" PC board, but there is no keyboard or display. (An external terminal is used if the JBE 4K monitor ROM is used in the EPROM socket.) Otherwise, the user must furnish a mode of communication. A fully loaded *SLIM* has four 2114 RAM chips, one EPROM socket which may be jumpered for 2716/2516 or 2532 EPROMs, and two 6522 VIAs. Sixteen port lines and four handshake lines from each 6522 are brought to four separate 16-pin sockets, and the industry-standard dual 22-pin edge connector duplicates the KIM expansion connector pinout, except for signals which are unique to KIM. In addition, the address and data lines are fully buffered, and the address decoding scheme from KIM is used. However, since the memory map is significantly different, KIM software would require considerable modification to be used. The circuit design of this product is well done, and relatively minor modifications can be made to effect major changes in board function and capability. The JBE Monitor [$39.95 in a 2532 EPROM] has provision for 110-2400 baud.

Pluses: Good design, standard size, KIM/AIM bus compatibility, and easy expansion. Kit and bare-board users will have little trouble expanding to 8K of read/write memory directly; with additional decoding, 32K or 48K of memory can be added easily with modern memory technology. Similarly, the I/O space and ROM space can be further decoded.

Minuses: Bare-board users may have difficulty finding a 16 MHz crystal [a 7493 TTL counter divides this down to 1 MHz], and should try to purchase the crystal with the board.

Documentation: Skimpy, but adequate for the experienced hacker or engineer.

Skill level required: Even users who buy the assembled and tested unit must have above average experience to apply this product. I found this point to be common with all single-board computer/controller products.

Reviewer: Ralph Tenny

Product Name:	**DTL BASIC**
Equip. req'd:	32K CBM/PET with CBM disk to compile. Some BASIC and adequate memory to hold object file required at run-time. Both require a cassette-port dongle protection device.
Price:	$350; Run-time dongle $50
Manufacturer:	Canadian Micro Distributers 365 Main St. Milton, ONT L9T1P7, Canada

Description: PET BASIC compiler. Takes as input any normal BASIC program file on disk and converts it into a machine-language equivalent. The new file may be loaded and run normally. Uses a 4K run-time module, but compacts source lines. As a result, any program will still fit in memory after being compiled. Compiles 1-2 lines/second. Various listings and information can be printed during compilation. Errors are always shown. At run-time, errors

Reviews in Brief *(continued)*

halt with the usual message and a decimal address. An Error Locate program calculates the matching source line number. The major benefit is speed. Even optimized BASIC programs execute twice as fast as usual, unless slowed by peripherals. Well-structured programs execute at triple speed. DTL BASIC also includes a special integer mode. Programs using only integer variables execute twenty times faster than usual. Pseudo-ops are included for changing some or all variables in existing programs to integers during compilation. A side effect of compilation is that the program becomes difficult to view or change. DTL BASIC also requires a special connector be in place at all times while the compiler or compiled programs run. If a second optional connector is placed on the other casettte port during compilation, the program may later be run with it alone. It comes in standard and custom-keyed versions.

Pluses: Nearly bug-free and very easy to use. Re-compiles with a few keystrokes. Integer converter is helpful, though not automatic. Variables may be seen and changed normally. Compatible with packages that extend BASIC with non-standard syntax. Stopped programs may be continued with SYS, not CONT. The STOP key may be disabled without disturbing the clock. Compiler may be backed up for safety.

Minuses: Major disadvantage is the required run-time dongle. Run-time protection should be a user option. Not able to pass variables between modules of multi-module programs.

Documentation: Quite readable and fairly brief, all in a nice binder. Could use a summary page.

Skill level required: Almost none for first use.

Reviewer: Jim Strasma

Product Name:	**Screen Writer**
Equip. req'd:	Apple II or Apple II Plus (16K memory card optional)
Price:	$125.00
Manufacturer:	On-Line Systems 3675 Mudge Ranch Rd. Coarsegold, CA 93614
Author:	Michael D. Shetter
Copy Protection:	Yes
Language:	6502 Machine Language

Description: A word processor with extensive features at an extremely reasonable price.

Pluses: Provides the owner of a minimal Apple with upper- and lower-case characters and an up to 70-character display with no additional hardware. This is done through the use of the hi-res screen and generation of hi-res characters. By an electrical modification, the shift key can be used as on a standard typewriter (otherwise ESC is used for upper/lower case). The program uses the disk as virtual memory allowing the creation of a single document as large as disk space will permit. If a 16K memory card is present, the additional memory will be used. Has all the usual commands; search and replace, cursor movement, appending files from the disk, a delete/get buffer. Also allows for tab sets. Optionally generates a key click sound through the Apple speaker.

The program consists of two modules: the editor and the runoff (for printing). The user can obtain rough draft printing directly from the editor, but the listing does not contain format control The runoff program contains many of the most desirable print control features affecting margins, paging, text positioning, headings, hyphenation, and type style.

The package also includes a form letter capability for merging two text files, one containing the letter and the other the data base of names, addresses, etc.

A printing spooler routine permits the printing of one document while editing a second. This is accomplished by first creating an output file which contains all of the correctly formatted text. This file is fed to the printer while the user continues to edit a second document.

Minuses: Because the program uses the hi-res screen for display, the insertion of text in the middle of existing text can be annoyingly slow. While there is an input buffer which continues to accept characters as they are printed, it is possible to type faster than the display, so that the incoming text is not visible. After a pause, the display will catch up with the typed input. By moving the cursor to the bottom of the screen, this delay can be avoided since the amount of hi-res memory to be moved is reduced.

Skill level required: After some training, any Apple user can get maximum benefit using this word processor.

Reviewer: David Morganstein

MICRO

WINCHESTER FOR MOTOROLA EXORCISOR/MDOS

☐ 10 MB Winchester hard disk runs MDOS on Motorola Exorcisor System ☐ No modification to MDOS required ☐ MDOS based software stays alive ☐ All user software operates without modification ☐ Optional SA-801R flexible diskette drive system ☐ Optional 10 MB removable cartridge.

CSA *For information call (714) 566-3911*
Computer System Associates
7562 Trade Street, San Diego, CA 92121

NEW 6809 SYSTEM!

Now, for about the same price as you would expect to pay for the memory capacity alone, you can have a complete single board computer with these features:

- ✲6809 CPU, 1MHz clock
- ✲192KB RAM included, sockets for 64KB more
- ✲84X24 display of a 7X12 character font
- ✲Keyboard interface for an un-encoded switch matrix
- ✲Floppy controller for two 5" drives, single or double sided, up to 80 tracks
- ✲Parallel printer port
- ✲Serial I/O port
- ✲General purpose 8-bit parallel I/O port
- ✲Parallel expansion port
- ✲Dimensions: 8.6 by 10.3 inches

The FLEX operating system is supported by our device drivers. BASIC, PASCAL, and C are available for FLEX. The device drivers (in EPROM) include advanced features like auto-repeat for the keyboard, and track buffering for the disks. Commented source code of all EPROM contents is supplied.

For more information, send a stamped self-addressed envelope and we will send you a configuration guide that explains how to set-up a system. An assembled board is purchased by sending check or money order for $735 per board. (California add 6% sales tax).

Chandler Microsystems
22051 COSALA
MISSION VIEJO, CA 92691

FLEX, trademark Technical Systems Consultants, Inc.

INTERFACE
for
COLOR COMPUTER
(Radio Shack, Tandy Corp.)

- permits simultaneous operation of printer and modem
- serial and parallel ports
- plugs into cartridge port

Print Videotex
$195.00

Dealer Inquiries Invited

Micromed Instrument Systems
2715 Dawson Avenue
Wheaton, MD 20902

Olympic Sales Company

SERVING YOU SINCE 1947

Main Showroom & Offices:
216 South Oxford Avenue
Los Angeles, CA 90004

WE HONOR
VISA and MASTERCHARGE

TELEX: 67 34 77
ORDER DESKS open 7 Days a Week!
7:00 AM to 7:00 PM Mon thru Sat
Sunday Noon to 5:00 PM
Order Desks: (213) 739-1130

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

NO ONE UNDERSELLS OLYMPIC SALES
Write & request our new 112 pg catalog-please include $1.00 to defray postage & handling-includes many more items-from TV's to Watches!

All goods subject to availability; this ad supersedes all previous ads; we are not responsible for typographical errors; we will meet or beat any advertised prices if the competition has the goods on hand.
Minimum shipping and handling $4.95.
All orders subject to verification and acceptance.

apple computer

We are an authorized servicing Apple dealer for Apple II & III
Immediate delivery on all models—we carry an enormous inventory of Apple products at all times!

Immediate delivery
16K–32K–48K–64K–Plus or Integer in stock!
NEW, IMPROVED APPLE III 128K VERSION

Large Inventory of:
Disk with controller DOS 3.3
Second Disk Drives
Pascal–Fortran–Cobol languages
Dow Jones & Quote reporter
Graphics Tablets
Visicalc for Apple II & III
Smarterm 80 column card
Micromodem II by DC Hayes
and more. . .

ATTENTION: Immediate delivery
WE ALSO CARRY SOFTWARE!
Personal Software
Peach Tree Software
Microsoft
Micropro
Innovative
American
System Plus
and more. . .

NOW IN STOCK!

HEWLETT PACKARD
2 NEW DELUXE CALCS FROM HP! Slim, shirt-pocket styling

NEW! HP-11C	Advanced Programmable Scientific LCD *Retail $130.00*	Your Cost: $114.95
NEW! HP-12C	Advanced Programmable Financial LCD *Retail $150.00*	Your Cost: $129.95

	Retail	Your Cost
HP-125 New Microcomputer 64K CPU/Terminal/Keyboard	3750.00	3195.00
HP-85 Microcomputer–built-in printer/monitor	3250.00	2498.00
HP-83 Microcomputer–built-in monitor	2250.00	1799.00
HP-2631B Printer, dot mtrx (ask for optns)	3950.00	3199.00
HP82905A 80 col printer, dot matrix	945.00	755.00
HP-2601A Letter quality prter, daisy wheel		3495.00
HP-82901M Dual master (256KB disk drive)	2500.00	1999.00
Call us for the lowest prices on 7 disk drives	Call us	Call us
HP-41CV New! 2.2K bytes of memory	325.00	256.00
Card Reader for 41CV	215.00	171.00
Printer for 41CV	385.00	294.00
Quad Ram	95.00	84.95
Optical Wand for 41CV	125.00	99.95
HP-41C Calculator	250.00	188.95
Memory mod. for 41C		26.95
HP-97 Progrmble printer	750.00	579.95
HP-34C Progrmble scientific	150.00	117.95
HP-38C Progrmble business R/E	150.00	119.95

We have the complete line of accessories, etc.

XEROX

Model 820-1 (5¼") 8" $3750.00
64K COMPUTER & WORD PROCESSOR
AS LOW AS $2995.00
Special discount available to Schools & Institutions—Inquire! Required software add'l.

NEW–FAMOUS CORVUS DISK DRIVES–5, 10, 20 MEGABYTES with fantastic new OMNINET Network
Call us for the best prices in the USA! System

Texas Instruments

New–1982 Model with Home Computer full typewriter-style keyboard,
TI-99/4A New KEYBOARD U/L case & more! *Retail* $350.00 *Your Cost* $369.95

$369.95

New keyboard

10" color monitor for 99/4	650.00	319.95
32K Exp. mem. module	399.95	314.95
Extended Basic, a MUST for 32K module	100.00	75.00
Speech synthesizer	150.00	129.95

We carry a large inventory of software, & accessories Your Cost

TI-30-2 LCS Stu Slide NEW	18.95
TI-35SP LCD SCIENTIFIC	22.50
TI-40 LCD Sci/NEW	28.95
TI-57 Prog. Scientific	39.95
TI-58C 480 Step, Prog.	89.95
TI-59 960 Step, Prog.	179.95
PC-100C Print/Plot	149.95
LCD-Programmer/NEW	59.95

ATARI Computer

Large inventory of periphls, access. etc.

		Retail	Your Cost
400	SPECIAL PRICE! 16K	595.00	339.95
	No language inc., opt'l basic,		54.95
800	16K Computer	1080.00	759.95
	SPECIAL! ATARI 800 48K Computer	1250.00	869.95

OHIO SCIENTIFIC

C8PDF–48K Retail:$3495.00 Y/C: $3195.00
- Dual 8" Drives
- 64 col x 32 line/color
- .7 MIPS – *FAST!*
- Many more stnd features

Fortran & Pascal available
Many other OSI products–at discounted prices

PRINTERS

	Retail	Your Cost
• DIABLO (Letter Quality)		
630 R102 bi-directnal/tractors	2965.00	2699.00
1640K109 keyboard, tractors	3072.00	2899.95
630 RO Receive only	2710.00	2499.95
1650K136 keyboard/tractors	3220.00	2999.95
• CENTRONICS dot matrix		
700-9 Parallel, heavy duty	1460.00	1199.95
704-9 Serial, heavy duty	1795.00	1599.95
737-1 Parallel	995.00	799.95
737-3 Serial	1045.00	899.95
704-11 Parallel	1870.00	1695.00
P-1 Electrostatic	495.00	189.95
• PAPER TIGER		
460	995.00	895.00
460G graphics	1094.00	969.95
560	1295.00	1099.00
560G graphics	1394.00	1195.00
445	795.00	695.00
445G	894.00	789.00

EPSON PRINTERS

	Retail:	Your Cost
MX80	645.00	539.95
Optional Graftrax Chip 80		95.00
MX80 FT	745.00	659.95
MX80 + GRAFTRAX 80	695.00	579.95
MX80 FT + GRAFTRAX 80	795.00	689.95
MX100	995.00	789.95

WE ALSO HAVE . . .

	Retail	Your Cost
• NOVATION Modems		
CAT	199.95	159.95
D-CAT	199.95	159.95
APPLE-CAT Direct connect	349.95	314.95

SANYO MONITORS High resolution

13" Color (new) high quality	550.00	419.95
12" Green phosphorous	360.00	259.95
12" Black & white	340.00	239.95
15" Black & white	370.00	259.95
9" Black & white (the best seller)	235.00	169.95

AMDEK (Leedex) High Quality Monitors

100	12" B/W, 12 MHz	179.00	139.95
100-G	12" Green, 12 MHz	199.00	174.95
300-G	12" Green, 18 MHz	249.00	199.95
Color I	13" Color, NTSC comp. input, audio amp & speaker	449.00	339.95
Color II	13" Color, RBG input, hi res graphics, speaker	999.00	599.95

- HAZELTINE Video Display Terminals
- SHUGART Disk Drives
- DEC VT100 & VT103

Call us for your DISCOUNTED price TODAY!

NEW! From TI–Series 10 Personal Information Terminal *Retail* 995.00 *Your Cost* 795.00

Structured Programming in 6502 Assembly Language

by Kim G. Woodward

This discussion of structured programming demonstrates how to use it to improve and simplify your programming methods.

Wouldn't it be nice if someone came along and gave you some good advice about assembly language or BASIC programming? The best advice I've heard lately has been based on the idea of structured programming. What is it, and what can it do for you?

For years the bane of programming has been that the programmer who writes the program is often not the one who will maintain it. Payroll programs are a good example of this problem. For one reason or another, over the life of a piece of software, many modifications may be made. The design, coding, and testing of a program follows the 40-20-40 rule: The design takes 40 percent of the time it takes to finish a program, coding takes 20 percent, and testing takes the last 40 percent. What is not obvious is that over the life cycle of a piece of software, the whole effort to put a program together takes only 20 percent of the time. The other 80 percent is taken up in maintenance (debugging and modification). Structured programming can make this phase much more efficient.

What Is Structured Programming?

Structured programming limits a programmer to a finite set of control structures (elements such as loops, "if" statements, etc.). A program's flowchart is sometimes considered to be the program's logic or set of control structures. A flowchart, however, is one of the most dangerous devices ever conceived by the programming industry. When a design is made, a flowchart is usually drawn up to describe what the program is to do logically. But once the coding is done from that flowchart, the flowchart is usually never updated to reflect any changes made to the program. As a result, a flowchart often does not reflect the actual program that it is supposed to describe. Use of that flowchart may create errors in the program when changes are made.

Structured programming is based on the mathematically-proven Structure Theorem (due in original form to Bohm and Jacopini), which states that any program with one entry and one exit is equivalent to a program that contains as logic structures only the following: sequences of two or more operations, decision (IF a THEN b ELSE c), and repetition of an operation (DOWHILE a). Variations include the case statement, which is a form of "if-then-else;" and the "do-until" loop, which is a variant of the "do-while." A flowchart representation of these constructs is shown in figure 1. The blocks can represent single or multiple statements, or one of the accepted control constructs. Over the years since structured programming was introduced, the software industry has proven over and over that these constructs reduce the complexity of the actual code writing, make the code easier to read, maintain or change, and reduce the programming problem to a number of easily defined modules.

If a programmer confines himself to the control structures of structured programming, and refrains from any others, he will find no need for a flowchart. This may be a little hard to live with from the viewpoint of the older programmers. The colleges and technical schools used to teach that a flowchart had to be drawn first before code could be put down. However, a simpler type of documentation results when you write software using structured programming and pseudo-English such as:

```
open a file
read a record
while (the record is not the end) do
   process the record
   if (the record is from a female) then
      denote female record
   else
      denote male record
   endif
   output processed information
   read a record
enddo
close the file
end of program
```

A flowchart of the above process is shown in figure 2. The "while-do-enddo" process is very easily defined. In a "while-do" construct you test the condition before entering the loop. If the condition is true, you do the body of the condition and come back for another test of the condition. However, if the condition is false, then the program goes to the next line following the while-do construct. Likewise, the "if-then-else-endif" is easily defined. If the test of the condition is true, the then clause is executed. The else clause of the "if-then-else-endif" construct is executed when the results of the condition are false.

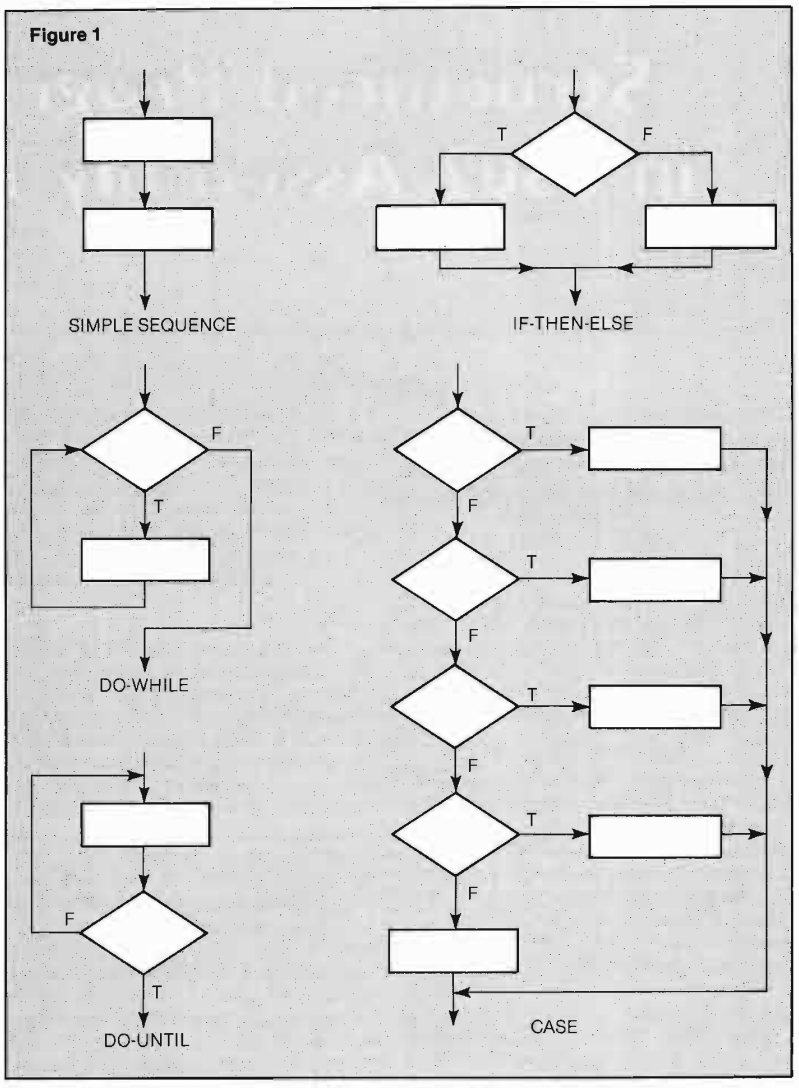
Pseudo-English is a hybrid language that uses the vocabulary of one language, English, and the overall syntax of another, structured programming. Each step of a process is described in English at whatever level of detail is appropriate at the particular level of design. This type of documentation turns out to be easier to update than a flowchart could ever be. Each pseudo-English line could represent many lines of actual programming code. This introduces the concept of leveling in the documentation. As an example, the line reading "output processed information" may represent many hundreds

of lines of pseudo code at the next level down. If I were to code this program in BASIC using the allowable control constructs, it would look like this:

```
1000 REM ** open a file **
1010 REM ** read a record **
1020 IF NOT(designation-for-end?)
     THEN 1130
1030 REM ** process the record **
1040 IF NOT(female-record?)
     THEN 1070
1050 REM ** denote female record **
1060 GOTO 1090
1070 REM cont.
1080 REM ** denote male record **
1090 REM cont.
1100 REM ** output processed infor-
     mation **
1110 REM ** read a record **
1120 GOTO 1020
1130 REM cont.
1140 REM ** close the file **
1150 END
```

The advantages of programming in this manner cannot be overstated. In BASIC the use of REM statements with ** will designate modules that have not yet been coded. When you begin writing the code that corresponds to the module, you simply remove the ** from the REM statement and insert the code. If your module is a subroutine, you remove the ** and then insert a GOSUB. You may object to the frequent use of REM as a dummy statement (such as the continue in FORTRAN). However, the use of dummy statements allows you to insert code at will into the control constructs if you have a reasonably intelligent line renumberer. BASIC control constructs corresponding to the structured programming constructs in figure 1 are shown in figure 3.

Figure 1



Now that we have a handle on what can be done in BASIC programming, how do we apply the same constructs to 6502 assembly-language programming?

What is the Problem in Assembly Language?

One problem in using decision points in 6502 assembly language is how to define the branches. A slightly more annoying problem is the 128-byte limit on the branch. We will assume that the flag registers will be set before the decision point is reached. Therefore, we can set up a number of constructs corresponding to the legal comparisons of the BASIC language. The table in figure 4 shows the flag results from a comparison between the A register and a memory location referred to by the CMP instruction. Thus the pseudo-English (with some 6502 assembly language) for three of the allowed constructs are:

(1) IF-THEN-ELSE-ENDIF

```
        setup for condition
        branch on condition to LBLZ
        JMP LBL1
LBLZ NOP
        do true part
        JMP LBL2
LBL1 NOP
        do false part
LBL2 NOP
```

(2) WHILE-DO-ENDDO

```
LBL1 NOP
        setup for condition
        branch on condition to LBLZ
        JMP LBL2
LBLZ NOP
        do body portion
        JMP LBL1
LBL2 NOP
```

(3) REPEAT-UNTIL-ENDO

```
LBL1 NOP
        do body portion
        setup for condition
        branch on condition to LBLZ
        JMP LBL1
LBLZ NOP
```

Note that in each case we have allowed for constructs that exceed 128 bytes. To illustrate the use of these constructs, suppose I had the following pseudo-English statement:

IF a = b THEN increment z ELSE decrement z ENDIF

This statement could be rendered, following our convention, in assembly language as:

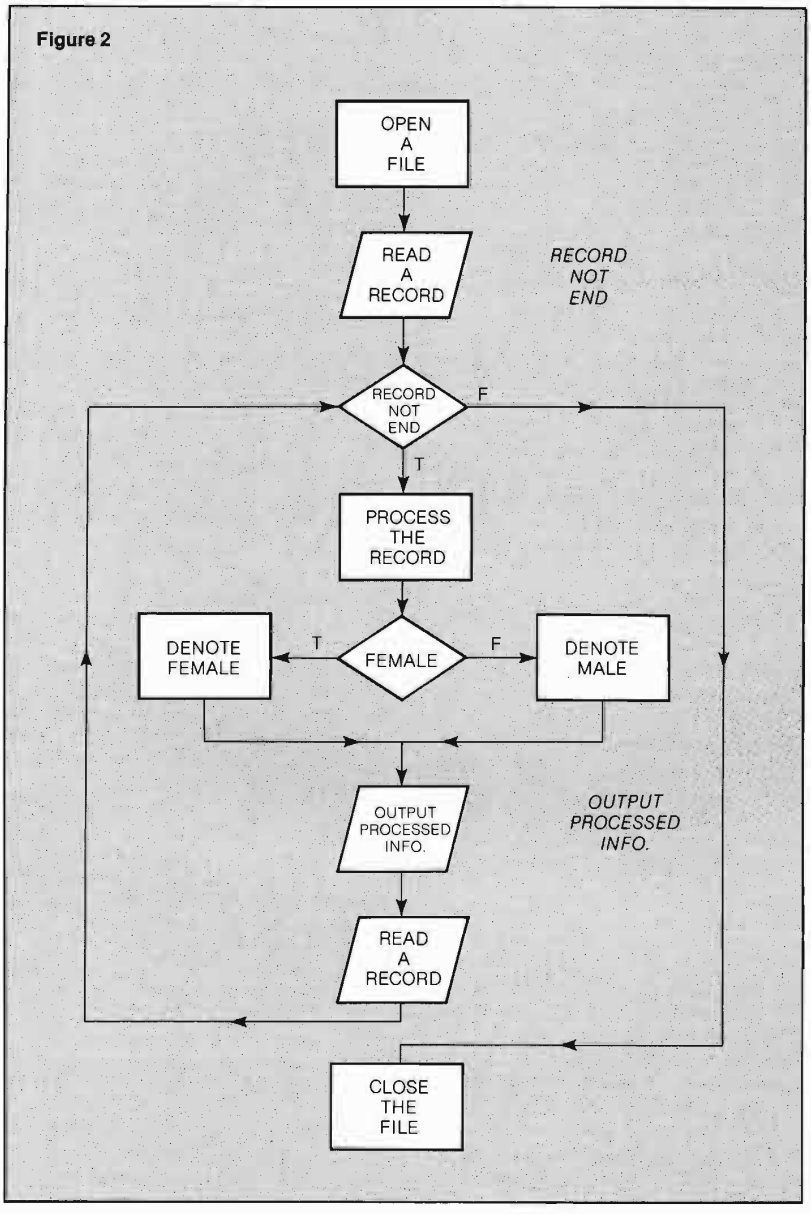
```
      LDA A
      CMP B
      BEQ LBLZ
      JMP LBL1
LBLZ NOP
      INC Z
      JMP LBL2
LBL1 NOP
      DEC Z
LBL2 NOP
```

In each of the allowed constructs the branching is done on the inverse of the condition tested for. If we set up the conditions properly and allow for greater than 128-byte branches the conditional tests are:

(1) if a < b then part-a else part-b endif

```
      LDA A
      CMP B
      BCC LBLZ
      JMP LBL1
LBLZ NOP
      do part-a
      JMP LBL2
LBL1 NOP
      do part-b
LBL2 NOP
```

Figure 2



(2) if a <= b then part-a else part-b endif

```
      LDA A
      CMP B
      BEQ LBLZ
      BCC LBLZ
      JMP LBL1
LBLZ NOP
      do part-a
      JMP LBL2
LBL1 NOP
      do part-b
LBL2 NOP
```

(3) if a <> b then part-a else part-b endif

```
      LDA A
      CMP B
      BNE LBLZ
      JMP LBL1
LBLZ NOP
      do part-a
      JMP LBL2
LBL1 NOP
      do part-b
LBL2 NOP
```

(4) if a = b then part-a else part-b endif

```
      LDA A
      CMP B
      BEQ LBLZ
      JMP LBL1
LBLZ NOP
      do part-a
      JMP LBL2
LBL1 NOP
      do part-b
LBL2 NOP
```

(5) if a > b then part-a else part-b endif

```
      LDA A
      CMP B
      BEQ LBLX
      BCS LBLZ
LBLX JMP LBL1
LBLZ NOP
      do part-a
      JMP LBL2
LBL1 NOP
      do part-b
LBL2 NOP
```

(6) if a >= b then part-a else part-b endif

```
      LDA A
      CMP B
      BCS LBLZ
      JMP LBL1
LBLZ NOP
      do part-a
      JMP LBL2
LBL1 NOP
      do part-b
LBL2 NOP
```

(7) while a<b do part-a enddo

```
LBL1 NOP
     LDA A
     CMP B
     BCC LBLZ
     JMP LBL2
LBLZ NOP
     do part-a
     JMP LBL1
LBL2 NOP
```

(8) while a<=b do part-a enddo

```
LBL1 NOP
     LDA A
     CMP B
     BEQ LBLZ
     BCC LBLZ
     JMP LBL2
LBLZ NOP
     do part-a
     JMP LBL1
LBL2 NOP
```

(9) while a<>b do part-a enddo

```
LBL1 NOP
     LDA A
     CMP B
     BNE LBLZ
     JMP LBL2
LBLZ NOP
     do part-a
     JMP LBL1
LBL2 NOP
```

(10) while a=b do part-a enddo

```
LBL1 NOP
     LDA A
     CMP B
     BEQ LBLZ
     JMP LBL2
LBLZ NOP
     do part-a
     JMP LBL1
LBL2 NOP
```

(11) while a>b do part-a enddo

```
LBL1 NOP
     LDA A
     CMP B
     BEQ LBLX
     BCS LBLZ
LBLX JMP LBL2
LBLZ NOP
     do part-a
     JMP LBL1
LBL2 NOP
```

(12) while a>=b do part-a enddo

```
LBL1 NOP
     LDA A
     CMP B
     BCS LBLZ
     JMP LBL2
LBLZ NOP
     do part-a
     JMP LBL1
LBL2 NOP
```

(13) do part-a until a<b enddo

```
LBL1 NOP
     do part-a
     LDA A
     CMP B
     BCC LBLZ
     JMP LBL1
LBLZ NOP
```

(14) do part-a until a<=b enddo

```
LBL1 NOP
     do part-a
     LDA A
     CMP B
     BEQ LBLZ
     BCC LBLZ
     JMP LBL1
LBLZ NOP
```

(15) do part-a until a<>b enddo

```
LBL1 NOP
     do part-a
     LDA A
     CMP B
     BNE LBLZ
     JMP LBL1
LBLZ NOP
```

(16) do part-a until a=b enddo

```
LBL1 NOP
     do part-a
     LDA A
     CMP B
     BEQ LBLZ
     JMP LBL1
LBLZ NOP
```

(17) do part-a until a>b enddo

```
LBL1 NOP
     do part-a
     LDA A
     CMP B
     BEQ LBLX
     BCS LBLZ
LBLX JMP LBL1
LBLZ NOP
```

(18) do part-a until a>=b enddo

```
LBL1 NOP
     do part-a
     LDA A
     CMP B
     BCS LBLZ
     JMP LBL1
LBLZ NOP
```

Let's use an example to see how these constructs work. Suppose that we need a routine to input and assemble characters to a buffer. We have a character-by-character input routine known as 'get'. Our rule is that we will fill the buffer up with each character we encounter until the first carriage return ($0D). The routine which inserts into the buffer is called 'put'. If along the way we encounter an escape character ($1B), then we will enter an escape routine called 'escp' instead of putting to the buffer. The pseudo-English version is as follows:

```
get a character
WHILE (character is not CR) DO
  IF (character is ESC) THEN
    do escape sequence
  ELSE
    add character to buffer
  ENDIF
  get a character
ENDDO
return
```

The assembly-language routine corresponding to this pseudo-English version is as follows:

```
     JSR GET          get a character
LB1  NOP                    while...
     CMP #$0D
     BNE LBZ
     JMP LB2
LBZ  NOP                    ...DO
     CMP #$1B               IF...
     BEQ LBLZ
     JMP LBL1
LBLZ NOP                    ...THEN
     JSR ESCP
     JMP LBL2
LBL1 NOP                    ELSE...
     JSR PUT
LBL2 NOP                    ENDIF
     JSR GET          get a character
     JMP LB1
LB2  NOP                    ENDDO
     RTS                    return
```

In Conclusion

I have shown that both 6502 assembly language and BASIC can use structured programming constructs. I have also shown that by designing the program with pseudo-English statements, the structured programming constructs of necessity stand out. I have shown that structured programming allows you to modify a program with little or no change to the logic. This method allows for simplicity of maintenance and thus reduces the cost factor of the maintenance portion of the program's life cycle.

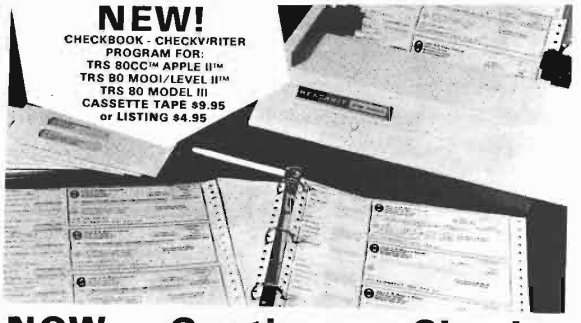
The author may be contacted at 6526 Delia Dr., Alexandria, VA 22310.

M.P. Computer Services Corporation
2396 Encinal Station
Sunnyvale, California 94087
(408) 735-0871

	Retail	Your Price
P.F.S.II "NEW IMPROVED"	$125.00	$ 96.00
P.F.S. REPORT	95.00	74.00
P.F.S. GRAPH "NEW" (Interfaces with P.F.S. and VisiCalc)	125.00	96.00
VISICALC 3.3	250.00	225.00
VISITERM	100.00	80.00
VISIDEX	250.00	195.00
VISIPLOT	200.00	158.00
SUPER DISK COPY II	30.00	25.00
PRO EASYWRITER/MAILER COMBO	300.00	225.00
EASYSPELLER	175.00	137.00
THE DICTIONARY	99.50	74.00
CREATIVE FINANCING	195.00	143.00
REAL ESTATE ANALYZER II	195.00	143.00
THE BOOKKEEPER MASTER	89.95	66.00
THE HOME ACCOUNTANT	74.95	60.00
TYPING TUTOR II	24.95	19.95
SUPER DISK COPY III	30.00	26.00
APPLE 21	24.95	21.95
CRAPS	24.95	21.95
CROSSFIRE	29.95	24.97
SABOTAGE	24.95	21.95
THREE MILE ISLAND	39.95	33.97
STAR THIEF	29.95	23.97
Hardware		
EPSON MX 80 PRINTER	599.00	525.00
EPSON MX 100 PRINTER	995.00	799.00
OKIDATA 82A PRINTER		510.00
OKIDATA 83A PRINTER		799.00
PROMETHEUS VERSAcard	199.00	167.00
16K RAM CARD	169.00	105.00
AMDEK VIDEO 100 MONITOR	179.00	117.00
AMDEK VIDEO 300 MONITOR	249.00	208.00
AMDEK COLOR I MONITOR	449.00	384.00

Send check or money order. CA residents add 6% sales tax. Add $2.00 for postage. All items subject to availability. Send $.50 for our catalog or a free catalog with your order.

NEW!
CHECKBOOK - CHECKWRITER PROGRAM FOR:
TRS 80CC™ APPLE II™
TRS 80 MOD I/LEVEL II™
TRS 80 MODEL III
CASSETTE TAPE $9.95
or LISTING $4.95



NOW. . .Continuous Checks

That Can be Used With or Without Your Computer!!
The Best in A Home Checking System

That's right. Continuous Checks fan-folded in a 3-to-a-page desk set design. And they can be computer printed, handwritten or typewritten — whichever suits the quantity or situation.

SPECIAL DESIGN

Our checks are not a high-volume business form adaptation. They're specially designed **Computer/Manual Checks** for the home user. And they're **easy to use**, too. No need to change printer form width when loading. Our checks are the same **9½" width** as standard tractor feed printer paper. Check tear down size is the same as the classic **personal-sized check** issued by all banks.

COMPLETE HOME SYSTEM

With this system you can print the bulk of your monthly checks on your computer using our **program**. Your **checks** and stubs can then be stored in our attractive Data Ring Binder **Checkbook**. Later, if you have a few checks to **write**, there's no need to load them into a printer - just write a check at your desk as **shown** above. And you can mail your checks in our **dual windowed envelopes** to eliminate addressing chores.

UNIQUE

You won't find **continuous** checks like these **anywhere**. And, our special small quantity printing process will give you **excellent quality** and appearance. Standard **color-coordinated** imprinting and encoding is as shown above (logo can be omitted) on **blue, grey, green, or buff** checks.

PRICED RIGHT

Two Hundred checks are just **$29.95** (envelopes **$14.95**). Five Hundred checks are **$49.95** (envelopes **$27.95**). Data Ring Checkbooks are only **$4.95**.

Special "ORDER NOW" Offer

So that you can start using your computer immediately to pay your monthly bills, we'll make you a special package offer. If you order directly from this ad, we can send you:

200 Checks, 100 Envelopes, A Binder, and Program for $49.95
— OR —
500 Checks, 300 Envelopes, A Binder, and Program for $74.95

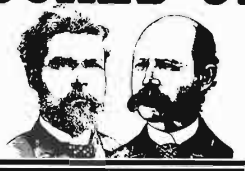
You'll save as much as **$9.85** over the separate purchase price. And with the special package you can begin your monthly checkbook balancing and bill paying as soon as you receive your checks. Just enclose a voided check (for encoding information) with your payment (VISA - MasterCard orders must show signature, expiration date, and account number). Or, send today for samples.

E-MAIL ADDRESS: SOURCE TCH-411 microNET CIS 70270,236

SYNERGETIC SOLUTIONS
PHONE: (813) 646-6557
4715 SHEPHERD RD.
DEPARTMENT M91
MULBERRY, FL 33860

TRS 80™ Tandy, Inc. Apple II™ Apple Computer, Inc.

DON'T BUY SOFTWARE THAT'S
LOCKED UP!



Beagle Bros MICRO SOFTWARE

All Beagle Bros Apple Utilities are BACKUP-ABLE, LISTABLE, CUSTOMIZABLE and fully compatible with normal Apple DOS.

APPLE MECHANIC
SHAPE WRITER/ZAP UTILITY by Bert Kersey

SHAPE EDITOR: Add professional hi-res animation to your programs. Design shapes & custom type characters, automatically written into shape tables. Many type fonts on disk & listable demo programs showing how to use shape tables for games & impressive hi-res CHARTS & GRAPHS. A valuable time-saving utility and Apple learning tool.

BYTE ZAP: A MUST utility. Rewrite any byte on a disk. Optional Hex/Decimal/Ascii display and input. Create illegal file names. Restore deleted files. Inspect, repair and protect disks. Change DOS. Clear illustrated instructions show how data is stored and how to access it.

MORE: A disk PACKED with useful music, text & hi-res tricks for use in YOUR PROGRAMS.

$29.50 Includes Apple Tip Book #5 and Peeks & Pokes Chart

UTILITY CITY
21 UTILITIES ON ONE DISK
by Bert Kersey

LIST FORMATTER makes custom listings with page breaks; each statement on new line, if-then's called out and loops indented. MULTI-COLUMN CATALOG in any page-width. Put invisible commands in programs. Alphabetize & store info. Make trick & invisible file names. Append programs. Convert hex. Dump text to printer. Auto-post Run Number/Date in programs...More: 21 LISTABLE PROGRAMS Total!

$29.50 Includes Apple Tip Book #3 and Peeks & Pokes Chart

ALPHA PLOT
HI-RES GRAPHICS/TEXT UTILITY
by Bert Kersey & Jack Cassidy

HI-RES DRAWING: Create pictures and charts on both hi-res pages; all appendable to YOUR PROGRAMS. Relocate any portion of a picture. Compress hi-res; store images in 1/3 DISK SPACE. Superimpose images too.

HI-RES TEXT: Upper/lower case with descenders. PROPORTIONAL SPACING. No tab limitations. Adjustable letter height, spacing & color. Multi-directional typing for graphs.

$39.50 Includes Apple Tip Book #4 and Peeks & Pokes Chart

DOS BOSS
DISK COMMAND EDITOR
by Bert Kersey & Jack Cassidy

A classic utility you will ENJOY. Rename commands/error messages. PROTECT PROGRAMS (unauthorized save-attempt produces "Not Copyable" message). LIST-PREVENTION too. One-key program-run from catalog. Change Disk Volume heading to your title with or without volume number. Fascinating documentation. Hours of good reading & experiments.

All changes may be appended to your programs, so that anyone using your disks (booted or not) will be using DOS the way YOU formatted it.

$24.00 Includes Apple Tip Book #2 and Peeks & Pokes Chart

TIP DISK #1

100 programs from Beagle Bros Tip Books 1, 2, 3 & 4— Hi-Res/Lo-Res/Text/Sound. All listable, copyable and changeable; each teaches another fascinating Apple programming trick!

(Note: No Tip Book with Tip Disk #1)

$20.00 With Peeks & Pokes Chart

GOTO your Apple Dealer.

Most Apple Dealers carry our software. If yours doesn't, he can have it in his store for you within just a few days through **Beagle Bros** or **Softsel.**

Or Order by Phone:
24-hour **TOLL-FREE** Order Desk:
Visa/MasterCard/COD* Orders, call
Nationwide: **800-854-2003** ext.827
California: **800-522-1500** ext.827
Alaska/Hawaii: **800-854-2622** ext.827
(ORDERS ONLY PLEASE) *COD, add $3.00.

VISA
master charge the interbank card

EACH DISK includes Our famous 11x17 PEEKS & POKES Chart and a different Apple Tip Book, each one a Gold Mine of juicy Apple info!

Or Mail us a check:
(or Visa/MC No. & Exp. Date)

☐ **Alpha Plot** ☐ **Utility City**
☐ **Dos Boss** ☐ **Apple Mechanic**
☐ **Tip Disk** ☐ **Game Pack 1-4**

Add $1.50 shipping (on any size order) and 6% tax if California. MAIL TO—
BEAGLE BROS, Dept.O
4315 Sierra Vista
San Diego, Ca 92103
Please add $4* if outside North America.

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

Pattern-Matching with the 6502 on the Apple

by Charles F. Taylor, Jr.

Pattern-matching algorithms attempt to find one or more occurrences of a given character string in a specified range of memory. This article presents elementary and advanced algorithms.

Pattern Matching

Developed on an Apple II Plus, but no Apple-specific features used. Should be usable on other 6502-based microcomputers.

In this article I discuss both elementary and advanced pattern-matching algorithms and present relocatable implementations of each in 6502 assembly language. Although these programs were developed using an Apple II Plus, no Apple-specific features were used; the resulting programs should therefore be usable on the AIM, SYM, KIM, or other 6502-based microcomputer.

You may often associate pattern matching with applications such as text editing. I was motivated to pursue the topic for quite a different reason: I wanted to find and modify the jump table used for processing commands in an assembler. As I am often at least as interested in methodology as in results, I did some research on the subject and would like to share my findings.

In pattern matching we are concerned with two strings, the pattern and the target. Each string consists of a sequence of bytes. Each byte will be assumed to be devoid of any meaning other than its numerical value (0-255 decimal); for the purposes of this discussion, however, the bytes in each string will be referred to as characters and will be represented as letters in the figures.

Elementary Pattern Matching

It is easy to tell whether a pattern occurs at a particular starting position in the target string. The first character of the pattern is lined up with that starting position in the string, and corresponding characters are compared until a mismatch is found, or until the last character of the pattern has been matched. The elementary pattern-matching algorithm thus begins by aligning the pattern with the first character of the target. If a match occurs, fine. If not, the pattern is aligned with the second character of the target, and so on. The algorithm is given in a form of structured pseudo-code in figure 1, and illustrated by example in table 1. In table 1 the * indicates the pattern character at which the first mismatch occurred in each step.

Implementation on the 6502 of the algorithm of figure 1 presented several problems. Comparison of successive characters is easy, provided that the length of the pattern is restricted to 255 characters so that the Y register can be used for post-indexed addressing. This is certainly a reasonable restriction. Determining when the target string is exhausted would also be easy if the target string were restricted to 255 characters, but this was rejected as an unreasonable restriction. Such a restriction might be reasonable for a line editor with length limited to 255 characters, but it is not sufficiently general for our purposes here.

It therefore appeared as if a 16-bit comparison would be necessary each time through the main loop to determine whether or not the target string has been exhausted. This would obviously be a slow procedure with an 8-bit microprocessor. Consequently, I decided to post a sentinel or End-Of-String (EDS) character at the end of the target string so that the end of the target can be detected more easily. This requires that the target string reside in RAM as opposed to ROM, but this restriction was deemed acceptable as a design trade-off. As long as a sentinel was being used for the target string, I decided to use one for the pattern string as well. Characters displaced by the sentinels are saved on the stack and replaced upon completion of the search.

If both the target and pattern were restricted as ASCII characters, selection of character for use as a sentinel would be easy: simply use $FF, or some other unused code. Allowing each character in the pattern and target strings to take on any value from $00 to $FF makes the problem more difficult.

For a pattern restricted in length to 255 characters, there must be at least one character (two-digit hexadecimal number) which does not occur in the pattern. The problem is to find it. A solution is to start with $FF as a candidate sentinel and to compare it with each character in the pattern; if a match is found, the candidate is decremented and the process repeated until a suitable sentinel is found. Typically the process takes only one or two passes.

For the target string, there is no guarantee that a sentinel exists that does not also occur somewhere in the target (because the target may consist of more than 255 characters). The solution is to use whatever sentinel is selected for the pattern. Whenever the sentinel character is found in the target string, perform a 16-bit comparison to determine whether the target string is exhausted. This procedure takes more code, but it will be relatively fast because only rarely will the sentinel occur in the target (except in bizarre cases).

The elementary pattern-matching program is shown in listing 1. To use the program the starting address of the pattern must be placed in locations

Figure 1: **Elementary Pattern-Matching Algorithm**

```
BEGIN
BEGTAR := 0;
J := 0;
WHILE BEGTAR(=N-M DO
        WHILE TARGET(BEGTAR+J)=PATTERN(J) DO
                J := J + 1;
                IF J(=M THEN GOTO PATTERN.FOUND
        ENDWHILE;
        BEGTAR := BEGTAR + 1;
        J := 0
ENDWHILE;
PATTERN.NOT.FOUND:  RETURN 0;
PATTERN.FOUND:  RETURN BEGTAR;
END.
```

Table 1: **Example of Elementary Pattern-Matching Alorithm**

```
PATTERN:  a  b  a  a  b  c

          0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16

TARGET:   a  a  b  b  a  a  b  a  a  b  a  a  b  c  a  b  c

Step: 1  a  b* a  a  b  c
      2     a  b  a* a  b  c
      3        a* b  a  a  b  c
      4           a* b  a  a  b  c
      5              a  b* a  a  b  c
      6                 a  b  a  a  b  c*
      7                    a* b  a  a  b  c
      8                       a  b* a  a  b  c
      9                          a  b  a  a  b  c

* First character of PATTERN at which mismatch occurs

Comparisons:
```

Step	TARGET	PATTERN	Match?
1	0	0	yes
	1	1	no
2	1	0	yes
	2	1	yes
	3	2	no
3	2	0	no
4	3	0	no
5	4	0	yes
	5	1	no
6	5	0	yes
	6	1	yes
	7	2	yes
	8	3	yes
	9	4	yes
	10	5	no
7	6	0	no
8	7	0	yes
	8	1	no
9	8	0	yes
	9	1	yes
	10	2	yes
	11	3	yes
	12	4	yes
	13	5	yes

Total Comparisons: 24

Figure 2: **Knuth-Morris-Pratt Algorithm**

```
BEGIN
Compute the NEXT table;
BEGTAR := 0;
J := 0;
WHILE BEGTAR(=N-M DO
        WHILE TARGET(BEGTAR+J)=PATTERN(J) DO
                J := J + 1;
                IF J)=M THEN GOTO PATTERN.FOUND
        ENDWHILE;
        IF NEXT(J)=-1 THEN
                BEGTAR := BEGTAR + J;
                J := 0
        ELSE
                BEGTAR := BEGTAR + (J - NEXT(J));
                J := NEXT(J)
        ENDIF
ENDWHILE;
PATTERN.NOT.FOUND:  RETURN 0;
PATTERN.FOUND:  RETURN BEGTAR;
END.
```

Table 2: **Example of Advanced Pattern-Matching Algorithm**

```
PATTERN:  a  b  a  a  b  c

          0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16

TARGET:   a  a  b  b  a  a  b  a  a  b  a  a  b  c  a  b  c

Step: 1  a  b* a  a  b  c
      2     a  b  a* a  b  c
      3              a  b* a  a  b  c
      4                 a  b  a  a  b  c*
      5                          a  b  a  a  b  c

Comparisons:
```

Step	TARGET	PATTERN	Match?
1	0	0	yes
	1	1	no
2	1	0	yes
	2	1	yes
	3	2	no
3	4	0	yes
	5	1	no
4	5	0	yes
	6	1	yes
	7	2	yes
	8	3	yes
	9	4	yes
	10	5	no
5	10	2	yes
	11	3	yes
	12	4	yes
	13	5	yes

Total Comparisons: 17

$02-$03, the starting address of the target in $04-$05, the ending address of the target in $06-$07 (all with low-order byte first), and the pattern length in $08. The main entry point is at the beginning of the program ($6000). The result, returned in $00-$01, will be the location where the pattern was found in the target, or 0000 if the pattern was not found. A secondary entry point is provided at $6020. This may be used to continue the search for subsequent occurrences of the pattern.

Analysis of the Elementary Algorithm

The elementary pattern-matching algorithm described above is straightforward and relatively easy to implement. Closer examination of table 1, however, suggests that improvements can be found.

For convenience, let us refer to the target and the pattern as arrays of characters whose subscripts begin with 0. The first element of the pattern is thus PATTERN(0), etc. At the completion of step 2 of table 1, we have just compared TARGET(3), the fourth element of the target string, with PATTERN(2) and found a mismatch. The elementary algorithm would next compare TARGET(2) with PATTERN(0). Earlier in step 2, however, we established that TARGET(2) is a "b" (because it matches PATTERN(1)) and cannot possibly match PATTERN(0), which is an "a". Step 3 of the elementary algorithm is therefore unnecessary.

Step 4 of the elementary algorithm compares TARGET(3) with PATTERN(0). Step 2, however, established that TARGET(3) didn't match PATTERN(2), which is an "a", and that it therefore cannot match PATTERN(0), which is also an "a". Step 4 is therefore also unnecessary. Similar analysis will show that steps 7 and 8 are also unnecessary.

The problem with the elementary pattern-matching algorithm can be characterized by stating that it does not make full use of the information available to it at any given time. What information is available to it at a given time? Suppose that we are now about to look at TARGET(I) and PATTERN(J) and that all previous characters of the pattern matched the corresponding target characters. We thus know PATTERN(0), PATTERN(1),... PATTERN (J-1). In addition, because of the fact that they match, we also know TARGET(I-J+1), TARGET (I-J+2),.., TARGET(I-1). Since all previous (before TARGET(I-J+1)) target characters have not resulted in a successful match, it is not necessary to keep track of them. It is therefore sufficient at any given time for the algorithm to have knowledge of the current characters in the target and pattern strings and the previously examined characters of the pattern string. It should never be necessary to go backwards in the target string!

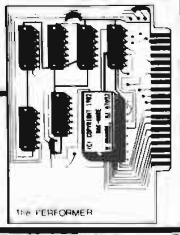
Advanced Pattern Matching

The algorithm demonstrated in table 2 corrects the deficiencies noted. It is called the Knuth-Morris-Pratt (KMP) algorithm, and was independently discovered by J. H. Morris in 1969 and by D.E. Knuth in 1970. V.R. Pratt, in collaboration with Knuth, refined the algorithm.

Recall that the elementary algorithm sometimes backs up to reconsider characters in the target that have already been considered. (For example, see step 7 of table 1.) It was Morris's objective to develop an algorithm that would eliminate the need to back up

HARDWARE

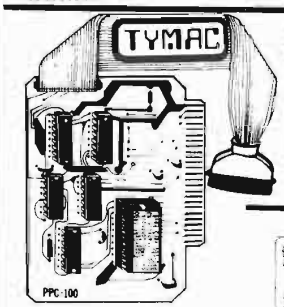
APPLE 201 - 839-3478
Dealer and Distributor Inquiries Invited.



THE PERFORMER PRINTER FORMATTER BOARD for Epson, OKI, NEC 8023, CITOH 8510 provides resident screen dump and print formatting in firmware. Plugs into Apple slot and easily accessed through PR# command — Use with standard printer cards. $49.00 specify printer.

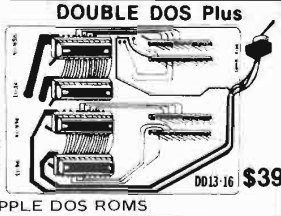
THE MIRROR FIRMWARE FOR NOVATION APPLE CAT II®
The Data Communication Handler ROM Emulates syntax of another popular Apple Modem product with improvements. Plugs directly on Apple CAT II Board. Supports Videx and Smarterm 80 column cards, touch tone and rotary dial, remote terminal, voice toggle, easy printer access and much more.
List $39.00 — Introductory Price $29.00

PARALLEL PRINTER CARD



A Universal Centronics type parallel printer board complete with cable and connector. This unique board allows you to turn on and off the high bit so that you can access additional features in many printers. Use with EPSON, C.ITOH, ANADEX, STARWRITER, NEC, OKI and other with standard Centronics configuration.
$139.00

DOUBLE DOS Plus



DOUBLE DOS Plus — a piggy-back board that plugs into the disk-controller card so that you can switch select between DOS 3.2 and DOS 3.3 $39
DOUBLE DOS Plus requires APPLE DOS ROMS

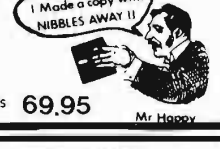
SOFTWARE

APPLE 48K

201 - 839-3478

NIBBLES AWAY II
AGAIN! Ahead of all others

- AUTO-LOAD PARAMETERS . . . Free's the user from having to Manually Key in Param values used with the more popular software packages available for the Apple II.
- EXPANDED USER MANUAL . . Incorporates new Tutorials for all levels of expertice; Beginners Flowchart for 'where do I begin' to 'Advanced Disk Analysis' is included.
- TRACK/SECTOR EDITOR . . . An all new Track/Sector Editor, including the following features: Read, Write, Insert, Delete Search, and impressive Print capabilities!
- DISK DIAGNOSTICS . . . Checks such things as: Drive Speed, Diskette Media Reliability, and Erasing Diskettes.
- HIGHEST RATED. . .Best back up Program in Softalk Poll (Rated 8.25 out of 10).
- CONTINUAL UPDATES . . . Available from Computer Applications and new listings on the source.



69.95
Mr Happy

Mr. Lister Customer Contact Profiler & Mailer
A Super Mail List Plus more up to 1000 Entries on single 3.3 Disk (only 1 Drive required) 2 second access time to any name full sort capabilities Dual Index Modes supports new 9 digit Zip. Easy to follow manual Not Copy Protected 4 user defined tables with 26 sort selections per table Beta tested for 6 months user defined label generation.
INTRODUCTORY PRICE ~~$139~~. $99.00 Dealer & Dist. Inquiries Invited.

Super PIX HIRES SCREEN DUMP
The Software package that will allow your printer to dump page 1 or page 2 of the Apple Hires screen horizontally or vertically. Use with EPSON® MX-80 with or without GRAFTRAX® Roms, MX-70 — OKI® Microline 80, 82, 83, 82A, 83A — C. ITOH® 8510 and NEC 8023A. Requires Tymac Parallel Printer Board PPC-100. . . . $24.95

THE APPLE CARD—Two sided 100% plastic reference card Loaded with information of interest to all Apple owners $3.98

MasterCard VISA

MICRO-WARE DIST. INC. P.O. BOX 113 POMPTON PLAINS, N.J. 07444
Dealer and Distributor Inquiries Invited.

the target string. This would allow him to consider characters of the target string one at a time as they were received from a file, thus eliminating the need to provide buffer space for the target file. (Notice that the KMP algorithm achieves this goal.) Considerations of efficiency were otherwise secondary. Pratt discovered that the running time of the KMP algorithm was proportional to M + N, where M is the pattern length and N is the target length. The running time of the elementary algorithm required 24 comparisons and the KMP algorithm 17 comparisons.

Elementary vs. Advanced Pattern Matching

The Knuth-Morris-Pratt algorithm is certainly more sophisticated than the elementary algorithm, but is it any better in practice? How each algorithm performs in a particular instance depends most importantly on the number of partial matches that will be encountered before the final match is found. The KMP algorithm is at its best, and the elementary algorithm at its worst, when such partial matches are numerous. When there are few partial matches, the sequence of comparisons generated by each algorithm is about the same. In such cases, however, the extra complexity required to implement the KMP algorithm incurs a penalty in speed.

It is my conclusion that for most microcomputer applications the elementary algorithm is preferable to the advanced algorithm. Not only is it simpler and more compact, it is often faster as well. A test was conducted to compare execution speed. A four-character pattern was imbedded near the end of a 32K target text. It took the elementary algorithm 1.6 seconds to find it; the KMP algorithm took 2.6 seconds.

The KMP algorithm is not the only advanced pattern-matching algorithm around. A new algorithm by Boyer and Moore shows promise for cases in which there are few partial matches. The algorithm is unfortunately even more complex in its implementation than the KMP algorithm, requiring not one but two precomputed tables. It thus seems likely that microcomputer users would do well to stay with the elementary algorithm.

References

1. Knuth, Donald E., Morris, James H. Jr., Pratt, Vaughn R., "Fast Pattern Matching in Strings," *SIAM Journal of Computing*, Vol.6, No.2, June 1977.

2. Boyer, R.S., and J.S. Moore, "A Fast String Searching Algorithm," *Communications of the ACM*, 20 (No.10), November 1977.

Charles F. Taylor may be reached at 587F Sampson Lane, Monterey, California 93940.

INTRODUCING!

System-68

For you, the 68XX user! An exciting new monthly magazine formatted for the 68XX enthusiast specializing in hardware applications.

REGULAR FEATURES WILL INCLUDE:
- Articles on 6800, 6809, and 68000
- Construction Articles
- New Product Announcements
- Product Reviews
- Classified Ads
- "2-Bits"
- Questions & Answers

ADVERTISERS WELCOME!
CASH COMPENSATION FOR ARTICLES
YEARLY SUBSCRIPTIONS AVAILABLE
1 Year $24.00 2 Years $45.00
Single Issue $2.95
CALL NOW FOR YOUR SUBSCRIPTION!

P. O. BOX 310
CONYERS, GEORGIA 30207
404-929-0606
MASTER CHARGE, VISA, and AMERICAN EXPRESS ACCEPTED

SOUTHEASTERN MICRO SYSTEMS, INC.
1080 IRIS DRIVE
CONYERS, GEORGIA 30207
404-922-1620

SX-9
Single Board System

- Uses 6809 CPU
- Two RS232 Serial Ports
- Two Parallel 8 Bit Ports
- One High-Speed Serial Port
- System Clock at 1.22 MHZ, Fixed
- System provides for selectable diagnostics on power up or by command
- Provides for automatic disk boot on power up
- Provides for board ID for future multi-user configuration or user defined
- Expansion capabilities via ribbon cable
- Disk controller provides control for up to four 5 1/4" drives from SS/SD up to DS/DD control
- Real time clock
- Memory 64K RAM (62K User Memory)
- All signals via plug in ribbon connectors
- Compatible with TSC FLEX 9 and all TSC 6809 single user software. DS/DD requires SEMS-1 Disk Drivers
- Board size: 10.25" by 10.5"
- Power requirements: +5 VDC at 3 AMPS, +12 VDC at 250 ma, -12 VDC at 100 ma
- Optional Cabinet with Power Supply

LIMITED OFFER	PRICES	
Bare Board With SEMS-1 Monitor & Documentation	ASSEMBLED & TESTED	$995.00
$250.00	BOARD WITH CABINET & POWER SUPPLY	$1295.00
ONLY 100 Bare Boards Will Be Sold! ORDER NOW! First order basis	SOFTWARE: TSC FLEX 9 With SEMS Disk Drivers (with Editor & Assembler)	$200.00

US SHIPPING $10.00, FOREIGN SHIPPING $50.00

Listing 1

```
                        0010 ;ELEMENTARY PATTERN MATCHING
                        0020 ;FOR THE 6502
                        0030 ;
                        0040 ; BY C. F. TAYLOR, JR.
                        0050 ; 19 MAY 1981
                        0060 ;
                        0070 ;
                        0080 ; PATTERN:  UP TO 255 CHARS
                        0090 ;           PUT ADDR IN $02-$03 (LO-HI)
                        0100 ;           PUT LENGTH OF PATTERN IN $08
                        0110 ; TARGET:   ANY LENGTH (IN RAM)
                        0120 ;           PUT START ADDR IN $04-$05
                        0130 ;           PUT END ADDR IN $06-$07
                        0140 ; RESULT:   $00-$01 WILL CONTAIN LOC
                        0150 ;           WHERE PATTERN FOUND OR
                        0160 ;           0000 IF PATTERN NOT FOUND
                        0170 ;
                        0180            .OS
                        0190 ;
                        0200 ; PAGE ZERO EQUATES
                        0210 ;
                        0220 LOC        .DE $00         ;RESULT LEFT HERE
                        0230 PADDR      .DE $02         ;PTR TO 1ST BYTE OF PATTERN
                        0240 TADDR      .DE $04         ;PTR TO 1ST BYTE OF TARGET
                                                         STRING
                        0250 TRGTEND    .DE $06         ;PTR TO LAST BYTE OF TARGET
                        0260 LENGTH     .DE $08         ;LENGTH OF PATTERN
                        0270 ;
                        0280            .BA $09
0009-                   0290 EOS        .DS 1           ;END-OF-STRING CHAR
000A-                   0300 BEGTAR     .DS 2           ;POINTER TO TARGET STRING
000C-                   0310 TEMP       .DS 2
                        0320 ;
                        0330            .BA $6000
                        0340            ;
                        0350            ;
                        0360            ;INITIALIZE POINTER TO TARGET STRING
                        0370            ;
6000- A5 04             0380 MATCH      LDA *TADDR
6002- 85 0A             0390            STA *BEGTAR
6004- A5 05             0400            LDA *TADDR+1
6006- 85 0B             0410            STA *BEGTAR+1
                        0420            ;
6008- A9 FF             0430            LDA #$FF            ;EOS CHAR
600A- 85 09             0440            STA *EOS
600C- A4 08             0450 L1         LDY *LENGTH         ;BE SURE EOS CHAR
600E- 88                0460 L2         DEY                 ;IS NOT IN PATTERN
600F- B1 02             0470            LDA (PADDR),Y
6011- C5 09             0480            CMP *EOS
6013- D0 05             0490            BNE L3
                        0500            ;
                        0510            ;EOS CHAR IN PATTERN, SO TRY ANOTHER
                        0520            ;
6015- C6 09             0530            DEC *EOS
6017- 18                0540            CLC
6018- 90 F2             0550            BCC L1
                        0560            ;
601A- 98                0570 L3         TYA                 ;LOOP UNTIL ALL
601B- D0 F1             0580            BNE L2              ;CHARS CHECKED
601D- 18                0590            CLC
601E- 90 06             0600            BCC MATCH2
                        0610            ;
                        0620 CONTINUE   ;
                        0630            ;ALTERNATE ENTRY
                        0640            ;CONTINUE SEARCH AT BEGTAR+1
                        0650            ;
6020- E6 0A             0660            INC *BEGTAR
6022- D0 02             0670            BNE MATCH2
6024- E6 0B             0680            INC *BEGTAR+1
6026- A0 01             0690 MATCH2     LDY #1
6028- B1 06             0700            LDA (TRGTEND),Y
602A- 48                0710            PHA                 ;SAVE ORIG VALUE
602B- A5 09             0720            LDA *EOS            ;SENTINEL VALUE
602D- 91 06             0730            STA (TRGTEND),Y     ;SENTINEL FOR TARGET
602F- A4 08             0740            LDY *LENGTH
6031- B1 02             0750            LDA (PADDR),Y
6033- 48                0760            PHA                 ;SAVE
6034- A5 09             0770            LDA *EOS
6036- 91 02             0780            STA (PADDR),Y       ;SENTINEL FOR PATTERN
                        0790 ;
                        0800 ;MAIN LOOP
                        0810 ;          TEST FOR END OF TARGET
6038- A4 08             0820 WHIL1      LDY *LENGTH
603A- B1 0A             0830            LDA (BEGTAR),Y
```

(Continued)

Listing 1 *(Continued)*

```
603C- C5 09             0840            CMP *EOS            ;SENTINEL REACHED?
603E- D0 17             0850            BNE OK
                        0860            ;
                        0870            ;IF EOS FOUND IN TARGET,
                        0880            ;MAKE SURE PAST TRGTEND
                        0890            ;
6040- 18                0900            CLC
6041- 98                0910            TYA                 ;TEMP:=BEGTAR+Y
6042- 65 0A             0920            ADC *BEGTAR
6044- 85 0C             0930            STA *TEMP
6046- A5 0B             0940            LDA *BEGTAR+1
6048- 69 00             0950            ADC #0              ;ADD IN CARRY
604A- 85 0D             0960            STA *TEMP+1
604C- 38                0970            SEC
604D- A5 06             0980            LDA *TRGTEND        ;TRGTEND-TEMP
604F- E5 0C             0990            SBC *TEMP
6051- A5 07             1000            LDA *TRGTEND+1
6053- E5 0D             1010            SBC *TEMP+1
6055- 90 23             1020            BCC ENDWH1          ;IF PAST TRGTEND
6057- A0 00             1030 OK         LDY #0              ;NOT PAST TRGTEND
6059- B1 02             1040            LDA (PADDR),Y
                        1050            ;
                        1060            ;CHECK FOR MATCH
                        1070            ;
605B- D1 0A             1080 WHIL2      CMP (BEGTAR),Y      ;INNER LOOP
605D- D0 12             1090            BNE ENDWH2
605F- C8                1100            INY
6060- B1 02             1110            LDA (PADDR),Y
6062- C5 09             1120            CMP *EOS
6064- D0 F5             1130            BNE WHIL2
                        1140            ;
                        1150            ;PATTERN FOUND
                        1160            ;
6066- A5 0A             1170            LDA *BEGTAR
6068- 85 00             1180            STA *LOC
606A- A5 0B             1190            LDA *BEGTAR+1
606C- 85 01             1200            STA *LOC+1
606E- 18                1210            CLC
606F- 90 0F             1220            BCC EXIT
                        1230            ;
6071- E6 0A             1240 ENDWH2     INC *BEGTAR         ;ADVANCE POINTER
6073- D0 C3             1250            BNE WHIL1
6075- E6 0B             1260            INC *BEGTAR+1
6077- 18                1270            CLC
6078- 90 BE             1280            BCC WHIL1
                        1290            ;
                        1300 ENDWH1     ;PATTERN NOT FOUND
                        1310            ;
607A- A9 00             1320            LDA #0
607C- 85 00             1330            STA *LOC
607E- 85 01             1340            STA *LOC+1
                        1350            ;
6080- A4 08             1360 EXIT       LDY *LENGTH         ;REPLACE SENTINELS
6082- 68                1370            PLA
6083- 91 02             1380            STA (PADDR),Y
6085- A0 01             1390            LDY #1
6087- 68                1400            PLA
6088- 91 06             1410            STA (TRGTEND),Y
608A- 60                1420            RTS
                        1430            .EN
```

Listing 2

```
                        0010 ;ADVANCED PATTERN MATCHING
                        0020 ;KNUTH-MORRIS-PRATT ALGORITHM
                        0030 ;
                        0040 ; 6502 VERSION BY C. F. TAYLOR, JR.
                        0050 ; 20 MAY 1981
                        0060 ;
                        0070 ;
                        0080 ; PATTERN:  UP TO 255 CHARS
                        0090 ;           PUT ADDR IN $02-$03 (LO-HI)
                        0100 ;           PUT LENGTH OF PATTERN IN $08
                        0110 ; TARGET:   ANY LENGTH (IN RAM)
                        0120 ;           PUT START ADDR IN $04-$05
                        0130 ;           PUT END ADDR IN $06-$07
                        0140 ; RESULT:   $00-$01 WILL CONTAIN LOC
                        0150 ;           WHERE PATTERN FOUND OR
                        0160 ;           0000 IF PATTERN NOT FOUND
                        0170 ;
                        0180            .OS
                        0190 ;
                        0200 ; PAGE ZERO EQUATES
```

(Continued)

Listing 2 *(continued)*

```
                        0210 ;
                        0220 LOC        .DE $00     ;RESULT LEFT HERE
                        0230 PADDR      .DE $02     ;PTR TO 1ST BYTE OF PATTERN
                        0240 TADDR      .DE $04     ;PTR TO 1ST BYTE OF TARGET STRING
                        0250 TRGTEND    .DE $06     ;PTR TO LAST BYTE OF TARGET
                        0260 LENGTH     .DE $08     ;LENGTH OF PATTERN
                        0270 ;
                        0280            .BA $09
0009-                   0290 EOS        .DS 1       ;END-OF-STRING CHAR
000A-                   0300 BEGTAR     .DS 2       ;POINTER TO TARGET STRING
000C-                   0310 TEMP       .DS 2
000E-                   0320 M          .DS 1       ;LENGTH-1
000F-                   0330 T          .DS 1       ;POINTER
0010-                   0340 J          .DS 1       ;POINTER
                        0350 ;
                        0360            .BA $6000
                        0370            ;
6000- A6 08             0380 SETUP      LDX *LENGTH ;COMPUTE NEXT TABLE
6002- CA                0390            DEX
6003- 86 0E             0400            STX *M      ;LENGTH-1
6005- A2 FF             0410            LDX #$FF
6007- 86 0F             0420            STX *T
6009- A0 00             0430            LDY #0
600B- 84 10             0440            STY *J
600D- 8A                0450            TXA
600E- 99 0B 61          0460            STA NEXT,Y          ;NEXT(0):=-1
6011- C4 0E             0470 WHIL3      CPY *M              ;FINISHED?
6013- B0 39             0480            BCS ENDWHIL3        ;J(M
6015- A6 0F             0490            LDX *T
6017- 30 12             0500            BMI ENDWHIL4        ;T(=0
6019- A4 10             0510 WHIL4      LDY *J
601B- B1 02             0520            LDA (PADDR),Y       ;PATTERN(J)
601D- A4 0F             0530            LDY *T
601F- D1 02             0540            CMP (PADDR),Y       ;=PATTERN(T)?
6021- F0 08             0550            BEQ ENDWHIL4
6023- BD 0B 61          0560            LDA NEXT,X
6026- AA                0570            TAX
6027- 86 0F             0580            STX *T              ;T:=NEXT(T)
6029- 10 EE             0590            BPL WHIL4
602B- E8                0600 ENDWHIL4   INX
602C- 86 0F             0610            STX *T              ;T:=T+1
602E- E6 10             0620            INC *J              ;J:=J+1
6030- A4 10             0630            LDY *J
6032- B1 02             0640            LDA (PADDR),Y       ;PATTERN(J)
6034- A4 0F             0650            LDY *T
6036- D1 02             0660            CMP (PADDR),Y       ;=PATTERN(T)?
6038- D0 0B             0670            BNE L5
603A- B9 0B 61          0680            LDA NEXT,Y
603D- A4 10             0690            LDY *J
603F- 99 0B 61          0700            STA NEXT,Y          ;NEXT(J):=NEXT(T)
6042- 18                0710            CLC
6043- 90 CC             0720            BCC WHIL3
6045- A4 10             0730 L5         LDY *J
6047- 8A                0740            TXA
6048- 99 0B 61          0750            STA NEXT,Y          ;NEXT(J):=T
604B- 18                0760            CLC
604C- 90 C3             0770            BCC WHIL3
                        0780 ENDWHIL3
                        0790            ;
                        0800            ;INITIALIZE POINTER TO TARGET STRING
                        0810            ;
604E- A5 04             0820 MATCH      LDA *TADDR
6050- 85 0A             0830            STA *BEGTAR
6052- A5 05             0840            LDA *TADDR+1
6054- 85 0B             0850            STA *BEGTAR+1
                        0860            ;
6056- A9 FF             0870            LDA #$FF            ;EOS CHAR
6058- 85 09             0880            STA *EOS
605A- A4 08             0890 L1         LDY *LENGTH         ;BE SURE EOS CHAR
605C- 88                0900 L2         DEY                 ;IS NOT IN PATTERN
605D- B1 02             0910            LDA (PADDR),Y
605F- C5 09             0920            CMP *EOS
6061- D0 05             0930            BNE L3
                        0940            ;
                        0950            ;EOS CHAR IN PATTERN, SO TRY ANOTHER
                        0960            ;
6063- C6 09             0970            DEC *EOS
6065- 18                0980            CLC
6066- 90 F2             0990            BCC L1
                        1000            ;
6068- 98                1010 L3         TYA                 ;LOOP UNTIL ALL
6069- D0 F1             1020            BNE L2              ;CHARS CHECKED
606B- 18                1030            CLC
606C- 90 06             1040            BCC MATCH2
```

(continued)

Listing 2 *(continued)*

```
                        1050            ;
                        1060 CONTINUE   ;
                        1070            ;ALTERNATE ENTRY
                        1080            ;CONTINUE SEARCH AT BEGTAR+1
                        1090            ;
606E- E6 0A             1100            INC *BEGTAR
6070- D0 02             1110            BNE MATCH2
6072- E6 0B             1120            INC *BEGTAR+1
                        1130            ;
6074- A0 01             1140 MATCH2     LDY #1
6076- B1 06             1150            LDA (TRGTEND),Y
6078- 48                1160            PHA                 ;SAVE ORIG VALUE
6079- A5 09             1170            LDA *EOS            ;SENTINEL VALUE
607B- 91 06             1180            STA (TRGTEND),Y     ;SENTINEL FOR TARGET
607D- A4 08             1190            LDY *LENGTH
607F- B1 02             1200            LDA (PADDR),Y
6081- 48                1210            PHA                 ;SAVE
6082- A5 09             1220            LDA *EOS
6084- 91 02             1230            STA (PADDR),Y       ;SENTINEL FOR PATTERN
                        1240 ;
                        1250 ;MAIN LOOP
                        1260 ;
6086- A0 00             1270            LDY #0
6088- 84 10             1280            STY *J              ;INITIALIZE POINTER
                        1290            ;
                        1300            ;END OF TARGET?
                        1310            ;
608A- A4 08             1320 WHIL1      LDY *LENGTH
608C- B1 0A             1330            LDA (BEGTAR),Y
608E- C5 09             1340            CMP *EOS            ;SENTINEL REACHED?
6090- D0 17             1350            BNE OK
                        1360            ;
                        1370            ;IF EOS FOUND IN TARGET,
                        1380            ;MAKE SURE PAST TRGTEND
                        1390            ;
6092- 18                1400            CLC
6093- 98                1410            TYA                 ;TEMP:=BEGTAR+Y
6094- 65 0A             1420            ADC *BEGTAR
6096- 85 0C             1430            STA *TEMP
6098- A5 0B             1440            LDA *BEGTAR+1
609A- 69 00             1450            ADC #0              ;ADD IN CARRY
609C- 85 0D             1460            STA *TEMP+1
609E- 38                1470            SEC
609F- A5 06             1480            LDA *TRGTEND        ;TRGTEND-TEMP
60A1- E5 0C             1490            SBC *TEMP
60A3- A5 07             1500            LDA *TRGTEND+1
60A5- E5 0D             1510            SBC *TEMP+1
60A7- 90 51             1520            BCC ENDWH1          ;IF PAST TRGTEND
60A9- A4 10             1530 OK         LDY *J              ;NOT PAST TRGTEND
60AB- B1 02             1540            LDA (PADDR),Y
                        1550            ;
                        1560            ;CHECK FOR MATCH
                        1570            ;
60AD- D1 0A             1580 WHIL2      CMP (BEGTAR),Y      ;INNER LOOP
60AF- D0 12             1590            BNE ENDWH2
60B1- C8                1600            INY
60B2- B1 02             1610            LDA (PADDR),Y
60B4- C5 09             1620            CMP *EOS
60B6- D0 F5             1630            BNE WHIL2
                        1640            ;
                        1650            ;PATTERN FOUND
                        1660            ;
60B8- A5 0A             1670            LDA *BEGTAR
60BA- 85 00             1680            STA *LOC
60BC- A5 0B             1690            LDA *BEGTAR+1
60BE- 85 01             1700            STA *LOC+1
60C0- 18                1710            CLC
60C1- 90 3D             1720            BCC EXIT
                        1730            ;
                        1740 ENDWH2     ;COMPUTE WHICH PATTERN CHAR TO EXAMINE NEXT
                        1750            ;
60C3- B9 0B 61          1760            LDA NEXT,Y
60C6- C9 FF             1770            CMP #$FF            ;SPECIAL CASE
60C8- F0 18             1780            BEQ L6
60CA- 98                1790            TYA                 ;BEGTAR:=
60CB- 38                1800            SEC                 ;BEGTAR+(Y-NEXT(Y))
60CC- F9 0B 61          1810            SBC NEXT,Y
60CF- 18                1820            CLC
60D0- 65 0A             1830            ADC *BEGTAR
60D2- 85 0A             1840            STA *BEGTAR
60D4- A5 0B             1850            LDA *BEGTAR+1
60D6- 69 00             1860            ADC #0
60D8- 85 0B             1870            STA *BEGTAR+1
60DA- B9 0B 61          1880            LDA NEXT,Y
```

(continued)

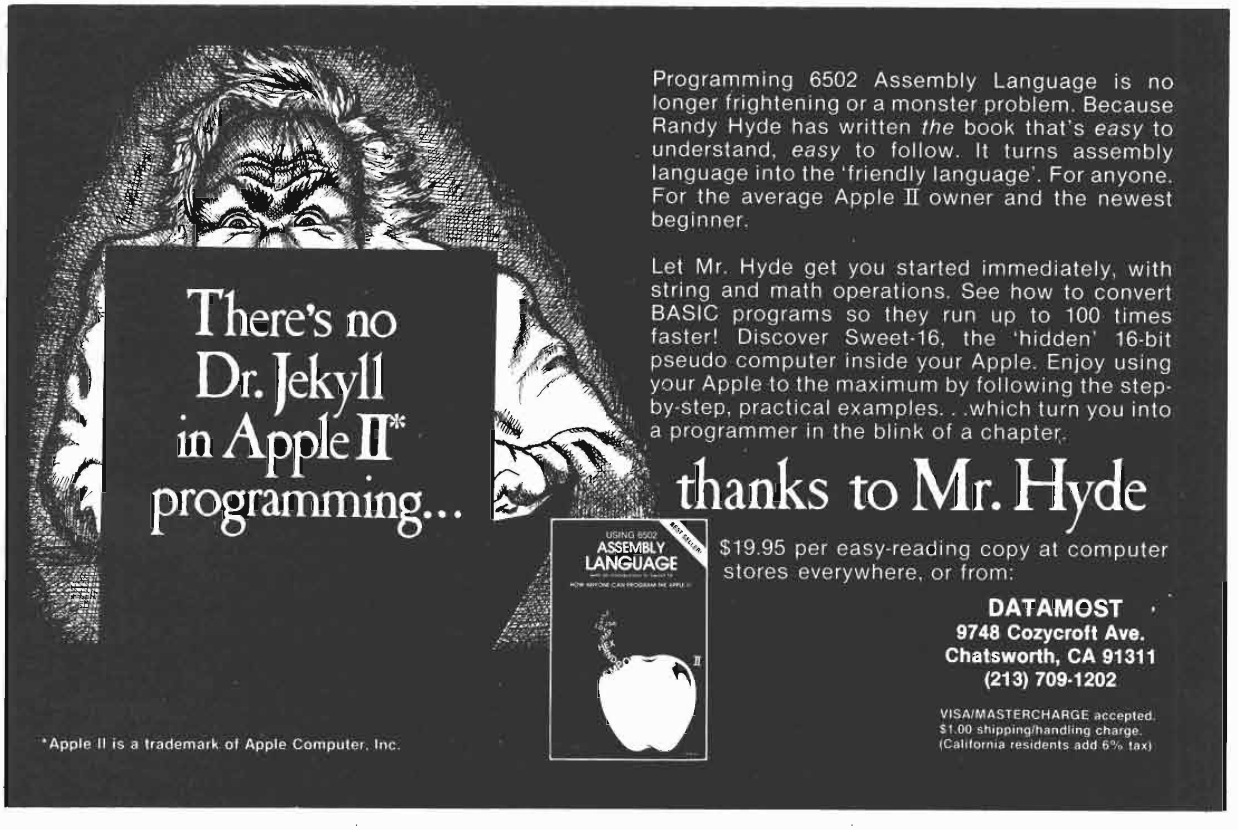
Listing 2 *(continued)*

```
60DD- 85 10   1890          STA *J
60DF- 18      1900          CLC
60E0- 90 A8   1910          BCC WHIL1
60E2- C8      1920 L6       INY               ;GET NEW TARGET CHAR
60E3- 98      1930          TYA               ;BEGTAR:=
60E4- 18      1940          CLC               ;BEGTAR+Y
60E5- 65 0A   1950          ADC *BEGTAR
60E7- 85 0A   1960          STA *BEGTAR
60E9- A5 0B   1970          LDA *BEGTAR+1
60EB- 69 00   1980          ADC #0
60ED- 85 0B   1990          STA *BEGTAR+1
60EF- A0 00   2000          LDY #0
60F1- 84 10   2010          STY *J
60F3- F0 95   2020          BEQ WHIL1         ;ALWAYS TAKEN
60F5- E6 0B   2030          INC *BEGTAR+1
60F7- 18      2040          CLC
60F8- 90 90   2050          BCC WHIL1
              2060          ;
              2070 ENDWH1   ;PATTERN NOT FOUND
              2080          ;
60FA- A9 00   2090          LDA #0
60FC- 85 00   2100          STA *LOC
60FE- 85 01   2110          STA *LOC+1
              2120          ;
6100- A4 08   2130 EXIT     LDY *LENGTH       ;REPLACE SENTINELS
6102- 68      2140          PLA
6103- 91 02   2150          STA (PADDR),Y
6105- A0 01   2160          LDY #1
6107- 68      2170          PLA
6108- 91 06   2180          STA (TRGTEND),Y
610A- 60      2190          RTS
610B-         2200 NEXT     .DS 256           ;ROOM FOR TABLE
              2210          .EN
```

Figure 3: **Construction of NEXT Table**

```
BEGIN
T := -1;
J := 0;
NEXT(0) := -1;
WHILE J<M DO
      WHILE T>0 AND PATTERN(J)<>PATTERN(T) DO
            T := NEXT(T)
      ENDWHILE;
      T := T + 1;
      J := J + 1;
      IF PATTERN(J)=PATTERN(T) THEN
            NEXT(J) := NEXT(T)
      ELSE
            NEXT(J) := T
      ENDIF
ENDWHILE;
END.
```

MICRO™



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

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

ARK COMPUTING

Your Salvation In The Sea Of Inflation.

P.O. Box 2025 ● Corona, CA ● 91720

Computer Stop 16K Ram Board $79.95

This high quality 16K Ram board acts like a Language Card when plugged into slot 0 of your APPLE II. Compatible with Basic, DOS 3.3, CP/M, PASCAL, LISA 2.5, and VISICALC. With this card you get high quality, low price, compatibility, and a 1 Year Manufacturer Warranty all for $79.95.

LAZER Lower Case +Plus III $34.95

The best lower case adapter available for the APPLE II. This feature packed board has twice the features of competing boards. The Lazer Lower Case +Plus III is the only lower case adapter that works with VISICALC and is recommended by Stoneware for DB MASTER. The Lower Case Plus III is expandable to 4 character sets (2 on board), has inverse lower case, includes ASCII and Graphics, and is compatible with most word processors. NOTE: For REV. 6 and earlier order Lower Case +Plus at $44.95.

LAZER Keyboard +Plus $69.95

The Lazer Microsystems Keyboard +Plus has a 64 character type ahead buffer. The buffer can be cleared or disabled. The Keyboard +Plus lets you use the shift-key as a typewriter shift-key, allowing you to enter directly the 128 ASCII character set from the APPLE keyboard. The Keyboard +Plus may be installed on any APPLE II.

Computer Stop Omnivision $129.95

Looking for an 80-column card? Look no further! Now all your Basic, CP/M, and Pascal programs can take advantage of a full 80-column display.

Wizard-BPO
(Buffered Printer Interface)
$139.95

This outstanding parallel interface card comes with 16K of Ram installed, expandable to 32K This card also has Graftrax-like features. So now you no longer need to wait to use your APPLE II while your printer is busy. Compatible with Basic, CP/M, Pascal.

Microtek Parallel Printer Interface $59.95

This popular printer interface card, manufactured by Microtek, is a steal at $59.95. The printer card comes complete with a cable and a Centronics compatible connector (Amphenol). Works with basic, CP/M, and Pascal. This card also has graphics capabilities.

ORDERING INFORMATION

We accept: VISA/MASTERCARD (include card #, expiration date, and signature), Cashier or Certified Checks, Money Orders, or Personnal Checks (please allow 10 business days to clear). We also accept COD's (please include $2.00 COD charge).

Please add 3% for shipping and handling (minimum $2.00). Foreign orders please add 10% for shipping and handling (minimum $10.00).

California residents add 6% sales tax. All equipment is subject to price change and availability without notice. All equipment is new and complete with manufacturer's warranty.

(714) **735-2250**

Random Number Generator in Machine Language for the Apple

by Arthur Matheny

This simple subroutine can easily be implemented in a machine-language program whenever random numbers are needed. Two examples are provided.

Random Number Generator
requires:

Apple II or Apple II Plus

A random-number generator is needed in many applications — games, simulations, Monte Carlo methods, etc. The BASIC interpreter in my Apple II has a pseudo-random-number generator built in, but what about machine-language programs? I need to find the random-number routine in the BASIC ROM, but my manual does not tell me where it is located!

I decided to make a trip to the library, where I found several shelves of books about random numbers. I selected one: *The Generation of Random Variates*, written by T.G. Newman and P.L. Odell. Chapter 2 tells how to design a random-number generator, and chapter 9 tells how to test it.

The program I wrote passed all three tests that I put to it: the frequency test, the run test, and the serial test. In comparison, the random-number generator used by Apple's Integer BASIC passed the frequency test and the run test, but not the serial test. Failure to pass the serial test indicates that sequential pairs are not perfectly uncorrelated. That does not mean that the routine used by BASIC is no good; it means that mine is even better.

The execution speed can be a big factor in some applications. In the form presented here, the routine runs in 68 cycles or occasionally a few more. You can speed it up considerably by eliminating the loops and counters, thus making it into a straight-through calculation with absolute addressing.

The random number generator is given in listing 1. Before this routine is called for the first time, the main program should load seed values into memory locations $10, $11, $12, and $13. This is a pseudo-random-number generator, which means that the random sequence is predetermined by the values used to initialize these four locations.

There are two general methods to pick the seed values. At execution time you can input a number that your program uses to load the four bytes above. [The specific method that it would use to do this is not important.] If you enter the number that you executed yesterday, then you will get the results that you did yesterday. This may or may not be desirable, depending on what the program does.

The second method is to generate the seed values in some unpredictable way. For example, if you have a real-time clock, you could use its reading as a random-number seed. I do not have a real-time clock, but on the Apple, the values in memory locations $4E and $4F roll around and around whenever the computer is waiting for a keystroke from the keyboard. These make excellent seed values. All I have to do is move $4E to $10 and $12, and move $4F to $11 and $13. Whatever method

Listing 1

```
                ;****************************
                ;*                          *
                ;*  6502 MACHINE LANGUAGE   *
                ;* RANDOM NUMBER GENERATOR  *
                ;*                          *
                ;*      BY ART MATHENY      *
                ;*                          *
                ;****************************
                ;
                ; SUBROUTINE FOR USE WITH ML PROGRAMS
                ; USAGE: JSR $800
                ;
                ARRAY  EQU $10
                ;
                ; ARRAY IS A 4-BYTE INTEGER.
                ; MOST SIGNIFICANT BYTE IS AT LOCATION $10.
                ; LEAST SIGNIFICANT BYTE IS AT LOCATION $13.
                ; LOCATIONS $10 THROUGH $13 SHOULD BE SEEDED BEFORE
                ;     THE SUBROUTINE IS FIRST CALLED.
                ; RESULT APPEARS IN LOCATION $10.
                ;
0800                    ORG $800
0800 18         RND     CLC
0801 A203               LDX #3          ; NUMBER OF BYTES IN ARRAY - 1
0803 B510               LDA ARRAY,X
0805 CA                 DEX
0806 7510       LOOP1   ADC ARRAY,X
0808 9510               STA ARRAY,X
080A CA                 DEX
080B 10F9               BPL LOOP1
080D A203               LDX #3          ; ADD 1 TO ARRAY
080F F610       LOOP2   INC ARRAY,X
0811 D003               BNE RTS1
0813 CA                 DEX
0814 10F9               BPL LOOP2
0816 60         RTS1    RTS
                        END
```

you use, you only need to seed these memory locations once. When that is done, you can call the random-number generator as many times as you like.

To call the routine from a machine-language program, use JSR $800. This randomizes memory location $10. Its new value is virtually uncorrelated with its previous value. Memory location $11 is also randomized, but to a lesser degree. To call the routine from a BASIC program, use CALL 768, and then retrieve the random value by PEEK(16).

The "period" of this random-number generator is rather long. The sequence of numbers in memory location $10 will not begin to repeat itself until after 2^{32} numbers have been generated. If you expect to generate more numbers than that, change lines $801 and $80D to LDX #$4 and include memory location $14 in the initialization process.

The program in listing 2 is an example of an application of the random-number generator. It only works on an Apple II or Apple II Plus, and produces a visual effect that I call "Globular Cluster." Enter the monitor and input the program beginning at $800. To execute it, type 800G in the monitor, and use the RESET key to stop it.

I have included this program to make a point: the pattern of stars will eventually repeat, but according to a crude calculation, I estimate that this will not happen until about four years from now.

Reference

1. T.G. Newman and P.L. Odell, *The Generation of Random Variates*, Hafner Publishing Company, New York, 1971.

Art Matheny is director of computer-assisted instruction in the Biology Department at the University of South Florida. He has an MS degree in physics from Purdue University and taught physics for several years before recently turning to full-time programming. At home he has a personal computer that he uses for writing games and learning assembly language. He lives at 1405 Four Seasons Blvd., Lutz, Florida 33549.

Listing 2

```
                ;*********************
                ;*                   *
                ;* GLOBULAR CLUSTER  *
                ;*   FOR APPLE IC    *
                ;*                   *
                ;*  BY ART MATHENY   *
                ;*                   *
                ;*********************
                ;
                ; TO EXECUTE FROM MONITOR: 800G
                ;
                QUOT   EPZ $4          ; QUOTIENT
                REM    EPZ $5          ; REMAINDER
                DVDEND EPZ $6          ; DIVIDEND
                DVSER  EPZ $7          ; DIVISOR
                SCRNX  EPZ $8          ; X-LOCATION
                KNTR   EPZ $A          ; STAR COUNTER
                DENS   EPZ $B          ; NUMBER OF STARS IN VIEW
                SPEED  EPZ $C          ; SPEED OF TRAVEL
                POINT  EPZ $D
                TUBE   EPZ $E
                RANBYT EPZ $10         ; RANDOM BYTE
                HORIZL EPZ $18         ; HORIZONTAL COORD TABLE
                HORIZH EPZ $19
                VERTL  EPZ $1A         ; VERTICAL COORD TABLE
                VERTH  EPZ $1B
                BASEL  EPZ $1C         ; DEPTH COORD TABLE
                BASEH  EPZ $1D
                TINTL  EPZ $1E         ; STAR COLOR TABLE
                TINTH  EPZ $1F
                COLOR  EPZ $30         ; COLOR FOR PLOT ROUTINE
                RNDL   EPZ $4E         ; APPLE'S RANDOM NUMBER SEED
                RNDH   EPZ $4F
                ;
                ; ROUTINES IN APPLE'S ROM
                ;
                PLOT   EQU $F800       ; LO-RES PLOT ROUTINE
                CLRSCR EQU $F832       ; CLEARS LO-RES SCREEN
                ;
                ; * * *  INITIALIZATION  * * *
                ;
0800                    ORG $800
0800 A90C               LDA #C         ; SETS NUMBER OF STARS IN VIEW
0802 850B               STA DENS
0804 A903               LDA #3         ; SETS WARP FACTOR
0806 850C               STA SPEED
0808 A90D               LDA #D
080A 850D               STA POINT
080C A9FE               LDA #FE
080E 850E               STA TUBE
0810 AD56C0             LDA $C056      ; SET LOW-RES MODE
0813 AD54C0             LDA $C054      ; DISPLAY PAGE 1
0816 AD52C0             LDA $C052      ; ALL GRAPHICS
0819 2032F8             JSR CLRSCR     ; CLEAR SCREEN
081C AD50C0             LDA $C050      ; SET GRAPHICS MODE
081F A54E               LDA RNDL       ; TRANSFER APPLE'S RANDOM SEED
0821 8510               STA $10        ; TO LOCATION $10 THRU $13
0823 8512               STA $12
0825 A54F               LDA RNDH
0827 8511               STA $11
0829 8513               STA $13
082B A90C               LDA #C         ; LOAD BASE ADDRESSES OF
                                         THE 4 TABLES
082D 8519               STA HORIZH
082F A90D               LDA #D
0831 851B               STA VERTH
0833 A90E               LDA #E
0835 851D               STA BASEH
0837 A90F               LDA #F
0839 851F               STA TINTH
083B A900               LDA #0
083D 8518               STA HORIZL
083F 851A               STA VERTL
0841 851C               STA BASEL
0843 851E               STA TINTL
0845 A40B               LDY DENS       ; CHOOSE INITIAL STAR COORDINATES
0847 20F008 NEXT        JSR NEW1
084A 88                 DEY
084B D0FA               BNE NEXT
084D                    ;
084D                    ; * * *  MAIN LOOP  * * *
084D                    ;
084D A40B               RESCAN LDY DENS
084F 840A               STY KNTR
0851 A40A               ERASE  LDY KNTR
0853 A900               LDA #0
0855 8530               STA COLOR      ; SET COLOR TO BLACK
```

(Continued)

Listing 2 *(Continued)*

```
0857 208008          JSR CALC        ; FIND OLD STAR AND ERASE IT
085A A40A            LDY KNTR        ; MOVE STAR TOWARD OBSERVER
085C B11C            LDA (BASE),Y
085E 38              SEC
085F E50C            SBC SPEED
0861 911C            STA (BASE),Y
0863 B11E            LDA (TINT),Y    ; RECALL THIS STAR'S COLOR
0865 8530            STA COLOR
0867 208008          JSR CALC        ; PLOT STAR IN ITS NEW POSITION
086A C60A            DEC KNTR        ; NEXT STAR
086C D0E3            BNE ERASE
086E F0DD            BEQ RESCAN      ; ALWAYS TAKEN
0870           ;
0870           ; * * *  CALC ROUTINE  * * *
0870           ;
0870           ; IF STAR IS ON SCREEN, PLOT IT, ELSE CREATE NEW ONE.
0870           ; ENTER WITH STAR INDEX IN Y-REG AND COLOR SET.
0870           ;
0880                 ORG $880
0880 B118      CALC  LDA (HORIZ),Y   ; HORIZONTAL COORDINATE
0882 20B008          JSR TAN         ; FIND HORIZONTAL SCREEN POSITION
0885 A504            LDA QUOT
0887 18              CLC
0888 6914            ADC #14         ; SHIFT ORIGIN
088A 8508            STA SCRNX
088C C928            CMP #28
088E B013            BCS EDGE        ; OFF SCREEN?
0890 B11A            LDA (VERT),Y    ; LIKEWISE FOR VERTICAL COORDINATE
0892 20B008          JSR TAN
0895 A504            LDA QUOT
0897 18              CLC
0898 6914            ADC #14
089A C928            CMP #28
089C B005            BCS EDGE
089E A408            LDY SCRNX       ; POSITION IS ON THE SCREEN
08A0 4C00F8          JMP PLOT        ; PLOT THE STAR AND RETURN
08A3 A982      EDGE  LDA #82         ; OLD STAR WENT OUT OF VIEW...
08A5 911C            STA (BASE),Y    ; CREATE A NEW STAR...
08A7 4CF908          JMP NEW2        ; AND RETURN
08AA           ;
08AA           ; * * *  PROJECT 3-D POSITION ONTO 2-D SCREEN  * * *
08AA           ;
08AA           ; USES THE APPROXIMATION: X = TAN(X)
08AA           ; ACCUMULATOR = DISTANCE PERPENDICULAR TO LINE OF TRAVEL
08AA           ; (BASE),Y = DISTANCE PARALLEL TO LINE OF TRAVEL
08AA           ;
08B0                 ORG $8B0
08B0 8506      TAN   STA DVDEND      ; DIVIDE PERPENDICULAR DISTANCE...
08B2 B11C            LDA (BASE),Y
08B4 8507            STA DVSER       ; BY THE PARALLEL DISTANCE
08B6 A506            LDA DVDEND      ; CHECK THE SIGN
08B8 1016            BPL DIV         ; IF +, DIVIDE AND RETURN
08BA 49FF            EOR #FF         ; TWO'S COMPLEMENT DIVIDENT
08BC 8506            STA DVDEND
08BE E606            INC DVDEND
08C0 20D008          JSR DIV         ; DIVIDE
08C3 A504            LDA QUOT        ; TWO'S COMPLEMENT QUOTIENT
08C5 49FF            EOR #FF
08C7 8504            STA QUOT
08C9 E604            INC QUOT
08CB 60              RTS
08CC           ;
08CC           ; * * *  DIVISION ROUTINE  * * *
08CC           ;
08CC           ; QUOT = 32 * DVDEND / DVSER
08CC           ; REM = REMAINDER
08CC           ;
08D0                 ORG $8D0
08D0 A900      DIV   LDA #0
08D2 8504            STA QUOT
08D4 8505            STA REM
08D6 A60D            LDX POINT       ; POSITION OF DECIMAL POINT
08D8 0604      DIV1  ASL QUOT
08DA 0606            ASL DVDEND
08DC 2605            ROL REM
08DE A505            LDA REM
08E0 38              SEC
08E1 E507            SBC DVSER
08E3 9004            BCC DIV2
08E5 8505            STA REM
08E7 E604            INC QUOT
08E9 CA        DIV2  DEX
08EA D0EC            BNE DIV1
08EC 60              RTS
```

(Continued)

1 Mhz - 12 Bit A/D

for your Apple II Computer

The APPLESCOPE-HR12 analog to digital converter uses a high stability buried zener voltage reference and a flash A/D to give 12 bit accuracy with a 14 bit dynamic range.

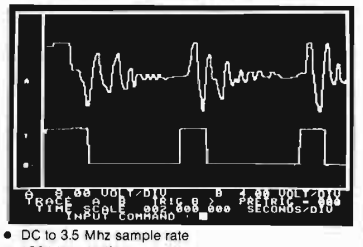
- DC to 1 Mhz Programmable Sample Rate
- 2048 Sample Buffer Memory
- Pretrigger Viewing
- Continuous or Single Sweep
- 4 Channel Software Support (requires additional power supply)
- External Trigger Input

The standard software provided with each APPLESCOPE-HR 12 includes all of the functions necessary to turn your Apple II computer into a high quality digital storage oscilloscope. In addition all of the SCOPE DRIVER options are being up-graded to handle the higher resolution data.

Price per channel **$695**

The original APPLESCOPE still provides the optimum price/performance trade off for those users requiring 8 bit converter resolution.

APPLESCOPE INTERFACE

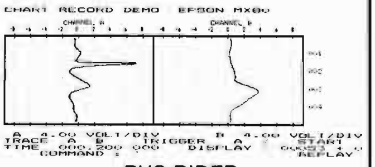


- DC to 3.5 Mhz sample rate
- 1024 byte buffer memory
- Pretrigger Viewing
- Programmable Scale Select
- Continuous and Single Sweep Modes
- Single or Dual Channel Trace

Price for the two board Applescope system is **$595**
EXTERNAL TRIGGER ADAPTER **$29**

SCOPE DRIVER Advanced software for the APPLESCOPE analog to digital converters makes full use of the computing power of the Apple II to create a total data acquisition system. Available options include:

- Signal Averaging-Acquires 1 to 999 signal sweeps and displays the averaged result.
- Digital Volt Meter - Allows use as real time DVM or use to measure points on an acquired sweep.
- Disk Storage - Allows automatic storage and recover of acquired data on floppy disks.
- Spectrum Analyzer-Calculates and displays frequency spectrum of acquired data.



BUS RIDER
LOGIC ANALYZER for the APPLE II

The BUS RIDER circuit card silently rides the Apple II peripheral bus and allows real time tracking of program flow. Software provided allows set up of trace parameters from the keyboard and read back of disassembled code after a program has been tracked.

- 32 bit by 512 sample memory buffer
- Monitors Data and Address bus plus 8 external inputs
- Trigger on any 32 bit word or external trigger
- Pretrigger viewing

The BUS RIDER is an invaluable development tool for anyone working with Apple II or Apple II+ computers.

Price **$395**

RC ELECTRONICS INC.
7265 Tuolumne Dr., Goleta, CA 93117
(805) 968-6614

AT LAST...
...FOR OSI

For investors
and financial managers
Stock portfolio analysis
$150.00
•in your office - instant valuations
•compound growth measurement
•pertinent company operating statistics
Stock financial statement analysis
$250.00
•input your interpretation of financial data
•analyze up to 10 years of data
•see mean, trend and stability
On-line data retrieval
$50.00
Accounting package
$150.00
DBM system
$200.00
for 8" floppy/hard disc under OS65U
NEW! **Full Screen Editor for Polled Keyboard**
$75.00
•for OS65D & OS65U
•machine language based
•type or cursor mode

write for details

Genesis Information Systems, Inc.
P.O. Box 3001 • Duluth, MN • 55803
Phone 218/724-3944

MICRObits

MICRObits
Color Computer Games
KIM-1 Hardware
PET Arcade Software
Business Software
Disassemblers
Indexes, Journals
Want Ads
Adventure Games
Graphic Utilities
Accessories
Computer Covers
Clocks
Expansion Products
News Items
Trade Offers
Books/Newsletters

These are just a few items you can find (or sell) through our MICRObits classified column. Have an item to sell, trade or locate? Try MICRObits. See page 115 in this issue for details.

Listing 2 *(Continued)*

```
08ED            ;
08ED            ; * * *  RANDOMIZE  * * *
08ED            ;
08F0                    ORG $8F0
08F0 203009 NEW1        JSR RND         ; CHOOSE DISTANCE
08F3 A510               LDA RANBYT
08F5 297F               AND #7F
08F7 911C               STA (BASE),Y
08F9 203009 NEW2        JSR RND         ; CHOOSE HORIZONTAL COORDINATE
08FC A510               LDA RANBYT
08FE 250E               AND TUBE        ; AVOID COLLISIONS
0900 F0F7               BEQ NEW2
0902 9118               STA (HORIZ),Y
0904 203009 NEW3        JSR RND         ; CHOOSE VERTICAL COORDINATE
0907 A510               LDA RANBYT
0909 250E               AND TUBE        ; AVOID COLLISIONS
090B F0F7               BEQ NEW3
090D 911A               STA (VERT),Y
090F 203009 NEW4        JSR RND         ; CHOOSE COLOR
0912 A510               LDA RANBYT
0914 290F               AND #F
0916 F0F7               BEQ NEW4        ; COLOR SHOULD NOT BE BLACK
0918 8505               STA REM         ; LET UPPER NIBBLE OF COLOR-
091A 0A                 ASL             ; MASK EQUAL LOWER NIBBLE
091B 0A                 ASL
091C 0A                 ASL
091D 0A                 ASL
091E 0505               ORA REM
0920 911E               STA (TINT),Y
0922 60                 RTS
0923            ;
0923            ; * * *  RANDOM NUMBER GENERATOR  * * *
0923            ;
0923            ; IDENTICAL TO LISTING 1
0923            ;
0930                    ORG $930
0930 18         RND     CLC
0931 A203               LDX #3
0933 B510               LDA $10,X
0935 CA                 DEX
0936 7510       LOOP1   ADC $10,X
0938 9510               STA $10,X
093A CA                 DEX
093B 10F9               BPL LOOP1
093D A203               LDX #3
093F F610       LOOP2   INC $10,X
0941 D003               BNE RTS1
0943 CA                 DEX
0944 10F9               BPL LOOP2
0946 60         RTS1    RTS
                        END
```

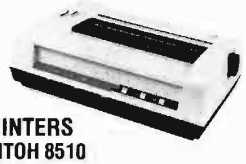
Listing 3

```
                ;* STOCHASTIC MUSIC DEMONSTRATION *
                ;*          FOR APPLE II           *
                ;*                                 *
                ;*        BY ART MATHENY           *
                ;***********************************
                ; MERGE WITH LISTING 1
                ; TO EXECUTE FROM MONITOR: 820G
                ;
                PITCH   EPZ $5          MUSICAL PITCH
                TIME    EPZ $6          DURATION OF THE SOUND
                RANBYT  EPZ $10
                RNDL    EPZ $4E
                RNDH    EPZ $4F
                RND     EQU $800
                SPKR    EQU $C030
                ;
0820                    ORG $820
0820 A54E       BEGIN   LDA RNDL        ; TRANSFER APPLE'S RANDOM SEED
0822 8510               STA $10         ; TO LOCATIONS $10 THRU $13
0824 8512               STA $12
0826 A54F               LDA RNDH
0828 8511               STA $11
082A 8513               STA $13
082C 200008     NUNOTE  JSR RND         ; CHOOSE PITCH
082F A510               LDA RANBYT
0831 0980               ORA #%10000000 ; PUTS ALL NOTES IN SAME OCTAVE
0833 29F0               AND #%11110000 ; SPACES THE PITCHES WITHIN THE
0835                                   ; MUSICAL SCALE
0835 8505               STA PITCH
0837 200008             JSR RND         ; CHOOSE DURATION OF NOTE
083A A510               LDA RANBYT
083C 8506               STA TIME
083E A605       PLAY    LDX PITCH       ; X-REG TIMES THE CYCLES
0840 88         YCOUNT  DEY             ; Y-REG TIMES THE PLAYING TIME
0841 D004               BNE XCOUNT
0843 C606               DEC TIME
0845 F0E5               BEQ NUNOTE
0847 CA         XCOUNT  DEX
0848 D0F6               BNE YCOUNT
084A 2C30C0             BIT SPKR        ; CLICK THE APPLE SPEAKER
084D 18                 CLC
084E 90EE               BCC PLAY
                        END
```

MICRO

FOR YOUR APPLE II

Industry standard products at super saver discount prices



PARALLEL PRINTERS
NEC 8023 or C-ITOH 8510

(Virtually identical) Specifications: • 100 CPS dot matrix printer • 80 column print–136 characters per line • Tractor/friction feed • 7 different print fonts included • 2K printer buffer • Proportional spacing • Bit image graphics and graphic symbols.

NEC 8023 or C-ITOH **$495**
NEC 8023 or C-ITOH 8510 with Parallel Interface and Cable **$550**
EPSON 100 with Parallel Interface and Cable **$749**

Z-80 CARD FOR YOUR APPLE
MICROSOFT SOFTCARD

With CP/M* and MBASIC.

(List: $399) **$289**

NEW! *Best Buy!!!*
ADVANCED LOGIC SYSTEM Z-CARD With C-PM*

Has everything the Softcard has except MBASIC. Works with Microsoft's disks too.

(List $269) ..... **Special at $195**

NEW!
ALS SYNERGIZER

CP/M* operating package with an 80 column video board, CP/M* interface, and 16K memory expansion for Apple II. Permits use of the full range of CP/M* software on Apple II. Includes SuperCALC.

(List: $749) **$549**

NEW!
U-Z-80 PROCESSOR BOARD (From Europe)

Software compatible with Softcard and ALS Software **$149**

MICROSOFT + PREMIUM SYSTEM

Includes Videx Videoterm, Softswitch, Microsoft and Softcard, Microsoft and Z-80 Card, and Osborn CP/M* Manual **$595**

JOYSTICK

Takes the place of two Apple Paddle Controllers.

From BMP Enterprises. Heavy duty industrial construction and cable. Non-self centering. With polarity switches for consistent motion control.

(List: $59) **$39**

MONITORS FOR YOUR APPLE

AMDEK 300G (18MHZ Anti-Glare Screen) **$179**
NEC 12" HIRES GREEN **$179**

SUPER SPECIAL!
SPECIAL 12" GREEN MONITOR **$99**

SPECIAL AND NEW

5 MEGABYTE HARD DISK

For Apple II. Supplied with controller. Use with CP/M, Apple DOS, & Apple Pascal **$1995**

5¼" DISK DRIVE

Use with standard Apple II disk controller. **$295**

5¼" FLOPPY DISKS

With hub rings. Box of 10.

With other purchase **$19.95**
Without purchase **$23.00**

16K MEMORY EXPANSION MODULE

The preferred 16K RAM Expansion Module from PROMETHEUS. Fully compatible with CP/M* and Apple Pascal*. With full 1-year parts and labor warranty. (List: $169) **$75**

WORD PROCESSING SPECIAL
WITH WORDSTAR AND SUPERCALC!

Do professional word processing on your APPLE. All necessary hardware and software included. Complete 80 column video display, enhanced character set, 16K memory board, Z-Card with CP/M* software, Wordstar and word processing software and SuperCALC.

(List: $1,128) .... **Special at $695**

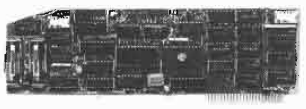
NEW! *from Prometheus!*
ExpandaRAM

The only 128K RAM card that lets you start with 16K, 32K, or 64K of memory now and expand to the full 128K later. Fully compatible with Apple Pascal, CP/M*, and Visacalc. No Apple modification required. Memory management system included with all ExpandaRAMs. Disk emulators included with 64K and 128K versions.

MEM-32 Two rows of 16K RAMS make a 32K RAM Card **$209**
MEM-64 One row of 64K RAM. With DOS 3.3 disk emulator **$299**
MEM-128 Two rows of 64K RAMS installed make a 128K Card. With DOS 3.3 disk emulator **$399**
MEM-RKT 64K RAM Add-On-Kits– 64K Dynamic RAMS. Each **$125**
VISICALC Expansion Program for MEM-128 ..... **$75**
MEM-PSL Pascal disk emulator for MEM-128 ..... **$45**

MODEMS FOR YOUR APPLE II

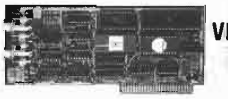
HAYES Smartmodem **$229**
MICROMODEM II **$279**



VERSAcard FROM PROMETHEUS

Four cards on one! With true simultaneous operation. Includes: (1) Serial Input/Output Interface, (2) Parallel Output Interface, (3) Precision Clock/Calendar, and (4) BSR Control. All on one card. Fully compatible with CP/M* and Apple Pascal*.

(List: $249) **$169**



80 COLUMN VIDEO DISPLAYS FOR APPLE II
SMARTERM

(Not to be confused with SUPRTERM)

Software switching from 80 to 40 and 40 to 80 characters. 9 new characters not found on the Apple keyboard. Fully compatible with CP/M* and Apple PASCAL*. With lowest power consumption of only 2.5 watts.

(List: $345) **$225**

SMARTERM EXPANDED CHARACTER SET

7" x 11" matrix with true decenders. Add to above **$40**

Best Buy! **Combination SMARTERM and EXPANDED CHARACTER SET**

Special at **$275**
VIDEX, VIDEOTERM **$249**
VIDEX ENHANCER II **$119**

NEW!
CENTRONICS COMPATIBLE PARALLEL INTERFACE

From PROMETHEUS. For use with Epson, NEC, C-ITOH, and other printers. Fully compatible with CP/M* and Apple Pascal*.

PRT-1, Only **$69**

GRAPHITTI CARD

Prints HIRES page 1 or 2 from onboard firmware. Features: True 1:1 aspect ratio, prints emphasized mode, reverse mode, rotates 90 degrees . . . plus more. Compare all this with the Grappler. We think you'll agree that this is the best graphics card on the market. Specify for use with EPSON, NEC-8023, C-ITOH Prowriter, or Okidata.

(List: $125) **$89**

SOFTWARE

WORDSTAR ..... **Special at $195**
SPELLSTAR **$125**
SUPERCALC **$175**
D BASE II **$525**
VISICALC **$149**
DB MASTER **$189**

All equipment shipped factory fresh. Manufacturers' warranties included. Please add $3.00 per product for shipping and handling. California: add 6% tax; BART Counties: 6½%

All items are normally in stock

Phone for Quick Shipment!

(415) 490-3420

. . . And we'll be here to help after you receive your order. Feel free to call the SGC Technical Staff for assistance.

SGC

The mail order specialists

342 Quartz Circle, Livermore, CA 94550

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

AT LAST, ENOUGH MEMORY TO GET THE JOB DONE RIGHT! AND AT THE RIGHT PRICE!!

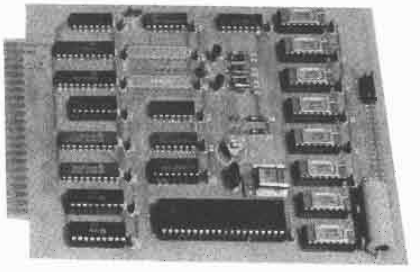
- *EASY, PLUG-IN INSTALLATION*
- *NO SOLDERING REQUIRED*
- *APPEARS AS 2 MEMORY BANKS, EACH 32K BYTES LONG*
- *OCCUPIES ADDRESS RANGE 0000-7FFF*
- *DESIGNED FOR COMPATIBILITY WITH ALL AIM AND SYM PROGRAMMING LANGUAGES, AND WITH BYTE-DOS AIM-65 DISK SYSTEM**
- *ALL ADDRESS AND DATA LINES FULLY BUFFERED*
- *USER UPGRADABLE TO ADD PIGGYBACK COLOR VIDEO GRAPHICS BOARD (SOON TO BE RELEASED)*

SPECIAL INTRODUCTORY PRICE

$399.00

(Effective 9/1/82 will be $475.00)

Includes assembled, tested board, connectors and complete instructions. Dealer inquiries invited.



SEND ME THE FULL STORY!

Name_____

Company_____

Address_____

City_____ State_____ Zip_____

Telephone_____ ☐ home ☐ work

master charge VISA

BYTE MICROSYSTEMS
CORPORATION
1477 ELKA AVENUE, SAN JOSE, CA 95129 408/446/0559

*BYTE-DOS $499 SYSTEM INCLUDES DISC CONTROLLER CARD, TEAC FD-50A DISC DRIVE

®AIM-65 is a trademark of Rockwell International
SYM-1 is a trademark of Synertek Systems Corp.

&Amper-Magic™

MACHINE LANGUAGE SPEED WHERE IT COUNTS... IN YOUR PROGRAM!

For the first time, Amper-Magic makes it easy for people who don't know machine language to use its power! Now you can attach slick, finished machine language routines to your Applesoft programs in seconds! And interface them by name, not by address!

You simply give each routine a name of your choice, perform the append procedure once at about 15 seconds per routine, and the machine language becomes a permanent part of your BASIC program. (Of course, you can remove it if you want to.)

Up to 255 relocatable machine language routines can be attached to a BASIC program and then called by name. We supply some 20 routines on this disk. More can be entered from magazines. And more library disks are in the works.

These routines and more can be attached and accessed easily. For example, to allow the typing of commas and colons in a response (not normally allowed in Applesoft), you just attach the Input Anything routine and put this line in your program:

```
xxx  PRINT "PLEASE ENTER THE DATE."; : & INPUT,DATE$
```

&-MAGIC makes it Easy to be Fast & Flexible!

PRICE: $75

Some routines on this disk are:

Binary file info
Delete array
Disassemble memory
Dump variables
Find substring
Get 2-byte values
Gosub to variable
Goto to variable
Hex memory dump
Input anything
Move memory
Multiple poke decimal
Multiple poke hex
Print w/o word break
Restore special data
Speed up Applesoft
Speed restore
Store 2-byte values
Swap variables

Anthro - Digital Software
P.O. Box 1385
Pittsfield, MA 01202
The People - Computers Connection

&-Magic and Amper-Magic are trademarks of Anthro-Digital, Inc.
Applesoft is a trademark of Apple Computer, Inc.

A New Character Set for the VIC-20

by Mike Bassman

A technique to design your own VIC characters is described. This involves changing the character-ROM printer and copying character definitions into RAM from their ROM locations.

Custom Characters
requires:
VIC-20

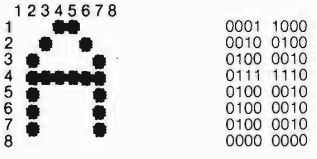
The new VIC-20 from Commodore packs a lot of wallop for $300. It includes many features that were previously found only on more expensive computers. The VIC uses a character set nearly identical to the PET's and very similar to those of Ohio Scientific, Sorcerer, and Atari computers. Special graphic characters are provided along with the regular alphanumerics. With these graphic characters, you can draw lines, build pictures, etc. The problem is that there are never enough characters available.

Although the VIC allows you to draw almost anything made of straight or curved lines, many games are more difficult to program because of the lack of tank, boat, or plane characters such as those found in OSI computers. Though the manual does not document it, it is possible to change some of the VIC's characters to those of your own choosing.

To create new characters, you must understand how the VIC stores its old ones. As with most computers, the VIC's characters are defined in an 8 by 8 dot matrix. That is, any character can be made of up to eight small dots (called pixels) across by eight down. Each pixel is represented in memory by one bit. If the bit is a 1, then the corresponding pixel will be lit when the character is displayed on the screen. An entire character consists of 64 bits, or eight bytes. The format in which the bytes are stored is as follows: the first byte represents the first row of pixels going across, the second byte is the second row, and so forth. For a more detailed example, see figure 1, where the character 'A' is shown in both pixel and binary formats.

The VIC keeps all of its characters in 4K of ROM. Since each character occupies eight bytes of memory, the ROM must contain 512 characters. At first glance, it does not seem to have that many, but this is how the 4K is used. The first 128 characters (1K) are the familiar upper case/graphics set. The next 128 are the same ones, only in reverse. This is followed by the lower/upper case set and its reverse set.

Figure 1: **Binary Representation of Character in Memory**

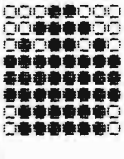


On most computers, including the PET, the character ROM is not addressed in the microprocessor's normal memory space, but is addressed so that it is only available to the CRT controller chip. In the VIC the ROM is actually addressed from $8000 to $8FFF in the 6502's address space!

Since the character information is stored in ROM, it can't be changed. However, since the pointer to the character ROM is stored in RAM, it can be changed to point anywhere else, including RAM. The pointer is at 36869, 36870. Its normal value is 240, enabling the upper case/graphic set, while a value of 242 will enable the alternate lower/upper case set. POKEing a 255 into 36869 will move the pointer from its normal 32768 address to 7168. This just happens to be right at the top of BASIC RAM, which normally runs from 4096 to 7679. By lowering the upper limit from 7679 to 7168, we have stolen 512 bytes from BASIC, or enough for 64 characters. Since it is impossible to put 4K worth of characters into 512 bytes, you have to select the 64 used most. Normally these would be the upper case letters, the numbers, and various punctuation marks, or the characters with screen codes ranging from 0 to 63. Of these, certain ones, such as the @ and the British pound sterling symbol, are seldom used, and can be replaced with your custom characters.

There are a few tricks you should know before attempting to make new characters. Most importantly, you have to reserve locations 7168 to 7679 so they are not disturbed by BASIC. There are two pointers that must be adjusted to accomplish this: the top-of-BASIC pointer (51-52), and the top-of-string storage pointer (55-56). As with every

Figure 2: **Alien Character**



Binary	Hex	Decimal
00011000	18	24
00111100	3C	60
01011010	5A	90
11111111	FF	255
11111111	FF	255
11111111	FF	255
01111110	7E	106
01111110	7E	106

6502 pointer, the low byte is stored first, followed by the high byte. Both pointers point to 7680, and must be changed to point to 7168. Instead of recreating all of these characters, it is faster to copy them from their original ROM locations.

We are now ready to define a custom character — an alien, which you might find useful in game programs. The easiest way to go about this is to take a piece of graph paper, mark off an 8 by 8 segment, and fill in the appropriate squares. Now take each row and make a binary representation for it. Convert the binary for each row into decimal, because this information will be stored in BASIC DATA statements. Our alien is defined and digitized in figure 2.

This program will put the character shown in figure 2 over the character with screen code 0 (the '@'). Now when you hit the '@' key, you will get an alien. The alien can now be used in your own program for gaming or anything else. The unfortunate part is that all those other characters are no longer available; only the 64 you originally moved down are available. If you try to use any others, you will get whatever random gibberish happens to lie beyond 7679.

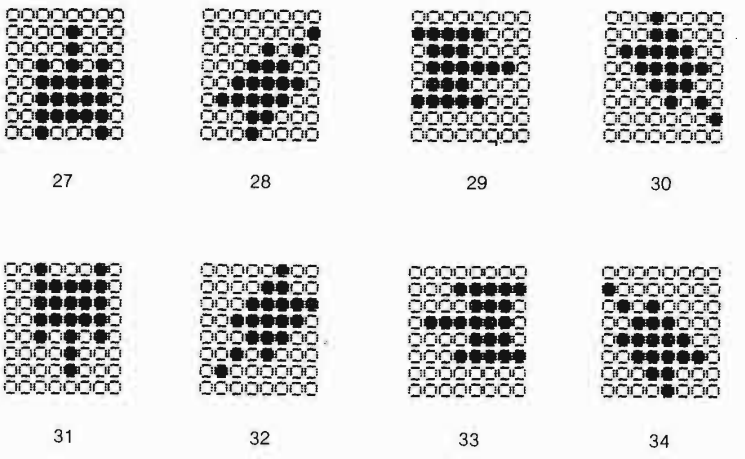
Listing 2 is a program that defines a whole series of tank characters — one for each of eight 45-degree angles. They are shown in figure 3, in their pixel formats. Below them are the screen codes they will be assuming. Now many games that were before restricted to the computers with the appropriate game characters can be programmed on the VIC.

The screen codes for the tank characters are from 27 to 34, as shown in figure 3. They are contiguous for programming ease. The tanks replace the brackets, the up and left arrows, the British pound sign, the space, the exclamation point, and the quote (in that order). If you list a program with this character set implemented, your punctuation is going to look awfully funny!

In addition to defining the tank characters, the program lets you drive the tank around the screen. The controls are 'Z' to turn left, '/' to turn right, and space to move in the direction you are pointed. To make a complete program, the lines of the program in listing 2 must be overlayed onto those of the program in listing 1.

Contact the author at 39-65 52nd St., Woodside, NY 11377.

Figure 3: **Eight Tank Characters**



Listing 1

```
10 POKE 52,28
20 POKE 56,28
30 S=7168
40 F=32768
50 FOR K=0 TO 63*8
60 X=PEEK(F+K)
70 POKE S+K,X
80 NEXT K
90 READ X
100 IF X=-1 THEN POKE 36869,255: GOTO 1000
110 FOR K=0 TO 7
120 READ N
130 POKE S+8*K,N
140 NEXT K
150 GOTO 90
160 DATA 00,24,60,90,255,255,255,126,126
500 DATA -1
1000 PRINT "CHARACTER SET COMPLETED": END
```

Listing 2

```
5 PRINT"£": FOR K=7680 TO 8185: POKE K,35: NEXT
200 DATA 27,0,8,8,42,62,62,62,34
210 DATA 28,0,1,10,28,62,124,24,16
220 DATA 29,0,248,112,126,112,248,0,0
230 DATA 30,16,24,124,62,28,10,1,0
240 DATA 31,34,62,62,62,42,8,8,0
250 DATA 32,4,12,31,62,28,40,64,0
260 DATA 33,0,31,14,126,14,31,0,0
270 DATA 34,0,128,80,56,124,62,24,8
280 DATA 35,0,0,0,0,0,0,0,0
1000 POKE 36879,14: UL=7680: LL=22: S=35
1005 FOR K=0 TO 21: POKE UL+K,45: POKE 8164+K,45
1010 POKE UL+K*LL,45: POKE UL+K*LL+21,45: NEXT
1020 L=UL+LL*5+5:C=27:D(1)=-22:D(2)=-21:D(3)=1:D(4)=23:D(5)=22
1025 D(6)=21: D(7)=-1: D(8)=-23
1030 POKE L,C
1050 GET A$: IF A$="" THEN 1050
1060 IF A$<>"Z" THEN 1090
1070 C=C-1: IF C=26 THEN C=34
1080 GOTO 1030
1090 IF A$<>"/" THEN 1120
1100 C=C+1: IF C=35 THEN C=27
1110 GOTO 1130
1120 IF A$<>" " THEN 1050
1130 IF PEEK(L+D(C-26))<>35 THEN 1050
1140 POKE L,S: L=L+D(C-26): GOTO 1030
```

MICRO™

ARK COMPUTING

Your Salvation In The Sea Of Inflation.

P.O. Box 2025 ● Corona, CA ● 91720

Microtek Parallel Printer Interface $59.95

This popular printer interface card, manufactured by Microtek, is a steal at $59.95. The Printer card comes complete with cable and a Centronics compatible connector (Amphenol). Works with Basic, CP/M, and Pascal. This card also has graphics capabilities.

Diskettes w/Hubring 10 $19.95

High quality diskettes at a bargain price. Everyone needs diskettes for backing up other disks, saving programs, etc. We buy these diskettes in bulk and then pass the savings onto you. Remember, they do have hubring and come with a 1 year guaranty. NOTE: Please call for quanities of 100 or more for special pricing.

Auto-Repeat Device $14.95

For those who want the feature that many Main-Frame Computers have, here is the Auto-Repeat device. This device does not take up a slot or crowd your APPLE II. Auto-Repeat fits right on the newer style Apple Keyboards. The speed of the Auto-Repeat can be varied to suit your needs. NOTE: Auto-Repeat device will only work on newer APPLE II keyboards.

Lazer Lower Case + Plus II $19.95

For the budget minded user with quality in mind. This lower case adapter will work with all Rev. 7 and later APPLE II's. The Lower Case + Plus II includes Basic and Pascal software. Works with many popular word processors.

Also Available From LAZER

ANIX 1.0 $34.95, This software program is a set of incredible disk utilities with a UNIX-like operating system. LAZER Pascal $29.95 A unique systems programming language for Anix 1.0 with many features of the 'C' programming language.

Disk Drive Cables $24.95

These cables replace the cables that are connected to the Apple Disk Drive already. If you feel that can not put your Disk Drives where you want them, here's your answer. The Cables are 4' long and are pre-tested for your assurance.

ORDERING INFORMATION

We accept: VISA/MASTERCARD (include card #, expiration date, and signature), Cashier or Certified Checks, Money Orders, or Personal Checks (please allow 10 business days to clear). We also accept COD's (please include $2.00 COD charge).

Please add 3% for shipping and handling (minimum $2.00). Foreign orders please add 10% for shipping and handling (minimum ($10.00).

California residents add 6% sales tax. All equipment is subject to price change and availability without notice. All equipment is new and complete with manufacturer's warranty.

(714) **735-2250**

Decision Systems

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

SOFTWARE FOR THE APPLE II*

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

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

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

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

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

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

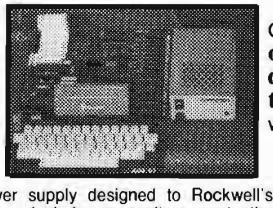
(Add $4.00 for Foreign Mail)

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

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.

ORDERS: (714) 369-1084

VISA

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

master charge

80 x 25

PET/CBM™

2000/3000/4000 Series

not using a CRT, or display controller chip

$275.00*

Select either
80 x **25** or **40** x **25**

On The Built-in Display

From the keyboard or program

Displays the full, original character set

Available from your local dealer or:

EXECOM CORP.
1901 Polaris Ave.
Racine, WI 53404
Ph. 414-632-1004

*Plus installation charge of $75.00

Available only for Basic 3.0 & Basic 4.0
PET& CBM™a
trademark of Commodore Business Machines

computer case company

comp case
NOW TOLL FREE
800-848-7548

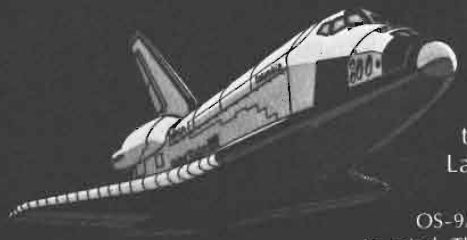
• AP104

Attaché style cases for carrying and protecting a complete computer set-up. Constructed of the highest quality luggage material with saddle stitching. Will accommodate equipment in a fully operational configuration along with manuals, working papers and disks. Never a need to remove equipment from case. Simply remove lid, connect power and operate.

• AP101	Apple II with Single Disk Drive	$109
• AP102	Apple II with Double Disk Drives	119
• AP103	Apple II, 9 inch Monitor & Double Drives	129
• AP104	Apple *III*, two additional Drives & Silentype	139
• AP105	12 inch monitor plus accessories	99
• P401	Paper Tiger 440/445/460	99
• P402	Centronics 730/737	89
• P403	Epson MX70 or MX80	89
• P404	Epson MX100	99
• P405	IDS 560 or Prism Printer	109
• CC80	Matching Attaché Case (5")	85
• CC90	Matching Attaché Case (3")	75
• CC91	Matching Accessory Case	95

computer case company
5650 INDIAN MOUND CT. COLUMBUS, OHIO 43213 (614) 868-9464

What's Good for the Space Shuttle is good for your Apple II! . . .



MICROWARE® creator of OS-9/BASIC Ø9 (used by N.A.S.A., and leading Universities, government agencies, and corporations Worldwide) joins with STELLATION TWO to deliver the same Operating system and Programming Language to the APPLE II.

OS-9/BASIC Ø9 are the result of a 3 year research project-designed with the 6809 in mind. This "Operators dream machine" combines with THE MILL microprocessor board to provide Apple II users with software features previously reserved for Mainframes and mini's.

JUST PLUG IN THE MILL AND LET BASIC Ø9 WORK FOR YOU! *other Stellation Two products include:*

Spooler: *Allows Apple II to print while processing*
Pascal Speedup: *THE MILL with software for a 50% faster Apple with Pascal*
Floating Point: *Extends the MILL's power to floating point numbers*

STELLATION TWO
The Lobero Building P.O. Box 2342
Santa Barbara, Ca, 93120
(805) 966-1140 TELEX 658439

MICROWARE®

OS-9 is a trademark of Microware. BASIC Ø9 is a trademark of Microware and Motorola.

Apple II is a trademark of Apple Computer.

MICRO™

PET Vet

By Loren Wright

PET and the IEEE-488 Bus

The PET is one of the few personal computers to incorporate, as standard, support of the IEEE-488 bus. In fact, the system is built around it! Nearly all communication with peripheral devices is through the bus, and even the PET's internal devices — the screen, keyboard, and cassette interfaces — are treated as IEEE devices. Most PET owners probably know that CBM printers and disk drives are driven from the IEEE bus. However, in addition to CBM devices, the PET can also communicate with a wide variety of peripherals, produced by different manufacturers, who never even thought of connecting them to a PET. These manufacturers include Hewlett-Packard, Tektronix, Fluke, and many others; the devices include non-CBM disk drives, logic analyzers, X-Y plotters, frequency analyzers, digital voltmeters, and much more sophisticated instruments. Most other computers that can act as controllers on the bus cost much more than the PET.

A typical IEEE system includes a controller and one or more devices, which are considered listeners or talkers. Each device is assigned a number, and it will neither talk nor listen unless addressed using this number. There can be as many as 32 devices on the PET's bus, all connected at once.

The PET manipulates the devices through logical files. When a file (identified with a unique number) is OPENed, the device number is specified, as well as a secondary address, which identifies a particular function of the device. For instance, for the CBM cassette units, a secondary address of 0 allows the PET to read from the unit, a 1 to write to it, and a 2 to write, with an end-of-tape mark written when the file is CLOSEd. Once a file is open, all subsequent operations involving the specified function of the specified device need only refer to the file number.

Control of the devices is not limited to the secondary addresses. Many devices are set up to handle commands in the form of character strings. CBM disk drives are good examples of this. The secondary address 15 is the command channel, and information in the form of characters is passed in both directions on this channel. For instance, the strings "10" and "INITIALIZE0" both mean to initialize the disk in drive 0. Error and status information is sent over the channel in the other direction. For devices such as digital X-Y plotters, a character string may tell it to move the pen to 5,7 and draw a line from there to 10,12.

Programming control of IEEE devices is very easy, but physically putting together the system is even easier. Every instrument (besides the PET itself) has the same connector, and it doesn't matter which end of a cable goes where.

IEEE Tricks

When running a program that outputs data to a printer, it is often handy to be able to try it out on the screen first. You might expect to have to write your output routine first with PRINT statements, and then go back and replace them all with PRINT# statements. Here is an illustration of a technique to avoid this.

```
 10 input "to printer";y$
 20 if y$="y" then dv=4: goto 40
 30 dv=3
 40 open 1,(dv)
 .
 .
 .
100 print#1,"first line"
110 print#1,"second line"
 .
 .
 .
999 close 1
```

When you answer "y" to the prompt, the file will be opened with a device number of 4 and the output will go to the printer. If you answer anything else, 3 will be the assigned device number, and all output will go to the screen. If, instead of including the prompt, you had just written the 3 into the OPEN statement in line 40, then you would only have to change that one line when you finally wanted output to go to the printer. The same method could be used if you were offering the user a choice between writing data to cassette or to disk. However, the secondary addresses would have to be taken care of, too.

A question mark is always printed on a PET BASIC INPUT statement. What if you want to suppress that question mark? One way is to specify the keyboard (IEEE device 0) in an OPEN statement and use an INPUT# statement instead.

```
10 open 1,0
20 print"word: ";
30 input#1,w$
40 if w$<>"end" then 20
50 close 1
```

Notice that, unlike the INPUT command, the INPUT# command cannot include a prompt string. Notice also that hitting the return key alone doesn't phase INPUT# one bit — it must have at least one character before it is happy.

The PET's CMD command can be very handy. You have probably used it often to get BASIC listings on a printer. What it actually does is direct all output that normally goes to the screen to a logical file. After you have finished sending your listing to the printer, you may have noticed that the cursor behaves peculiarly. When you hit RETURN, it will move ahead only one space, instead of moving to the next line.

One way to restore normal keyboard operation is to type PRINT#1 (if 1 is the file number). If you aren't going to use the file again right away, it is also a good idea to close it. Remember that PRINT# can be abbreviated by typing P, followed by a shifted R. My favorite method of exiting the CMD mode is to just type some nonsense, like 'dfd.' Of course you get a syntax error, but you also exit CMD.

BASIC cannot take advantage of everything the IEEE-488 bus has to offer. With machine-language programming, it is not only possible to get the maximum in speed, but you can also do some unusual things like spooling. Wordcraft 80 and other programs allow you to send a file directly from the disk drive to the printer while you have control of the PET.

For more detailed information on the IEEE-488 bus, consult the three-part article by Gregory Yob that began with the July, 1980 issue of *BYTE* ("Get Your PET on the IEEE-488 Bus") and the book *PET and the IEEE Bus* by Eugene Fisher and C.W. Jensen, published by Osborne/McGraw-Hill (1980).

Commodore's New Machines Set Straight

With the official U.S. announcement of Commodore's newest computers at NCC in early June, it is finally clear exactly what the new models are:

Commodore-64 is a 64K color-and-sound computer designed to compete with Apple II and Atari computers. It will be available at dealers late this summer at a suggested retail of $595. It physically resembles the VIC-20, requiring a color TV or monitor and supporting all the VIC's peripherals. However, its 25 × 40 screen size, 64K of RAM, and more sophisticated graphics distinguish it. Options available include a PET emulator, IEEE-4888 cartridge, and Z-80 processor board.

The MAX machine (previously identified here and elsewhere as the Ultimax) also physically resembles the VIC-20, but it uses a flat membrane-type keyboard instead. The graphic and sound capabilities are more sophisticated. However, only 2K of RAM is included. Add-ons include a BASIC language cartridge, cassette machine, a wide variety of games, and, of course, more memory. The suggested retail will be $179.95.

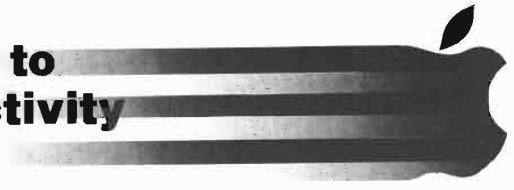
The P128 is billed as the third generation in Commodore's PET line (i.e. home, educational, recreational). At a suggested retail price of $995, there is quite a bit included: 128K of RAM, expandable 256K internally and 640K externally; 25 × 40 16-color display on the user's TV or monitor; 200 × 320-pixel high-resolution graphics; built-in RS-232 and IEEE-488 interfaces; 10 programmable function keys; individual cursor movement keys; sophisticated music synthesis capability. It is also possible to add a Z-80 processor board to allow CP/M compatibility. The P128 should be available sometime in the fall.

The B128 is the new business entry in Commodore's line at a suggested retail of $1695. Internally it is very similar to the P128 described above. However, it includes an attached 80-column green phosphor screen, two 5¼" floppy disk drives, and a business keyboard. The B128 should also be available in the fall.

The BX256, very much like the B128, comes with 128K additional memory, including the 8088 processor for 16-bit capability. Suggested retail will be $2995, and it should be available in the fall.

MICRO

S&H Software enables Apple][™ users to increase speed and productivity as much as 500%.



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

UBI 4.0, with The DOS Enhancer,™ allows execution of BASIC and Binary programs up to 500% faster, depending on file length, and is completely compatible with standard Apple DOS 3.3 programs. For improved file management a new "FREE" command in DOS determines free space on disks.

UBI 4.0 breaks the language and time barriers between Apple][hardware and software by loading the RAM card with FPBASIC/INTBASIC (or your own program) in 1.7 seconds and by providing one-stage booting with DOS 3.3 or DOS 3.2.1 PROMS.

The UBI 4.0 package includes: training, utility, support and demo disks with complete documentation.

System requirements: 48K Apple][or][+, ROM/RAM card, DOS 3.3/DOS 3.2.1 and 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][data files, none does the job nearly so fast as A-S/M.

A-S/M can sort/merge from one to five unsorted files into a single file of up to 125K per disk.

Because sort routines can take up to 50% of computer running time in many business applications, you'll reap continuing benefits having the A-S/M "invisible speed demon" on your Apple][team – saving 20 - 30 minutes a day of a human clerk's time spent "waiting" needlessly.

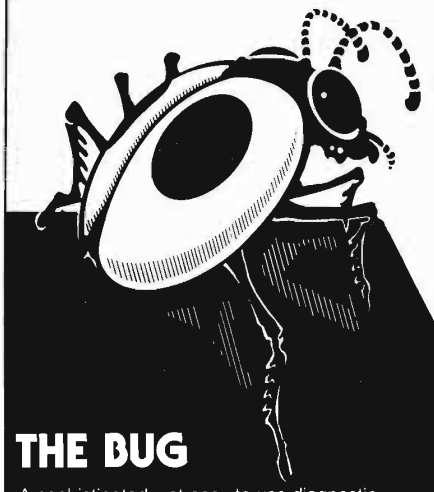
The A-S/M "speed demon" package includes: training disk and utility disks with complete documentation.

System requirements: 48K Apple][, ROM or RAM card, DOS 3.3 and one or more disk drives or 48K Apple][+, DOS 3.3 and one or more disk drives.

Available from your dealer. **Mail Order:** Send check to S&H Software, Box 5, Manvel, ND 58256. **Credit Cards:** Phone Cybertronics International Clearinghouse at 212-531-3089. (Overseas Airmail: Add $3.00 postage and handling.)

S&H Software

Box 5 Manvel ND 58256
(701) 696-2574



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**

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.

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. MO
West Bloomfield, Michigan
48033 • (313) 399-8877
Visa and Mastercard
Welcome
Please add $1.25
postage and
handling per
diskette

FORTH-79

Ver. 2 For your APPLE II/II+

The complete professional software system, that meets ALL provisions of the FORTH–79 Standard (adopted Oct. 1980). Compare the many advanced features of FORTH–79 with the FORTH you are now using, or plan to buy!

FEATURES	OURS	OTHERS
79-Standard system gives source portability.	YES	
Professionally written tutorial & user manual	200 PG.	
Screen editor with user-definable controls.	YES	
Macro-assembler with local labels.	YES	
Virtual memory.	YES	
Both 13 & 16-sector format.	YES	
Multiple disk drives.	YES	
Double-number Standard & String extensions.	YES	
Upper/lower case keyboard input.	YES	
LO-Res graphics.	YES	
80 column display capability	YES	
Z-80 CP/M Ver. 2.x & Northstar also available	YES	
Affordable!	$99.95	
Low cost enhancement option:		
Hi-Res turtle-graphics.	YES	
Floating-point mathematics. Powerful package with own manual, 50 functions in all, AM9511 compatible.	YES	
FORTH–79 V.2 (requires 48K & 1 disk drive)		$ 99.95
ENHANCEMENT PACKAGE FOR V.2 Floating point & Hi-Res turtle-graphics		$ 49.95
COMBINATION PACKAGE		$139.95

(CA res. add 6% tax: COD accepted)

MicroMotion
12077 Wilshire Blvd. #.506
L.A., CA 90025 (213) 821-4340
Specify APPLE, CP/M or Northstar
Dealer inquiries invited.

IS THIS YOUR CUSTOMER?

When your customer needs you, he needs you NOW! Why make him search through a ton of local phone books or back issues of magazines for your number?

THE COMPUTERIST'S DIRECTORY YELLOW PAGES are the one quick reference to your products or services for the personal and small business computerist. Hardware, software, services, supplies and much more are conveniently indexed and cross referenced, placing your company on your customer's desk six months at a time. Now you can have inexpensive advertising insurance assuring that once your product announcement and ads have made their impression, your customers will be able to find you when they are ready to buy!

THE COMPUTERIST'S DIRECTORY WHITE PAGES are the first attempt by any publication to catalog the individuals and groups that are creating the Information Revolution. The White Pages contain listings by individuals, clubs, user groups, computerized bulletin boards and professional associations. Many listings also include network I.D. numbers (Source, CompuServe, etc.) making the Computerist's Directory a pioneer in facilitating Electronic Mail & Conferencing.

One Computerist's Directory is worth a room full of local phone directories.

HE ONLY HAD TO LOOK IN ONE PLACE!

the Computerist's Directory

The National Phone Book of Computing

PO BOX 405
FORESTVILLE, CA 95436

(707) 887-1857

BOOK/DISK COMBINATIONS

BECAUSE YOU DIDN'T BUY YOUR APPLE™ TO PRACTICE YOUR TYPING

■ Practical manuals that show you how to program your Apple™ for business, learning, and pleasure.
■ Convenience disks that contain all the programs and subroutines in the books they accompany—error free and ready to run.

PLUS the Wiley expertise that has helped more than a million people learn how to program, use, and enjoy microcomputers.

Look for them at your favorite bookshop or computer store. Or, check the sets that interest you, fill in the ordering information, and mail us this ad.

1807 175 YEARS OF PUBLISHING 1982

JOHN WILEY & SONS, Inc.
605 Third Avenue, New York, N. Y. 10158
In Canada: 22 Worcester Road, Rexdale, Ontario M9W 1L1

For faster service call toll free: 800-526-5368. In New Jersey, call collect: (201) 797-7809. Order Code # 3-6768. VISA, MasterCard, American Express accepted on phone orders.

☐ **APPLE™ BASIC: DATA FILE PROGRAMMING SET**
LeRoy Finkel & Jerald R. Brown

How to program and maintain data files for billings, catalogs and lists, numerical and statistical data, and much more. Includes one 5¼" disk for Apple II™. (Requires one 16 sector disk drive, 32K of memory) 1-89843-0 $32.90

☐ **GOLDEN DELICIOUS GAMES FOR THE APPLE™ COMPUTER SET**
Howard M. Franklin, Joanne Koltnow, & LeRoy Finkel

Step-by-step instructions for designing game programs that turn your Apple II™ into a home entertainment center—whether you're a novice, intermediate or advanced programmer. Includes two 5¼" disks for Apple II™. (Requires one 16 sector disk drive, 32K of memory) 1-89842-2 $47.90

Please send the sets indicated for 15-DAY FREE EXAMINATION. (Restricted to the continental U.S. and Canada.)
MAIL TO: JOHN WILEY & SONS, Inc.
P.O. Box 092, Somerset, N.J. 08873

☐ **Payment enclosed,** plus sales tax. Wiley pays normal postage/handling. We normally ship within 10 days. If shipment cannot be made within 90 days, payment will be refunded.
☐ **Bill me.** ☐ **Bill firm or institution.**

NAME AFFILIATION
ADDRESS CITY/STATE/ZIP
SIGN HERE 092-3-6768

Prices subject to change without notice. Apple and Apple II are trademarks of Apple Computer, Inc.

POWER-Aid for the PET

by F. Arthur Cochrane

This machine-language program adds commands to the BASIC programming utility POWER for the Commodore PET. The source code is in the MAE assembler format and takes advantage of conditional and interactive assembly. This allows the assembler to make decisions in its assembly and thus assemble Power-Aid to function with all three versions of POWER.

POWER-Aid
requires:
PET/CBM (all except original operating system)
POWER ROM

Editor's note: POWER was reviewed by Loren Wright in the July, 1982 "PET Vet" (MICRO 50:69).

POWER is a programmer's utility package for the PET written by Brad Templeton and sold by Professional Software, Inc. It is programmed in a 4K ROM for the $9000 socket inside the PET.

The manual, by Jim Butterfield, does an excellent job of describing POWER's commands: AUTO, DELete, DUMp, FIX, Call monitor (MLM), OFF, RENumber, SELect, TRaCe, WHY, Execute, Find, and Change. The manual also contains several appendices, including a thorough description of the memory locations used by POWER and a description of routines that can be called in POWER by the machine-language programmer. Another appendix covers adding more commands to POWER.

Three locations are used by POWER to add commands. Two of these are used for an indirect jump vector that points to the added machine code. The third location is used by POWER as a checksum for the indirect vector.

This program, Power-Aid, adds 15 more commands to POWER. Power-Aid modifies the three locations to point to itself and uses one of the POWER routines to list a BASIC line. Power-Aid can be assembled for all three POWERs by using conditional and interactive assembly available in the MAE assembler.

As presented here, Power-Aid is for

***Listing 1:* POWER-Aid initialization routine demonstrates how to add commands to POWER.**

```
                    0010 ; "POWER-AID.M2A"
                    0020
                    0030
                    0040
7900- 20 00 90       0050 COLD       JSR POWER
7903- AD F3 7F       0060            LDA WBEGIN          ;PROTECT EDITOR
7906- AE F4 7F       0070            LDX WBEGIN+1        ; POINTERS
7909- 85 FB          0080            STA *TMP0           ;SAVE TO PRINT RESTART
790B- 86 FC          0090            STX *TMP0+1
790D- E0 80          0100            CPX #$80            ;ABOVE RAM
790F- B0 07          0110            BCS COLD010
7911- 85 34          0120            STA *MEMSIZ
7913- 86 35          0130            STX *MEMSIZ+1
7915- 20 EB 9F       0140            JSR WRP             ;FIX POINTERS
                     0150
7918- AD F5 7F       0160 COLD010    LDA WENTRY
791B- AE F6 7F       0170            LDX WENTRY+1
791E- 8D D7 03       0180            STA UCOMVEC         ;STORE JUMP VECTOR
7921- 8E D8 03       0190            STX UCOMVEC+1
7924- 18             0200            CLC
7925- 6D F6 7F       0210            ADC WENTRY+1
7928- 49 A5          0220            EOR #%10100101
792A- 8D D6 03       0230            STA UCVCHK          ;STORE CHECK BYTE
                     0240
792D- A2 14          0250            LDX #MESSGE-NGMSG
792F- 20 35 79       0260            JSR PRTMSG
7932- 4C 1D 7E       0270            JMP OUTDEXHEX       ;PRINT RESTART ADDRESS
                     0290
                     0300
                     0310 ;PRINT A MESSAGE
                     0320
7935- BD 6B 7F       0330 PRTMSG     LDA NGMSG,X         ;OUTPUT MESSAGE
7938- F0 06          0340            BEQ MSGEND
793A- 20 D2 FF       0350            JSR PRINT
793D- E8             0360            INX
793E- D0 F5          0370            BNE PRTMSG
7940- 60             0380 MSGEND     RTS
                     0390
                     0400 ; HOLD ON SPACE KEY PRESSED OR SHIFT KEY
                     0410
7941- 24 98          0420 AD70EE     BIT *SFST           ;SHIFT KEY DOWN
7943- D0 FC          0430            BNE AD70EE          ;YES, HOLD
7945- 20 61 79       0440            JSR AD7103
7948- D0 21          0450            BNE AD710D
794A- 20 61 79       0460 AD70F8     JSR AD7103
794D- F0 FB          0470            BEQ AD70F8
794F- C9 FF          0480            CMP #$FF
7951- F0 F7          0490            BEQ AD70F8
7953- 48             0500            PHA
7954- 20 61 79       0510 AD7100     JSR AD7103
7957- C9 FF          0520            CMP #$FF            ;WAIT FOR KEY RELEASE
7959- D0 F9          0530            BNE AD7100
```

(Continued)

a 32K PET, but the assembly location can easily be modified for any memory size PET. Power-Aid could also be assembled and burned into an EPROM and plugged into the $A000 socket to work with POWER. In EPROM, Power-Aid would be available with a SYS and would not have to be loaded from disk each time the PET is reset or powered up.

You might question why you should go to all the trouble of machine code. Just about all of the functions of Power-Aid could be written as instant subroutines for use by POWER. This is true, but the subroutines would be written in BASIC and would run at the speed of BASIC rather than the much faster machine code. Also, Power-Aid stores itself at the top of memory and protects itself from BASIC. If these commands were written as instant subroutines they would be overwritten on any LOAD of another BASIC program. They could be merged back in with the XEC command but their line numbers might overwrite some of the lines of the BASIC program in memory.

Listing 1 is in MAE assembly language. The object code generated with this particular assembly is for the 4040 version of POWER. Appropriate responses to the prompt will generate a version that runs on your system.

Dump the Screen to a Printer

Syntax: CRTA [, printer#]
CRTC [, printer#]

The screen is output to a printer connected to the PET. The CRTC command (to a Commodore printer) will be exactly like the screen. The CRTA command (to an ASCII printer) will be in upper case only if the PET is in graphics mode, or lower/upper case if the PET is in lower-case mode.

List a Program from the Disk

Syntax: FIL "program filename"
[, disk#]

This command will list a BASIC program on the disk directly to the screen without affecting the contents in the memory.

Convert Hex and Decimal Numbers

Syntax: HEX $hex number
HEX decimal number

The HEX command will convert hex to decimal and decimal to hex. This will be very useful in figuring

***Listing 1** (Continued)*

```
                    0540
                    0550 ; ZERO CHARACTER COUNTER
                    0560
795B- A9 00         0570 ZEROCHAR   LDA #0
795D- 85 9E         0580            STA *NDX
795F- 68            0590            PLA
7960- 60            0600            RTS
                    0610
                    0620
7961- AD 12 E8      0630 AD7103     LDA $E812
7964- CD 12 E8      0640            CMP $E812
7967- D0 F8         0650            BNE AD7103
7969- C9 FB         0660            CMP #$FB            ;SPACE KEY(G)/6(B)
796B- 60            0670 AD710D     RTS
                    0680
                    0690 ; POWER-AID CHECKS FOR COMMANDS
                    0700
796C- A2 00         0710 ENTRY      LDX #0
796E- 86 FE         0720            STX *CMDLEN
7970- A1 77         0730            LDA (TXTPTR,X)      ;CHECK FOR TOKENS
7972- 30 1C         0740            BMI TOKEN
7974- A4 77         0750 AD7158     LDY *TXTPTR
7976- B9 00 02      0760 AD715A     LDA BUFOFS,Y
7979- 38            0770            SEC
797A- FD 9B 7F      0780            SBC TABLE,X
797D- F0 14         0790            BEQ AD7176
797F- C9 80         0800            CMP #$80
7981- F0 14         0810            BEQ AD717A
7983- E6 FE         0820            INC *CMDLEN
7985- E8            0830 AD7169     INX
7986- BD 9A 7F      0840            LDA TABLE-1,X
7989- 10 FA         0850            BPL AD7169
798B- BD 9B 7F      0860            LDA TABLE,X
798E- D0 E4         0870            BNE AD7158
7990- 4C FA 9F      0880 TOKEN      JMP NORMCOM         ;NOT FOUND
7993- E8            0890 AD7176     INX
7994- C8            0900            INY
7995- D0 DF         0910            BNE AD715A
7997- 84 77         0920 AD717A     STY *TXTPTR
7999- A5 FE         0930            LDA *CMDLEN
799B- 0A            0940            ASL A
799C- AA            0950            TAX
799D- BD CF 7F      0960            LDA JMPTBL,X
79A0- A8            0970            TAY                 ;LOW BYTE
79A1- BD D0 7F      0980            LDA JMPTBL+1,X
79A4- AA            0990            TAX                 ;HIGH BYTE
79A5- 4C F7 9F      1000            JMP ENTCMD          ;DO COMMAND
```

***Listing 2:** SPOOL command sends sequential file directly from disk to printer.*

```
                    ;****
7E75- F0 31         3180 SPOOL      BEQ UNSPOOL         ;TURN OFF PRINTER & DISK
                    3190 ;****
7E77- 20 BB 79      3200            JSR PARCHK
7E7A- A5 D3         3210            LDA *SA             ;SEE IF PRINTER #
7E7C- D0 02         3220            BNE SPOOL1
7E7E- A9 04         3230            LDA #4
7E80- 29 1F         3240 SPOOL1     AND #%00011111
7E82- C9 04         3250            CMP #4
7E84- 90 71         3260            BCC SYNERR
7E86- 85 FB         3270            STA *TMP0
7E88- A9 02         3280            LDA #2
7E8A- 85 FD         3290            STA *TMP2
7E8C- 20 C8 7E      3300            JSR PARSE1
7E8F- AD 40 E8      3310            LDA $E840           ;SET ATN
7E92- 29 FB         3320            AND #$FB
7E94- 8D 40 E8      3330            STA $E840
7E97- A5 FB         3340            LDA *TMP0
7E99- 85 D4         3350            STA *FA             ;PRINTER TO LISTEN
7E9B- 20 D5 F0      3360            JSR LISTEN          ;LEAVE ATN HIGH
7E9E- 20 48 F1      3370            JSR RELATNLST       ;RELEASE ATN
7EA1- A9 00         3380            LDA #0
7EA3- 85 AF         3390            STA *$AF            ;RESTORE INPUT TO KEYBOARD
7EA5- 85 AE         3400            STA *OPNFILE        ;ZERO OPEN FILES
7EA7- 60            3410            RTS                 ;EXIT TO BASIC
                    3420
7EA8- A9 02         3430 UNSPOOL    LDA #2
7EAA- 85 AE         3440            STA *OPNFILE        ;OPEN FILES
7EAC- A9 04         3450            LDA #4
7EAE- 85 B0         3460            STA *$B0            ;CMD DEVICE
7EB0- A9 08         3470            LDA #8
7EB2- 85 AF         3480            STA *$AF            ;INPUT DEVICE
7EB4- 20 CC FF      3490            JSR CLRCHN          ;UNTALK & UNLISTEN
7EB7- 4C 3C 7C      3500            JMP AD78C6          ;CLOSE FILES
                    3510
7EBA- A9 00         3520 PARSEPRG   LDA #0              ;ZERO SECONDARY FOR PRG FI
7EBC- F0 02         3530            BEQ PAR0
7EBE- A9 02         3540 PARSESEQ   LDA #2              ;SECOND SECONDARY FOR SEQ
7EC0- 85 FD         3550 PAR0       STA *TMP2
```

(Continued on page 74)

Engineers

THE ZENITH MICROCOMPUTER We Can't Wait To Get Your Hands On It

High visibility, high growth...at Zenith Data Systems, it's exceptional. We're a young, dynamic company, developing the next generation of microcomputer and terminal systems. Our engineering teams are small and organized around product lines. To the engineer, that means involvement with the product from start to finish.

We are experiencing a rapid increase in sales. To meet this demand, we need engineers **now** for the following newly created positions.

COMPUTER DESIGN ENGINEERS

Several engineering positions available in small business computer systems, terminal systems, and Continuing Engineering. These individuals are responsible for product design and development. A BSEE, MSEE, or equivalent degree plus a minimum of 3 years microcomputer, terminal, or data communication systems design experience using 8/16-bit technology is required.

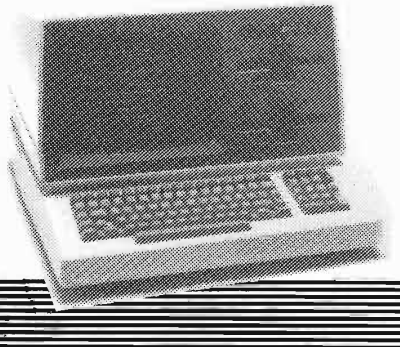
SOFTWARE DESIGN ENGINEERS

Several engineers needed for enhancement of operating systems for microcomputer products. Responsibilities include utility design, adaptation of existing products to microcomputer systems, and reviewing/specifying hardware interfaces. BSCS (or equivalent degree), demonstrated knowledge of assembly level language (8080/Z80, 68000/Z8000/8086) and experience with operating system development (preferably CP/M, HDOS, and UNIX) is required. Familiarity with microcomputer hardware is desirable.

Not only will you experience high visibility and excellent career growth, you also will enjoy the attractive and affordable quality of life in St. Joseph, Michigan. We're on Lake Michigan in the midst of clean air and beautiful countryside...yet only 90 minutes from Chicago. We also offer a competitive salary and excellent benefits including relocation assistance and the chance to work on your Master's Degree through an on-site University extension program.

For immediate consideration, call collect or send your resume to:

Kathy Kniola
616/982-3844



ZENITH | data systems

A Wholly Owned Subsidiary of Zenith Radio Corporation
Hilltop Road — Dept. MM8
St. Joseph, Michigan 49085
An Equal Opportunity Employer m/f

POKE, PEEK, and SYS addresses. If the number input is preceded by a dollar symbol, then the number is taken as hex and the decimal value for it is printed. If a decimal number is entered then the hex value for it is returned. The range for conversion is 0 to 65535 or $0000 to $FFFF.

Put the PET into Lower Case

Syntax: LOW

This command puts the PET into lower-case mode, the same as a POKE 59468,14.

Merge a Program with One in Memory

Syntax: MRG "program filename"
(, disk#)

This command will merge a BASIC program from the disk with one in memory. The lines of the program will be merged in just as if the program were typed in from the keyboard. Therefore, lines are merged between those in memory, if necessary, and duplicate lines in memory are replaced with the merged lines. The program is listed as it is merged. This merge uses a program file, not a sequential file, which is required by POWER's XEC command.

Pack a Program

Syntax: PAC

This command will remove remarks and waste spaces in a BASIC program. Be careful not to branch in a BASIC program to deleted remarks. This command is fooled easily, so keep a copy of the original in case the packing does not function properly.

Editorial note: There may be a bug in this command, so making a backup is especially important. It works most of the time, but occasionally fails.

Read a Sequential File

Syntax: VEW "seq filename"
(, disk#)

The VEW command will read a sequential file from the disk and print it to the screen. This command can be very handy for viewing data created by programs.

Give the Size of a Program

Syntax: SIZ
SIZ "program filename"
(, disk#)

Listing 2 *(Continued)*

```
7EC2- 20 E7 FF  3560 PARSE      JSR CLOSAL      ;CLOSE ALL FILES &  RESTOR
7EC5- 20 BB 79  3570            JSR PARCHK      ;SET UP FILENAME
7EC8- A6 D1     3580 PARSE1     LDX *FNLEN      ;SAVE FILENAME LENGTH
7ECA- F0 2B     3590            BEQ SYNERR
7ECC- 86 09     3600 PAR1       STX *TEMP
7ECE- A9 01     3610            LDA #1          ;OPEN1,8,15
7ED0- 85 D2     3620            STA *LF
7ED2- A9 0F     3630            LDA #15
7ED4- 85 D3     3640            STA *SA
7ED6- A9 00     3650            LDA #0
7ED8- 85 D1     3660            STA *FNLEN
7EDA- 20 63 F5  3670            JSR OPEN

7EDD- 20 CC FF  3680            JSR CLRCHN      ;RESET DEFAULT I/O
7EE0- A5 09     3690            LDA *TEMP       ;RESET FILENAME LENGTH
7EE2- 85 D1     3700            STA *FNLEN
7EE4- A9 02     3710            LDA #2          ;OPEN2,8,2,FILENAME
7EE6- 85 D2     3720            STA *LF
7EE8- A5 FD     3730            LDA *TMP2
7EEA- 85 D3     3740            STA *SA
7EEC- 20 63 F5  3750            JSR OPEN
7EEF- 20 28 7D  3760            JSR CHKERR      ;READ ERROR CHANNEL
7EF2- A2 02     3770            LDX #2
7EF4- 4C C6 FF  3780            JMP CHKIN
                3790
7EF7- A9 FF     3800 SYNERR     LDA #$FF
7EF9- 85 37     3810            STA *BASLIN+1
7EFB- 4C 00 BF  3820            JMP SYNERR1
```

***Listing 3:* YOU command allows loading a machine-language program at an address different from its original location.**

```
                3840 ;**
7EFE- F0 F7     3850 YOU        BEQ SYNERR
                3860 ;**
7F00- 20 72 7D  3870            JSR INPHEX      ;GET HEX ADDRESS
7F03- 20 F5 BE  3880            JSR CHKCOM
7F06- A9 00     3890            LDA #0
7F08- 85 9D     3900            STA *VERCK
7F0A- A5 5F     3910            LDA *FACC       ;LOAD ADDRESS
7F0C- A6 60     3920            LDX *FACC+1
7F0E- 85 FC     3930            STA *TMPO+1
7F10- 86 FB     3940            STX *TMPO
7F12- 20 BA 7E  3950            JSR PARSEPRG
7F15- 20 49 F4  3960            JSR PRTBER
7F18- 20 CF FF  3970            JSR INPUT       ;SKIP LOAD ADDRESS
7F1B- 20 CF FF  3980            JSR INPUT
                3990            IFE WHICH-$4030
                4000            JSR LOADIT+48
                4010            ***
                4020            IFN WHICH-$4030
7F1E- 20 8C F3  4030            JSR LOADIT+54
                4040            ***
7F21- 20 3C 7C  4050            JSR AD78C6
7F24- 20 2A 7C  4060            JSR CHKST
7F27- 4C FF B3  4070            JMP READY
```

***Listing 4:* DSK command allows change of disk drive device number.**

```
                4090 ;**
7F2A- 20 A2 F4  4100 DSK        JSR SCGBYT+3    ;NO COMMA CHECK
                4110 ;**
7F2D- 20 AD 79  4120            JSR DEVICE+3
7F30- 85 FD     4130            STA *TMP2       ;OLD
7F32- 20 AA 79  4140            JSR DEVICE
7F35- 85 FE     4150            STA *TMP2+1     ;NEW
7F37- A9 00     4160            LDA #0
7F39- 85 96     4170            STA *ST
7F3B- A5 FD     4180            LDA *TMP2
7F3D- 85 D4     4190            STA *FA
7F3F- 20 D5 F0  4200            JSR LISTEN
7F42- A9 6F     4210            LDA #$60+15
7F44- 20 43 F1  4220            JSR SNDSALST
7F47- A2 05     4230            LDX #5
7F49- BD 65 7F  4240 CHG005     LDA CHGMSG,X    ;SEND MESSAGE
7F4C- 20 9E F1  4250            JSR BASOUT

7F4F- CA        4260            DEX
7F50- 10 F7     4270            BPL CHG005
7F52- A9 20     4280            LDA #32
7F54- 20 5F 7F  4290            JSR CHG010
7F57- A9 40     4300            LDA #64
7F59- 20 5F 7F  4310            JSR CHG010
7F5C- 4C B9 F1  4320            JMP UNLSTN
                4330
7F5F- 18        4340 CHG010     CLC
7F60- 65 FE     4350            ADC *TMP2+1
7F62- 4C 9E F1  4360            JMP BASOUT
```

The SIZ command will give the size of a BASIC program in memory or any program on disk. The size of a program in memory is found by subtracting the end of the program location from the start of the program location. The size of a program on disk is found by counting the bytes in the file. The size is given in decimal and hex.

Spool a File from Disk to Printer

Syntax: SP "sequential filename"
[, disk#, printer#]
SP

This command will send a file directly from the disk to the printer. The PET can then perform other functions, such as editing a program or running a program (but with no access to the IEEE bus). The command is started with the command and a sequential file name. Power-Aid causes the PET to open the file, set the printer to listen, and then get off the IEEE bus (allowing the disk to talk directly to the printer). When the printer stops printing, enter SP with no file name to unlisten the printer, untalk the disk, and close the file.

Use the SP command to list a long program while you use the PET for something else. Create a file with

```
OPEN8,8,8,"0:TEMP,S,W":CMD8:LIST
PRINT#8:CLOSE8
```

Then spool this to the printer.

Find the Load Address

Syntax: STR "program filename"
[, disk#]

This command will give the load address of a program on the disk. The load address is found by reading the first two bytes of the file, which is the address where the program is loaded. The load address is given in decimal and hex. This command is particularly useful in determining where machine-language programs load.

Recover a Program after a NEW

Syntax: UNW

If, after a NEW command is entered, it is discovered that a program has not been saved, it can be recovered with this command.

Put the PET into Upper Case

Syntax: UP

This command puts the PET into upper-case mode, the same as a POKE 59468,12.

DOS Commands

Syntax: DC [, disk#]
DC "disk command" [, disk#]
DC "$0" [, disk#]
DL "program name" [, disk#]
EX "program name" [, disk#]

The DC command is used to read the error channel, send commands, and display the disk directory. The command alone will read the error channel and print it to the screen. The command followed by a disk command will send that command to the disk. The command followed by the dollar symbol will display the directory to the screen. The DL command will load a BASIC program from the disk. The EX command will load and run a BASIC program from the disk.

Change Disk Number

Syntax: DSK from#, to#

This command allows the disk device number to be software changed. This command performs the same function as the Change Unit Addr program on the Commodore Demo Disk.

Load at an Address You Specify

Syntax: YOU $/hex-address/,
"file name" [, disk#]

This command will force a file to load starting at the hex address you specify instead of where it would normally load.

Notes: POWER's FIX command disables Power-Aid so it must be re-enabled. The default disk number in commands is eight, and the default printer number in commands is four. The commands that print to the screen (File, Merge, View, Disk Directory) can be paused, held, or stopped. You may pause by holding the shift key down, and stop with the stop key. To hold the display, use the space bar on graphics keyboards and 6 on business keyboards. To continue the display use the ← key on graphics keyboards and 9 on business keyboards.

Note on Program Availability

The full listing for POWER-Aid is too long to include in MICRO. The program is available on disk from the ASM-TED Users' Group and the Toronto PET Users' Group. For 2040/4040 disk, send $10 (to cover disk, mailer, postage, and labor) and specify 'Power-Aid' disk to:

ATUG Disk Exchange
c/o Brent Anderson
200 S. Century
Rantoul, IL 61866

For 8050 disk, send $10 to:

ATUG Disk Exchange
c/o Baker Enterprises
15 Windsor Drive
Atco, NY 08004

Or send $10 for 4040 disk, $12 for 8050 disk to:

TORONTO PET Users' Group
c/o Chris Bennett
381 Laurence Avenue West
Toronto, Ontario
Canada M5M 1B9

The listings show selected portions of POWER-Aid. Listing 1 shows how new commands are added to POWER. Listings 2, 3, and 4 show the code for the added SPOOL, YOU, and DSK commands. These are assembled for the 4040 version of POWER.

F. Arthur Cochrane is a graduate from the University of South Carolina where he earned a Master of Science in electrical and computer engineering. He currently works in an electronic development group which uses PETs for data acquisition and control. His home system consists of a Commodore 4032 PET, Commodore 4040 disk, and Epson MX-80 F/T with Graftrax. Contact Mr. Cochrane at 1402 Sand Bar Ferry Rd., Beech Island, SC 29841.

SURCHANGE for OSI

by Kerry Lourash

This machine-language, search-and-change utility for OSI BASIC-in-ROM computers allows substitution of change strings longer than the original, and permits wild card characters in both search and change strings.

SURCHANGE
requires:
OSI machines
with BASIC-in-ROM

SURCHANGE searches for, displays, and changes code in BASIC programs. As many as seventy-one characters may be searched for and changed. Don't care characters are allowed in both search and change strings. The user may specify change strings shorter, equal to, or longer than the search string.

To avoid confusion, here are the definitions of some terms used in this article: *search string* refers to the characters SURCHANGE is told to look for. *Workspace string* is a set of characters in the BASIC program that match the search string. The *change string* is the set of characters that SURCHANGE POKEs into the BASIC program when it finds a match to the search string.

There are eight options, used singly or in pairs:

Default	Print line numbers of lines that contain workspace strings.
1. Print	Print line numbers plus workspace strings.
2. Stmt	Print line numbers plus the statements in which workspace strings are found.
3. Line	Print lines in which workspace strings are found.
4. Quote	Search only within quotes and REM statements (text).
Default	If option 4 is not chosen, search only outside of quotes and REMs (program).
5. Var	Search for occurrences of a BASIC variable (specified by the search string).
6. Change	Replace all workspace strings with the change string.

Don't care characters are allowed in both search and change strings. To il-

Listing 1

```
10          ; SURCHARGE
20          ;BY KERRY LOURASH
30          ;
40          ; ZERO PAGE
50          ;
60            BUF=$97       TEMP STORAGE FOR SEARCH CHAR.
70           BUFF=$13       START OF INPUT BUFFER
80          QFLAG=$9B       CHANGE OPTION FLAG
90          CHRCNT=$AB      # OF CHARS. IN CURRENT LINE
100           CLEN=$6C      LENGTH OF CHANGE STRING
110            DIF=$5D      CLEN MINUS SLEN
120         ORIGIN=$9A       START OF WORKSPACE INDEX
130          LFLAG=$9B       LINE OPTION FLAG
140         LINCNT=$5E       # OF SCREEN LINES USED
150          PFLAG=$6D       PRINT OPTION FLAG
160          POINT=$6F       POINTER TO BASIC WORKSPACE
170          QFLAG=$9C       QUOTE OPTION FLAG
180         SCNCNT=$F        # OF CHARS SINCE LAST CR/LF
190          SFLAG=$9E       STATEMENT OPTION FLAG
200            SLEN=$99      LENGTH OF SEARCH STRING
210            STAK=$9D      START OF SEARCH BUFFER IN STACK
220           START=$79       START OF BASIC WORKSPACE
230            TEXT=$6D       TEXT FLAG
240           VFLAG=$7D       VARIABLE OPTION FLAG
250         WPOINT=$AA        ADDRESS OF WORKSPACE STRING
260         YINDEX=$9F        TEMP STORAGE FOR POINT INDEX
270          YSAVE=$97        TEMP STORAGE FOR PRINT INDEX
280          ;
290          ; BROM ROUTINES
300          ;
310         DELETE=$D14F     DELETE CHARS FROM PROGRAM
320           BROM=$A2B4     BROM ROUTINES COPIED FROM STACK
330          CHAIN=DELETE+$5A  RECHAIN BASIC LINES
340         FILBUF=$A946     FILL BUFFER ROUTINE
350         INCHAR=$FFE6     INPUT ONE CHAR FROM KYBD.
360         LETTER=$A0B1     CHECK FOR LETTERS A-Z
370         LIFEED=$A86C     PRINT CR/LF
380         NUMBER=$00C5     CHECK FOR NUMBER 0-9
390         NUMPRT=$B95F     PRINT NUMBER IN A,X
400         OUTPUT=$A8E5     PRINT ONE CHARACTER
410         PUSHUP=DELETE+$35  MAKE ROOM FOR LINE
420         QUESTN=$A8F3     PRINT A QUESTION MARK
430          RESET=$A477     RESET BASIC POINTERS
440          SPACE=$A8F0     PRINT A SPACE
```

Listing 1 *(continued)*

```
450              TOGOUT=$A39D   TOGGLE VIDEO OUTPUT FLAG
460              TOKBUF=$A3A8   TOKENIZE LINE BUFFER
470              TOKTBL=$A084   START OF TOKEN TABLE
480              WARMST=$A274   ENTRY TO BASIC WARMSTART
490              ;
500              *=$7D00
510              ;
520 7D00 A200    OPTION LDX #0          SET PROMPT INDEX
530 7D02 20947F         JSR PROMPT     PRINT OPTION PROMPT
540 7D05 8598           STA CFLAG      ZERO FLAGS
550 7D07 859B           STA LFLAG
560 7D09 856D           STA PFLAG
570 7D0B 859C           STA QFLAG
580 7D0D 859E           STA SFLAG
590 7D0F 8570           STA VFLAG
600 7D11 855E           STA LINCNT
610 7D13 2046A9         JSR FILBUF     GET CHOICE OF OPTIONS
620 7D16 E8      OP     INX            AFTER FILBUF, X=$12
630 7D17 B500           LDA $0,X       EXAMINE BUFFER CONTENTS
640 7D19 F023           BEQ LOGIC      BRANCH IF DONE
650 7D1B 38             SEC            CONVERT ASCII TO NUMBER
660 7D1C E931           SBC #$31
670 7D1E A8             TAY            NUMBER TO Y REG.
680 7D1F D002           BNE OP1        SET CORRECT FLAG
690 7D21 C66D           DEC PFLAG
700 7D23 88      OP1    DEY
710 7D24 D002           BNE OP2
720 7D26 C69E           DEC SFLAG
730 7D28 88      OP2    DEY
740 7D29 D002           BNE OP3
750 7D2B C69B           DEC LFLAG
760 7D2D 88      OP3    DEY
770 7D2E D002           BNE OP4
780 7D30 C69C           DEC QFLAG
790 7D32 88      OP4    DEY
800 7D33 D002           BNE OP5
810 7D35 C670           DEC VFLAG
820 7D37 88      OP5    DEY
830 7D38 D0DC           BNE OP
840 7D3A C698           DEC CFLAG
850 7D3C D0D8           BNE OP         BRANCH ALWAYS
860 7D3E         ;
870 7D3E A698    LOGIC  LDX CFLAG      IS CHANGE FLAG SET?
880 7D40 F006           BEQ L1
890 7D42 859B           STA LFLAG      FORCE DEFAULT OPTION
900 7D44 856D           STA PFLAG
910 7D46 859E           STA SFLAG
920 7D48 A570    L1     LDA VFLAG      BOTH V & Q FLAGS SET?
930 7D4A 259C           AND QFLAG
940 7D4C F003           BEQ GETSUR
950 7D4E 20E3A8         JSR QUESTN     PRINT A QUESTION MARK
960 7D51         ;
970 7D51 A24C    GETSUR LDX #$4C
980 7D53 20947F         JSR PROMPT     PRINT SEARCH PROMPT
990 7D56 202E7F         JSR INPUT      GET SEARCH STRING
1000 7D59 A24E   STACK  LDX #$4E       SET STACK PTR TO $014E
1010 7D5B 9A            TXS
1020 7D5C AA            TAX            SLEN TO X REG.
1030 7D5D 8699          STX SLEN
1040 7D5F E8            INX
1050 7D60 B513   ST     LDA BUFF,X     PUSH SEARCH STRING
1060 7D62 48            PHA            ONTO STACK
1070 7D63 CA            DEX
1080 7D64 10FA          BPL ST
1090 7D66 BA            TSX            START OF SEARCH STRING
1100 7D67 869D          STX STAK       TO STAK
1100 7D67 869D          STX STAK       TO STAK
1110 7D69 A2FE          LDX #$FE       RESET STACK
1120 7D6B 9A            TXS
1130 7D6C        ;
1140 7D6C A598          LDA CFLAG
1150 7D6E F029          BEQ SEARCH
1160 7D70 A253   GETCNG LDX #$53
1170 7D72 20947F        JSR PROMPT     PRINT CHANGE PROMPT
1180 7D75 202E7F        JSR INPUT      PRINT & STORE CHANGE STRING
1190 7D78 856C          STA CLEN
1200 7D7A        ;
1210 7D7A A296          LDX #$96       MOVE BROM ROUTINES
1220 7D7C A089          LDY #$89       TO STACK
1230 7D7E BDB4A2 COPY   LDA BROM,X
```

(continued)

lustrate what a don't care character is, consider the following example:

SEARCH? YXXX

I'm using "X" for the don't care symbol; in the actual program, it is CTRL-G, the ASCII BEL character. This search string finds all strings of four characters starting with a "Y". For an example of don't care characters in a change string:

CHANGE? YXXX

This change string changes only the first letter of the workspace string. The last three letters remain the same.

Using SURCHANGE

SURCHANGE can be called by POKEing its starting address into the USR vector and typing X = USR(X). To avoid typing the USR command every time, you could insert the USR command in the program you are working on as line zero. Typing RUN would then call SURCHANGE.

First, SURCHANGE prints a list of options and a prompt to select options (OPTIONS?). Options are selected by typing a combination of digits (no commas). If you make a mistake, use the usual OSI backspace (shift O). You may terminate the line and start over with a shift P, although the prompt will not be repeated. RETURN signals the end of option selection. If this procedure seems familiar, it should; you're using the Fill-the-Buffer routine of OSI BASIC.

Next, the search prompt (SEARCH?) is printed. The FTB routine is used here, too. Don't care characters are input by typing CTRL-G. If you hit RETURN without an input when typing the search or change string, SURCHANGE prints the exit prompt. If you type a "Y", SURCHANGE exits to the immediate mode. Hitting any other key causes a jump to the start of SURCHANGE.

The change prompt (CHANGE?) appears if you've chosen the change option. Only the line numbers of changes will be printed when the change option is selected. If a line is made too long (longer than 71 characters), the graphics symbol $E9 is printed after the line number.

I have attempted to provide a paged display of SURCHANGE's output. It would be nice to be able to count the number of CR/LFs generated by the video routine to determine when the screen is full. So far, I haven't figured out how to accomplish this, short of writing a separate video routine. After a certain number of lines have been printed, SURCHANGE pauses. If the space bar is hit, the display continues. Any other key causes an exit to the immediate mode without an "OK" to scroll the screen. If you use the line print option (3), you can display lines and edit them (assuming you have an editor program).

Options

Default options are automatically selected if options 1-3 or option 4 is not selected. When the change option is chosen, SURCHANGE automatically selects the default display option. If options 4 and 5 are both selected, SURCHANGE prints a question mark in front of the search prompt, since it is unlikely the user would look for a variable in the text area of a program.

The default display option displays the line numbers of lines that contain workspace strings. The numbers are displayed with a single space separating them. If a number is printed more than once, more than one workspace string is present in the line. This option allows a very dense display and calls attention to multiple occurrences of a workspace string in a line.

Option 1 displays line numbers plus the workspace string. Due to the presence of don't care characters in a search string, the workspace string may not be identical to the search string. This option is handy when don't care characters are used. Also, option one emphasizes multiple occurrences of workspace strings in a line, although its display format is not as compact as the default option's.

The statement option (2) prints the line number and the statement in which the workspace string is found (a line may contain multiple statements). Colons found at the beginning and end of the statement are also printed. The presence or absence of colons indicates the statement's position in the line:

X = 3:—statement at start of line.
:X = 3:—statement in middle of line.
:X = 3 —statement at end of line.
X = 3 —statement is the entire line.

Listing 1 *(continued)*

```
1240 7D81 994E01          STA DELETE-1,Y
1250 7D84 CA              DEX
1260 7D85 E067            CPX #$67
1270 7D87 D002            BNE CP
1280 7D89 A25A            LDX #$5A
1290 7D8B 88        CP    DEY
1300 7D8C D0F0            BNE COPY
1310 7D8E A960            LDA #$60      INSERT RTS INSTRUCTIONS
1320 7D90 8D8001          STA DELETE+$31
1330 7D93 8DA801          STA DELETE+$59
1340 7D96 8DB801          STA DELETE+$69
1350 7D99           ;
1360 7D99 A579      SEARCH LDA START    BASIC WORKSPACE POINTER
1370 7D9B 856E            STA POINT     STOREDIN POINT, POINT+1
1380 7D9D A57A            LDA START+1
1390 7D9F 856F            STA POINT+1
1400 7DA1 A003      NEXLIN LDY #3       SKIP LINE POINTERS
1410 7DA3 849A            STY ORIGIN
1420 7DA5 A900            LDA #0        INITIALIZE TEXT FLAG
1430 7DA7 8560            STA TEXT
1440 7DA9 E69A      SETBUF INC ORIGIN
1450 7DAB A49A            LDY ORIGIN
1460 7DAD A69D            LDX STAK      SET STACK POINTER TO
1470 7DAF 9A              TXS           START OF SEARCH BUFFER
1480 7DB0 68        NEXBUF PLA          GET SEARCH CHAR.
1490 7DB1 F04D            BEQ MATCH     FOUND A MATCH?
1500 7DB3 C907            CMP #7        DON'T CARE CHAR?
1510 7DB5 D002            BNE STOBUF
1520 7DB7 B16E            LDA (POINT),Y
1530 7DB9 8597      STOBUF STA BUF      SAVE CHAR. IN BUF
1540 7DBB B16E      NEXBYT LDA (POINT),Y
1550 7DBD AA              TAX
1560 7DBE F01E            BEQ FIXLIN    END OF BASIC LINE?
1570 7DC0 E08E      REM   CPX #$8E      REM TOKEN?
1580 7DC2 F011            BEQ TOGGLE    YES, TOGGLE TEXT FLAG
1590 7DC4 E022      QUOTE CPX #'"
1600 7DC6 F00D            BEQ TOGGLE
1610 7DC8 A59C      CKTEXT LDA QFLAG    CHECK TEXT FLAG
1620 7DCA C560            CMP TEXT
1630 7DCC D0DB            BNE SETBUF
1640 7DCE E497      COMPAR CPX BUF      DO CHARS MATCH?
1650 7DD0 D0D7            BNE SETBUF
1660 7DD2 C8              INY           INCREMENT WORKSPACE INDEX
1670 7DD3 D0DB            BNE NEXBUF    BRANCH ALWAYS
1680 7DD5 A560      TOGGLE LDA TEXT     TOGGLE TEXT FLAG
1690 7DD7 49FF            EOR #$FF
1700 7DD9 8560            STA TEXT
1710 7DDB 4CCE7D          JMP COMPAR
1720 7DDE           ;
1730 7DDE A8        FIXLIN TAY          SET POINT TO NEXT LINE
1740 7DDF B16E            LDA (POINT),Y
1750 7DE1 AA              TAX
1760 7DE2 C8              INY
1770 7DE3 B16E            LDA (POINT),Y
1780 7DE5 866F            STX POINT
1790 7DE7 856F            STA POINT+1
1800 7DE9 D0B6            BNE NEXLIN    END OF PROGRAM?
1810 7DEB           ;
1820 7DEB A25A      END   LDX #$5A
1830 7DED 20947F          JSR PROMPT    PRINT EXIT PROMPT
1840 7DF0 20EBFF          JSR INCHAR    GET CHAR. FROM KYBD.
1850 7DF3 C959            CMP #'Y
1860 7DF5 F003            BEQ DONE
1870 7DF7 4C007D          JMP OPTION    LOOP TO START OF SURCHANGE
1880 7DFA 4C74A2    DONE  JMP WARMST    GOTO IMMEDIATE MODE
1890 7DFD 4C1C7F    RET   JMP RETURN
1900 7E00           ;
1910 7E00 88        MATCH DEY           SAVE WORKSPACE INDEX
1920 7E01 849F            STY YINDEX
1930 7E03 A2FE            LDX #$FE      RESET STACK
1940 7E05 9A              TXS
1950 7E06           ;
1960 7E06 A570      VARIBL LDA VFLAG    TEST VARIABLE FOUND
1970 7E08 F01C            BEQ LINE
1980 7E0A A49A            LDY ORIGIN    INDEX TO START OF STRING
1990 7E0C C004            CPY #4        FIRST CHAR. IN LINE?
2000 7E0E F006            BEQ VO
2010 7E10 88              DEY           GET PREVIOUS CHARACTER
2020 7E11 B16E            LDA (POINT),Y
```

Listing 1 *(continued)*

```
2030 7E13 20237F          JSR LEGAL     IS IT A ALPHANUMERIC CHAR?
2040 7E16 A49F    V0      LDY YINDEX    GET CHAR. IN FRONT OF STRING
2050 7E18 C8      V1      INY
2060 7E19 B16E            LDA (POINT),Y
2070 7E1B C924            CMP #'$
2080 7E1D F0DE            BEQ RET
2090 7E1F C928            CMP #'(
2100 7E21 F0DA            BEQ RET
2110 7E23 20237F          JSR LEGAL
2120 7E26         ;
2130 7E26 A002    LINE    LDY #2        GET 2-BYTE LINE #
2140 7E28 B16E            LDA (POINT),Y
2150 7E2A AA              TAX
2160 7E2B C8              INY
2170 7E2C B16E            LDA (POINT),Y
2180 7E2E 205EB9          JSR NUMPRT    CONVERT TO ASCII, PRINT
2190 7E31 E8      LIN     INX           PUT # OF DIGITS IN CHRCNT
2200 7E32 BD0001          LDA $0100,X
2210 7E35 D0FA            BNE LIN
2220 7E37 866B            STX CHRCNT
2230 7E39         ;
2240 7E39 A56D    PCHECK  LDA PFLAG
2250 7E3B D044            BNE FINI
2260 7E3D A59E    SCHECK  LDA SFLAG
2270 7E3F F02E            BEQ LCHECK
2280 7E41 A49F            LDY YINDEX    FIND END OF LINE
2290 7E43 B16E    S0      LDA (POINT),Y  OR TERMINATING COLON
2300 7E45 F013            BEQ S2
2310 7E47 C8              INY
2320 7E48 C922            CMP #'"
2330 7E4A D006            BNE S1
2340 7E4C A560            LDA TEXT      TOGGLE TEXT FLAG
2350 7E4E 49FF            EOR #$FF      IF QUOTE IS FOUND
2360 7E50 8560            STA TEXT
2370 7E52 2460    S1      BIT TEXT      LOOP IF IN TEXT
2380 7E54 30ED            BMI S0
2390 7E56 C93A            CMP #':
2400 7E58 D0F9            BNE S0
2410 7E5A 88      S2      DEY
2420 7E5B 849F            STY YINDEX    SAVE NEW END OF STRING
2430 7E5D A49A            LDY ORIGIN    LOOK BACK THRU LINE
2440 7E5F B16E    BACKWD  LDA (POINT),Y
2450 7E61 88              DEY
2460 7E62 C93A            CMP #':
2470 7E64 F004            BEQ BA
2480 7E66 C003            CPY #3        AT START OF LINE?
2490 7E68 D0F5            BNE BACKWD
2500 7E6A C8      BA      INY
2510 7E6B 849A            STY ORIGIN    SAVE NEW START OF LINE
2520 7E6D D012            BNE FINI
2530 7E6F A59B    LCHECK  LDA LFLAG
2540 7E71 F03E            BEQ CHANGE
2550 7E73 A49F            LDY YINDEX    FIND END OF LINE
2560 7E75 C8      LC      INY
2570 7E76 B16E            LDA (POINT),Y
2580 7E78 D0FB            BNE LC
2590 7E7A 88              DEY
2600 7E7B 849F            STY YINDEX    SAVE END OF LINE
2610 7E7D A004            LDY #4
2620 7E7F 849A            STY ORIGIN    START OF LINE IS ALWAYS 4
2630 7E81 20E0A8  FINI    JSR SPACE     PRINT SPACE
2640 7E84 204E7F          JSR PLINE     PRINT LINE
2650 7E87         ;
2660 7E87 E65F    COUNTR  INC LINCNT    CHECK # OF CHARS. IN LINE
2670 7E89 A56B            LDA CHRCNT    AND INCREMENT COUNT AS NEEDED
2680 7E8B C917            CMP #$17      ** C2P: CHANGE TO #$3F **
2690 7E8D 9008            BCC CHEC
2700 7E8F C92F            CMP #$2F      ** C2P: CHANGE TO #$7F **
2710 7E91 9002            BCC ADD1
2720 7E93 E65E            INC LINCNT
2730 7E95 E65F    ADD1    INC LINCNT
2740 7E97 A55E    CHEC    LDA LINCNT
2750 7E99 C90F            CMP #$F       COUNT <= 15 LINES?
2760 7E9B 900E            BCC CONT
2770 7E9D A900            LDA #0
2780 7E9F 855E            STA LINCNT
2790 7EA1 20EBFF          JSR INCHAR    GET KYBD. INPUT
2800 7EA4 C920            CMP #$20      IS INPUT A SPACE CHAR?
2810 7EA6 F003            BEQ CONT
```

(continued)

Option 2 allows the user to follow the use of a variable throughout a program or to examine all occurrences of any token (and its arguments) in a program. A statement is printed only once, even if it contains more than one workspace string. For example, in the statement A=A–3 the variable A occurs twice. If "A" were the search string, the statement would only be printed once.

The line option (3) lets the user see the entire line that contains the workspace string. This option displays a maximum amount of information, but also fills the screen rapidly. Like the statement option, the line option prints a line only once, even if it contains more than one workspace string. The line option can be used as an aid to edit individual lines. With SURCHANGE, find the lines to be edited; exit the SURCHANGE program; and either use an editor to change the lines, or retype them.

The quote option (4) searches the text portion of a BASIC program. Text includes PRINT statements, INPUT prompts, string variables, string DATA elements, and REM statements. Due to the structure of SURCHANGE, the initial quotation mark of a string is not considered to be part of the text. If the quote option is not chosen, SURCHANGE searches the program area outside of quotes and REMs.

The reason for defining two areas of search is that BASIC tokenizes its keywords (USR, POKE, NULL, etc.), unless the words are in REMs or quotes. A token is a one-byte code for a keyword. BASIC saves memory space and increases execution speed because it stores and reads only one byte, instead of a whole keyword. Thus, if you're searching for "ON", SURCHANGE needs to know whether you mean the word "ON" or the one-byte token for the keyword ON.

The variable option (5) helps search for a BASIC variable. In a normal search, looking for the variable "A" might find other variables such as A$, AB, A(X), etc. When the variable option is chosen, every variable found is tested to be sure it's not a subset of another variable.

The change option (6) enables modification of a BASIC program. Change strings may be shorter, equal in length, or longer than the search string. This is a powerful option and should always be used with caution. Unless changing text, SURCHANGE will tokenize the

change string before it is inserted in the program. Therefore, the change string may look deceptively longer or shorter than the search string when it is printed on the screen. For example, "RETURN" is one byte long when tokenized, while "A=6" is three bytes long. If "A=6" is substituted for RETURN, all lines changed will be two bytes longer.

If a line is longer than 71 bytes, it can still be LISTed, SAVEd, and even RUN. When you try to LOAD a long line, however, you'll find that the line is too long to fit into the input buffer. SURCHANGE prints a graphic character $E9 after a line number when the line becomes too long. Be sure to remember which lines are too long; they are identified only when the line is being changed, not during search operations.

Finding Your Way Around

SURCHANGE takes getting used to. I suggest you type in a ten- to twenty-line program and practice finding and changing things before you do any serious work. Here are a few tricks I've used!

To delete all non-text spaces in a program, select option 6. Type a space and a don't care character for the search string. Now, type a single don't care character for the change string. This gets rid of almost all single spaces and partially erases multiple spaces. Repeat as needed to erase all spaces. This strategy may work with other items you wish to delete.

When typing in a program, use a "%" or other seldom-used character to stand in for a phrase which is inserted by SURCHANGE after the program is completed. Of course, you must be careful not to make a line too long by the insertion.

Lines of up to 255 characters can be created with the change option. They use less memory space and run faster than normal lines. The big disadvantage of long lines is that they have to be saved and loaded in a machine-language format.

Changing SURCHANGE

C2/4P owners should change the COUNTR routine as noted in the listing. They may also want to eliminate the CR/LF between the two lines of options in the option prompt. The easiest method is to substitute two spaces ($20) for the $D, $A after

Listing 1 *(continued)*

```
2820 7EA8 4C7DA2          JMP $A27D     GOTO IMM. MODE; NO OK MESS.
2830 7EAB 206CA8  CONT    JSR LIFEED    PRINT CR/LF
2840 7EAE 4C1C7F          JMP RETURN    RESUME SEARCH
2850 7EB1                 ;
2860 7EB1 A598    CHANGE  LDA CFLAG
2870 7EB3 F067            BEQ RETURN
2880 7EB5 18              CLC           CALCULATE ABSOLUTE ADDRESS
2890 7EB6 A59A            LDA ORIGIN    OF START OF WORKSPACE STRING
2900 7EB8 656E            ADC POINT
2910 7EBA 85AA            STA WPOINT
2920 7EBC A46F            LDY POINT+1
2930 7EBE 9001            BCC CH
2940 7EC0 C8              INY
2950 7EC1 84AB    CH      STY WPOINT+1
2960 7EC3 38              SEC
2970 7EC4 A56C            LDA CLEN      FIND CLEN MINUS SLEN
2980 7EC6 E599            SBC SLEN
2990 7EC8 F02E            BEQ CEQUAL    IF CLEN = SLEN
3000 7ECA 900C            BCC MOVDWN
3010 7ECC 855D    MOVEUP  STA DIF       MAKE ROOM FOR LONGER STRING
3020 7ECE C65D            DEC DIF
3030 7ED0 208401          JSR PUSHUP
3040 7ED3 20857F          JSR REPLAC    INSERT CHANGE STRING
3050 7ED6 301A            BMI MV3
3060 7ED8 A47B    MOVDWN  LDY $7B       SET UP VARIABLES FOR DELETESUB
3070 7EDA 8471            STY $71
3080 7EDC A4AB            LDY WPOINT+1
3090 7EDE 8474            STY $74
3100 7EE0 48              PHA
3110 7EE1 38              SEC
3120 7EE2 A599            LDA SLEN
3130 7EE4 E56C            SBC CLEN
3140 7EE6 18              CLC
3150 7EE7 65AA            ADC WPOINT
3160 7EE9 9001            BCC MV2
3170 7EEB C8              INY
3180 7EEC 8472    MV2     STY $72
3190 7EEE 68              PLA
3200 7EEF 204F01          JSR DELETE    ERASE XTRA CHARS. FROM PROGRAM
3210 7EF2 2077A4  MV3     JSR RESET     RESET BASIC POINTERS
3220 7EF5 20A901          JSR CHAIN     RECHAIN LINE POINTERS
3230 7EF8 20857F  CEQUAL  JSR REPLAC
3240 7EFB 209DA3  LONG    JSR TOGOUT    CHECK FOR LONG LINE
3250 7EFE A0FF            LDY #$FF
3260 7F00 849F            STY YINDEX
3270 7F02 A004            LDY #4
3280 7F04 20507F          JSR PLINE+2
3290 7F07 209DA3          JSR TOGOUT
3300 7F0A 18              CLC
3310 7F0B A56C            LDA CLEN      FIND NEW END OF STRING
3320 7F0D 659A            ADC ORIGIN
3330 7F0F 859F            STA YINDEX
3340 7F11 A56B            LDA CHRCNT    IS LINE TOO LONG?
3350 7F13 C947            CMP #$47
3360 7F15 9005            BCC RETURN
3370 7F17 A9E9    TOOLNG  LDA #$E9      PRINT GRAPHIC CHAR.
3380 7F19 20E5AB          JSR OUTPUT
3390 7F1C A49F    RETURN  LDY YINDEX
3400 7F1E 849A            STY ORIGIN
3410 7F20 4CA97D          JMP SETBUF    RESUME SEARCH
3420 7F23                 ;
3430 7F23 20C500  LEGAL   JSR NUMBER    IS CHAR =0-9?
3440 7F26 90F4            BCC RETURN
3450 7F28 2081AD          JSR LETTER    IS CHAR =A-Z?
3460 7F2B B0EF            BCS RETURN
3470 7F2D 60              RTS
3480 7F2E                 ;
3490 7F2E 850E    INPUT   STA SCNCNT    ZERO VIDEO CHAR COUNTER
3500 7F30 2046A9          JSR FILBUF    PRINT AND STORE INPUT
3510 7F33 88              DEY           Y=#$FF
3520 7F34 C8      LILOOK  INY           COUNT # OF CHARS. IN INPUT
3530 7F35 B91300          LDA BUFF,Y
3540 7F38 D0FA            BNE LILOOK
3550 7F3A 88      TOK12F  DEY
3560 7F3B 1003            BPL TK0
3570 7F3D 4CE87D          JMP END       IF NULL INPUT
3580 7F40 98      TK0     TYA           SHOULD STRING BE TOKENIZED?
```

Listing 1 *(continued)*

```
3590 7F41 A49C          LDY QFLAG
3600 7F43 D008          BNE RTN
3610 7F45 E8            INX
3620 7F46 20ABA3        JSR TOKBUF      TOKENIZE STRING
3630 7F49 98            TYA             FIND LENGTH OF STRING
3640 7F4A 38            SEC
3650 7F4B E906          SBC #6
3660 7F4D 60      RTN   RTS
3670 7F4E               ;
3680 7F4E A49A    PLINE LDY ORIGIN      PRINT WORKSPACE STRING
3690 7F50 8497    PO    STY YSAVE
3700 7F52 B16E          LDA (POINT),Y
3710 7F54 F0F7          BEQ RTN         END OF LINE?
3720 7F56 101E          BPL PRINT       BRANCH IF NOT A TOKEN
3730 7F58 38      TOKEN SEC             FIND KEYWORD IN TABLE
3740 7F59 E97F          SBC #$7F
3750 7F5B AA            TAX
3760 7F5C A0FF          LDY #$FF
3770 7F5E CA      TO    DEX
3780 7F5F F008          BEQ T2
3790 7F61 C8      T1    INY             PRINT KEYWORD
3800 7F62 B984A0        LDA TOKTBL,Y
3810 7F65 10FA          BPL T1
3820 7F67 30F5          BMI TO
3830 7F69 C8      T2    INY
3840 7F6A B984A0        LDA TOKTBL,Y
3850 7F6D 3007          BMI PRINT       PRINT LAST CHAR. IN KYWORD
3860 7F6F E66B          INC CHRCNT
3870 7F71 20E5A8        JSR OUTPUT
3880 7F74 D0F3          BNE T2
3890 7F76 297F    PRINT AND #$7F        ZERO HI BIT
3900 7F78 20E5A8        JSR OUTPUT      PRINT CHARACTER
3910 7F7B E66B          INC CHRCNT
3920 7F7D A497          LDY YSAVE       DONE PRINTING LINE?
3930 7F7F C49F          CPY YINDEX
3940 7F81 C8            INY
3950 7F82 90CC          BCC PO
3960 7F84 60            RTS
3970 7F85               ;
3980 7F85 A46C    REPLAC LDY CLEN       INSERT CHANGE STRING
3990 7F87 B91300  RE0   LDA BUFF,Y
4000 7F8A C907          CMP #7          DON'T CARE CHAR?
4010 7F8C F002          BEQ RE1
4020 7F8E 91AA          STA (WPOINT),Y
4030 7F90 88      RE1   DEY
4040 7F91 10F4          BPL RE0         BRANCH ALWAYS
4050 7F93 60            RTS
4060 7F94               ;
4070 7F94 BDA07F  PROMPT LDA TABL,X     PRINT A MESSAGE
4080 7F97 E8            INX
4090 7F98 C60E          DEC SCNCNT      AVOID AUTO CR/LF
4100 7F9A 20E5A8        JSR OUTPUT      PRINT ONE CHARACTER
4110 7F9D D0F5          BNE PROMPT      LOOP IF CHAR NOT A NULL
4120 7F9F 60            RTS
4130 7FA0               ;
4140 7FA0               TABL
4150 7FA0 0D            .BYTE $D,$A,'SEARCH '
4150 7FA1 0A
4150 7FA2 53
4150 7FA3 45
4150 7FA4 41
4150 7FA5 52
4150 7FA6 43
4150 7FA7 48
4150 7FA8 20
4160 7FA9 4F            .BYTE 'OPTIONS:',$D,$A
4160 7FAA 50
4160 7FAB 54
4160 7FAC 49
4160 7FAD 4F
4160 7FAE 4E
4160 7FAF 53
4160 7FB0 3A
4160 7FB1 0D
4160 7FB2 0A
4170 7FB3 20            .BYTE ' 1-PRINT 2-STMT'
4170 7FB4 31
4170 7FB5 2D
4170 7FB6 50
```

(continued)

"3-LINE" in TABL at the end of the program.

If you wish to examine the BASIC-in-ROM routines copied to the stack, or if you must move them to another location, simply change the DELETE label to the start of the new location.

SURCHANGE is relocatable from object code with the exception of references to the prompt table (TABL). All references to TABL should be adjusted to conform to its new location.

How SURCHANGE Works

SURCHANGE occupies three pages of RAM and uses part of the stack for BASIC-in-ROM routines and the search buffer. It wipes out the NMI and IRQ vectors. To conserve zero page space for other accessory programs, SURCHANGE uses only zero page addresses normally used by BASIC. The change buffer is located in the line buffer ($13-5A).

To start, OPTION prints a list of options and the option prompt. The option flags are zeroed and FILBUF is called to find out what options are desired. When the options have been specified, their respective flags are set. LOGIC selects the default print option if the change flag is set, and prints a question mark in front of the search prompt if both the variable and quote flags are set.

GETSUR prints the search prompt and calls INPUT. INPUT zeros the video character counter ($E) so a full 71-character line can be typed without a premature CR/LF. FILBUF is called again to store and print the search string. After the search string is typed in, the number of characters in the string is counted. If no string has been input, the routine goes to END to see if the user wishes to start over. If the search is to be conducted within quotes, the tokenize-the-buffer routine (TOKBUF) is skipped. The number of characters in the string is returned in the A register.

INPUT returns to STACK, where the stack pointer is set to $014E and the length of the search string is stored in SLEN. The search string is pushed onto the stack and the stack pointer position saved in STAK. The stack pointer is then reset to the top of the stack.

If the change option has been selected, GETCNG prints the change prompt and INPUT is called to get the

change string. When INPUT returns, the length of the change string is stored in CLEN.

COPY transfers BASIC-in-ROM routines for inserting, deleting, and rechaining BASIC lines to the stack, and inserts RTS instructions to make them subroutines.

The Search Begins

The start-of-BASIC workspace pointer is transferred to SURCHANGE's workspace pointer (POINT). NEXLIN sets the Y register to index the start of the BASIC line, and TEXT, the quote status flag, is cleared. ORIGIN is initialized to the start of the line. The stack pointer is set to the start of the search buffer. A character is pulled from the stack. Naturally, the contents of the stack are not altered by this operation, and SURCHANGE can re-examine the search buffer any number of times. If the character is a null, SURCHANGE has found a match to the search string and goes to the MATCH routine. If it is a don't care character, the next character in the BASIC workspace is stored in BUF. Later, when the workspace character is compared to BUF, the two will match. If the search character is not a null or don't care byte, it's stored in BUF.

NEXBYT tests the next character in the workspace. If the workspace character is a null, the end of the BASIC line has been reached. The routine branches to FIXLIN to reset POINT to the next line or to exit, if at the end of the program. If the workspace character is a REM token or a quotation mark, the TEXT flag is toggled. This means if TEXT is zero, it's changed to #$FF, and *vice versa*. If TEXT is not equal to the quote option flag, SURCHANGE loops back to SETBUF. Finally, at COMPAR, the search character is compared to the workspace character. If the two are identical, the next search character is pulled from the stack and the NEXBUF loop is done again. If the characters don't match, the stack pointer is reset to the start of the search buffer, the workspace counter (ORIGIN) is incremented, and SURCHANGE starts looking for a workspace string again.

FIXLIN, as mentioned before, transfers the BASIC next-line pointer to POINT. If the high byte of the pointer is zero, the end of the BASIC program has been reached. The stack pointer is set to the top of the stack, "EXIT?" is printed, and SURCHANGE waits for an input. At this point, the user can hit Y and exit to the BASIC immediate mode

Listing 1 *(continued)*

```
4170 7FB7 52
4170 7FB8 49
4170 7FB9 4E
4170 7FBA 54
4170 7FBB 20
4170 7FBC 32
4170 7FBD 2D
4170 7FBE 53
4170 7FBF 54
4170 7FC0 4D
4170 7FC1 54
4180 7FC2 20        .BYTE ' 3-LINE',$D,$A
4180 7FC3 33
4180 7FC4 2D
4180 7FC5 4C
4180 7FC6 49
4180 7FC7 4E
4180 7FC8 45
4180 7FC9 0D
4180 7FCA 0A
4190 7FCB 20        .BYTE ' 4-QUOTE 5-VAR 6-'
4190 7FCC 34
4190 7FCD 2D
4190 7FCE 51
4190 7FCF 55
4190 7FD0 4F
4190 7FD1 54
4190 7FD2 45
4190 7FD3 20
4190 7FD4 35
4190 7FD5 2D
4190 7FD6 56
4190 7FD7 41
4190 7FD8 52
4190 7FD9 20
4190 7FDA 36
4190 7FDB 2D
4200 7FDC 43        .BYTE 'CHANGE',$D,$A
4200 7FDD 48
4200 7FDE 41
4200 7FDF 4E
4200 7FE0 47
4200 7FE1 45
4200 7FE2 0D
4200 7FE3 0A
4210 7FE4 4F        .BYTE 'OPTIONS',0
4210 7FE5 50
4210 7FE6 54
4210 7FE7 49
4210 7FE8 4F
4210 7FE9 4E
4210 7FEA 53
4210 7FEB 00
4220 7FEC 53        .BYTE 'SEARCH',0
4220 7FED 45
4220 7FEE 41
4220 7FEF 52
4220 7FF0 43
4220 7FF1 48
4220 7FF2 00
4230 7FF3 43        .BYTE 'CHANGE',0
4230 7FF4 48
4230 7FF5 41
4230 7FF6 4E
4230 7FF7 47
4230 7FF8 45
4230 7FF9 00
4240 7FFA 45        .BYTE 'EXIT?',0
4240 7FFB 58
4240 7FFC 49
4240 7FFD 54
4240 7FFE 3F
4240 7FFF 00
```

or hit any other key to rerun SURCHANGE.

A Match is Found

If a match to the search string is found, the workspace index (Y) to POINT is stored in YINDEX. The stack pointer is set to the top of the stack.

If VFLAG is set, VARIBL tests the characters adjacent to the workspace string to see if the string is a subset of another variable. If the correct variable has not been found, LEGAL jumps back into the search loop.

LINE finds the current line number in the workspace and prints it. It also counts the number of digits in the line number for later use in the COUNTR or LONG routines.

PCHECK prints a space and the workspace string if the print flag is set.

SCHECK finds the terminating colon of the statement or the end of the line. BACKWD finds the start of the statement or the start of the line. I was strapped for space here; I didn't include a check in BACKWD to be sure a colon is really a statement separator and not part of a string.

LCHECK finds the start and end of the line. The start is easy; always the fourth byte from the beginning of the line. FINI prints a space to separate line number and line, and then PLINE prints all or part of the line and counts the characters in the line.

COUNTR looks at the number of characters in the line just printed and decides whether LINENT, the line counter, shall be incremented by one, two, or three. CHEC decides if enough lines have been printed. If so, it calls INCHAR, which waits for a keystroke. Any other key causes an exit to the immediate mode, without the "OK" message.

Changing Things Around

CHANGE tests CFLAG and, if it is set, subtracts the length of the search string (SLEN) from the length of the change string (CLEN). If the two are equal, CHANGE goes directly to CEQUAL, where the change string replaces the workspace string. If CLEN is longer than SLEN, MOVEUP calls PUSHUP, a routine copied from ROM. PUSHUP makes room in the BASIC workspace for the longer change string. REPLAC is called to insert the change string into the BASIC program. LONG tests the new line length to see if it's longer than 71 characters. A graphics character $E9 is printed after the line number if the line is too long.

If CLEN is less than SLEN, CHANGE branches to MOVDWN. Part of the BASIC-in-ROM line delete routine is paraphrased in MOVDWN, then DELETE is called to move the BASIC lines down and delete the extra bytes in the program. REPLAC is called to insert the change string. CHAIN rechains the BASIC line pointers. RETURN resets the BASIC workspace index (ORIGIN) and jumps back into the search loop.

Developing SURCHANGE was a real challenge. Many thanks to Earl Morris for advice and for finding the bugs in the program.

Kerry Lourash may be contacted at 1220 North Dennis, Decatur, Illinois 62522.

OSI ✻NEW✻ OSI

FROM THE PRETZELLAND ARCADE

HUMANOID DEFENDER

AS DEFENDER OF THE HUMANOID COLONIES, YOU'VE GOT TO STOP THE ALIEN LANDERS THAT ARE TRYING TO PICK UP AND MUTATE THE HUMANOIDS. IF A LANDER PICKS UP A HUMANOID, YOU HAVE TO BLAST THE LANDER, THEN CATCH THE HUMANOID IN MID-AIR AND LOWER IT SAFELY TO THE GROUND FOR A BONUS! EVERY NOW AND THEN, A BAITER SHIP APPEARS OUT OF HYPERSPACE TO KEEP THINGS INTERESTING! WITH COLOR AND LOTS OF SOUND! 8K CASSETTE.....$9.95
SPECIFY YOUR SYSTEM!

-ASTRO BLASTER-

ASSIGNMENT: CLEAR THE AREA OF HAZARDOUS ASTEROIDS WHICH ARE DRIFTING IN FROM DEEP SPACE, BY BLASTING THEM INTO RUBBLE. BE CAREFUL BECAUSE THE LARGE ONES SPLIT INTO MANY SMALLER ONES WHEN HIT, WHICH FLY OFF IN ALL DIRECTIONS! JUST WHEN YOU THINK YOU'VE BLASTED THEM ALL, MORE APPEAR!
ASTRO BLASTER by JOHN WILSON IS A **MACHINE CODE** OSI VERSION OF ONE OF THE MOST POPULAR ARCADE GAMES OF ALL TIMES! THE ACTION IS VERY SMOOTH AND THE ASTEROIDS ARE THE BEST LOOKING EVER ON OSI- NO LITTLE CIRCLES HERE! AVAILABLE FOR BOTH C1P AND ***C4P*** PLEASE SPECIFY YOUR SYSTEM. 8K CASSETTE $9.95

SEND $1.00 FOR OUR PHOTO-ILLUSTRATED CATALOG & GET A $1.50 CREDIT ON YOUR FIRST ORDER! CASSETTES ONLY!

Pretzelland Software
2005 D WHITTAKER RD.
YPSILANTI, MI. 48197

maxell

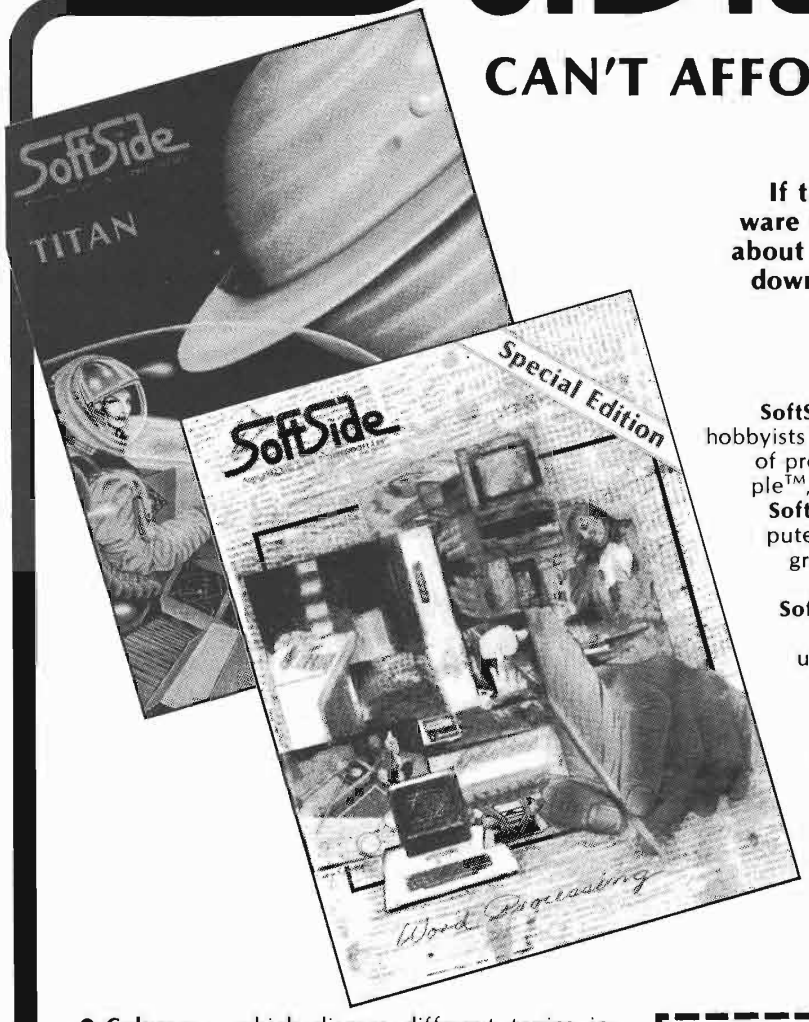
the top of the line in floppy diskettes

Write for free catalog with more than 400 fantastic values for all your word and data processing needs. Outside USA, enclose $1.00.

ABM PRODUCTS
8868 CLAIREMONT MESA BLVD.
SAN DIEGO, CALIFORNIA 92123

Toll Free 800-854-1555 Orders Only
For information or California Orders
(714) 268-3537

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 _____

ON ERROR GOTO for OSI ROM BASIC

by Earl Morris and Kerry Lourash

This run-time utility allows you to trap certain non-fatal errors and continue your program execution. You can also print out full error descriptions.

ON ERR
requires:
OSI C1P or 540-based computer

When OSI ROM BASIC encounters an error, program execution is halted and the screen displays the dreaded

? S* ERROR IN LINE xx

where the * is a graphics character rather than the correct letter. The following programs add an "ON ERR GOTO" function to your machine so that errors are detected and a jump is made to program line 50000. The line number where the error occurred is stored in the variable XX and the type of error is stored in X. At line 50000, the programmer can print out the expanded error message, fix the error, or jump back to the program. As an added bonus, the graphics character in the error message is converted to the correct alphabetic letter.

As an example, consider the program

```
10 INPUT "NUMBER"; A
20 PRINT:PRINT 1/A
30 GOTO 10
```

If a zero is input, the program will halt with a divide-by-zero error in line 20. With the error trap program in place, the following can be added:

```
50000 PRINT: IF XX < > 20 THEN END
50010 PRINT:PRINT "CAN'T DIVIDE
  BY ZERO — TRY AGAIN"
50020 GOTO 10
```

If an error occurs in line 20, the error trap program will print a message and continue program execution. Other errors will still end the program. The error trap resets the stack, effectively clearing all loops and subroutines. The jump back to the main program cannot enter within a FOR-NEXT loop or go directly to a subroutine.

Two versions of the ON ERR routine are listed: 1P and 540. Use the version appropriate for your machine. The method used to detect errors is different for each type of computer. The 1P version uses the output vector on page two. On every carriage return, the ON ERR program searches the stack to determine which routine is writing to the screen. If an $A252 is found on the stack, then the error routine is outputting and the ON ERRor program is triggered.

```
                          ;    ************************************
                          ;    * ON ERROR routine for 540 video *
                          ;    ************************************
                          ;
                          ;       Goes to line 50000 on error
                          ;       with line number in XX.
                          ;
                          ;       set-up  $0004=$40  $0005=$02
                          ;
1090 0240                 *=$0240
1100 0240 48              PHA
1110 0241 AD40D7          LDA $D740         READ SCREEN
1120 0244 C93F            CMP #$3F          QUESTION MARK ?
1130 0246 F004            BEQ J1            IF YES THEN ERROR OCCURED
1140 0248 68        J2    PLA               NORMAL MESSAGE OUTPUT
1150 0249 4CC3A8          JMP $A8C3         MESSAGE PRINTER
1160 024C AD42D7    J1    LDA $D742         FIX GRAPHICS CHARACTER
1170 024F 297F            AND #$7F          REMOVE HIGH BIT
1180 0251 8D42D7          STA $D742         STORE ON SCREEN
1190 0254 A588            LDA $88
1200 0256 C9FF            CMP #$FF          IMMED MODE ?
1210 0258 F0EE            BEQ J2            IF YES- GO TO BASIC
1220 025A 68              PLA
1230 025B A487            LDY $87           CURRENT LINE #
1240 025D 84AE            STY $AE           TO $AD,$AE
1250 025F A588            LDA $88
1260 0261 85AD            STA $AD
1270 0263 A290            LDX #$90
1280 0265 38              SEC
1290 0266 20E8B7          JSR $B7E8         CONVERT HEX TO FLOATING
1300 0269 A900            LDA #$00
1310 026B 855E            STA $5E
1320 026D 855F            STA $5F
1330 026F A958            LDA #$58          $58 = "X"
1340 0271 8593            STA $93
1350 0273 8594            STA $94
1360 0275 2049AD          JSR $AD49         FIND OR CREATE XX VARIABLE
1370 0278 8597            STA $97
1380 027A 8498            STY $98
1390 027C 2074B7          JSR $B774         PUT VALUE INTO XX
1400 027F A950            LDA #$50          HEX  50000 INTO $11,$12
1410 0281 8511            STA $11
1420 0283 A9C3            LDA #$C3
1430 0285 8512            STA $12
1440 0287 2032A4          JSR $A432         FIND LINE 50000
1450 028A 9006            BCC J3
1460 028C 20D9A6          JSR $A6D9         POINT TO LINE 50000
1470 028F 4CC2A5          JMP $A5C2         BASIC EXECUTION LOOP
1480 0292 A992      J3    LDA #$92          NO LINE 50000- PRINT "OK"
1490 0294 A0A1            LDY #$A1
1500 0296 4CC3A8          JMP $A8C3         MESSAGE PRINTER
```

Machines other than the 1P do not have the output vector in RAM, and must use a different hook into BASIC. The ON ERR program hooks into the OK message printer at $0003. The routine looks for the "?" which appears above the OK whenever an error occurs. A disadvantage of this hook is that the normal error message has already been printed and the type of error is no longer in memory. Thus the 540 version stores a value in XX (line number) but not in X (error type).

In both programs, after an error is detected, location $88 is inspected. If it contains a $FF, the computer is in the immediate mode and the ON ERRor routine is bypassed. Then the normal error message (corrected) is printed. If you wish to use ON ERRor in the immediate mode, change the following location:

1P — Change $0243 from $4C to $00
540 — Change $0259 from $EE to $00

The variable XX will contain 65xxx as a line number if the error occurs in the immediate mode.

If the computer is not in immediate mode, or if the above patch is made, the current line number is converted to

```
1000              ;  ***********************************
1010              ;  *  ON ERROR routine. 1P version    *
1020              ;  ***********************************
1030              ;
1040              ;   Goes to line 50000 on error with
1050              ;   line number in XX, error type in X.
1060              ;
1070              ;   Set up $021A=$22,  $021B=$02
1080              ;
1090 0222               *=$0222
1100              ;
1110 0222 C90D          CMP  #$0D      ;OUTPUT=CR?
1120 0224 D015          BNE  BYE       ;NO. EXIT
1130 0226 8A            TXA            ;SAVE X REGISTER
1140 0227 48            PHA
1150 0228 BA            TSX            ;GET STACK POINTER
1160 0229 BD0601        LDA  $106,X    ;IS CALLING ADDRESS
1170 022C C952          CMP  #$52      ;$A252 ?
1180 022E D007          BNE  A1
1190 0230 BD0701        LDA  $107,X
1200 0233 C9A2          CMP  #$A2
1210 0235 F007          BEQ  ERRTRP    ;YES. TO ERR TRAP
1220 0237 68       A1   PLA            ;RESTORE X REG.
1230 0238 AA            TAX
1240 0239 A90D          LDA  #$0D      ;RESTORE A-REG.
1250 023B 4C69FF   BYE  JMP  $FF69     ;GOTO REG. OUTPUT
1260              ;
1270 023E A588     ERRTRP LDA  $88     ;IF IN IMM. MODE
1280 0240 C9FF          CMP  #$FF      ;PRINT ERR MESSAGE
1290 0242 F04C          BEQ  ERROR
1300                                   ;STORE CURRENT LINE # IN XX
1310 0244 A487          LDY  $87       ;CURRENT LINE #
1320 0246 85AD          STA  $AD       ;TO F.P.A
1330 0248 84AE          STY  $AE
1340 024A A290          LDX  #$90
1350 024C 38            SEC
1360 024D 20E8B7        JSR  $B7E8
1370 0250 A900          LDA  #0        ;FIND OR CREATE XX
1380 0252 855E          STA  $5E
1390 0254 855F          STA  $5F
1400 0256 A958          LDA  #$58
1410 0258 8593          STA  $93
1420 025A 8594          STA  $94
1430 025C 2049AD        JSR  $AD49
1440 025F 8597          STA  $97       ;STORE F.P.A IN XX
1450 0261 8498          STY  $98
1460 0263 2074B7        JSR  $B774
1470                                   ;STORE ERROR #/2 IN X
1480 0266 68            PLA            ;PULL ERROR #
1490 0267 48            PHA
1500 0268 4A            LSR  A         ;HALVE IT
1510 0269 A8            TAY            ;STORE ERR # IN F.P.A
1520 026A A900          LDA  #0
1530 026C 20C1AF        JSR  $AFC1



1540 026F A900          LDA  #0        ;FIND OR CREATE X
1550 0271 8594          STA  $94
1560 0273 2049AD        JSR  $AD49
1570 0276 8597          STA  $97       ;STORE F.P.A IN X
1580 0278 8498          STY  $98
1590 027A 2074B7        JSR  $B774
1600                                   ;FIND LINE 50000
1610 027D A950          LDA  #$50      ;HEX 50000 IN $11,12
1620 027F 8511          STA  $11
1630 0281 A9C3          LDA  #$C3
1640 0283 8512          STA  $12
1650 0285 2032A4        JSR  $A432
1660 0288 9006          BCC  ERROR     ;BRANCH IF NO LINE
1670 028A 20D9A6        JSR  $A6D9     ;SET PARSER AT 50000
1680 028D 4CC2A5        JMP  $A5C2     ;BASIC EXEC. LOOP
1690                                   ;PRINT ERR MESSAGE
1700 0290 68       ERROR PLA           ;PULL ERR INDEX
1710 0291 AA            TAX
1720 0292 20E3A8        JSR  $A8E3     ;PRINT '?'
1730 0295 BD64A1        LDA  $A164,X   ;FIRST LETTER
1740 0298 20E5A8        JSR  $A8E5     ;PRINT IT
1750 029B BD65A1        LDA  $A165,X   ;SECOND LETTER
1760 029E 297F          AND  #$7F      ;ZERO HI BIT
1770 02A0 4C5FA2        JMP  $A25F     ;TO REG. ERR ROUTINE
```


O.S.I. C1P, C4P & C8P
TRS-80 Models I & III

The Room's of Cygnes IV
You are in a room with walls placed randomly throughout. There are three to ten ROBOTS bent on destroying you with laser fire or by touching you. You must destroy all the robots in a room to advance a level. The higher the level the faster the robots. WATCH-OUT the walls are electrified!! There are two skill levels.
O.S.I. C4P & C8P - Color, Sound & Joysticks 8K Cassette - $9.95
Trs-80 I & III Sound both 32K cassette and 32K disk version- $12.95

Murder Mansion - Adventure #1
You and seven other people are exploring a three story house for a treasure. WATCH-OUT!! Someone has decided to get a little greedy and is killing the others. You could be NEXT!!
Trs-80 version has a graphic representation of each room!!!
O.S.I. C1P, C4P & C8P - 8K Cassette - $12.95
Trs-80 I & III (Graphics) 32K Cassette and 48K Disk version - $16.95

Blaster Attack
This is a two player game where you have to protect your country from overhead spy-craft. You and your opponent must shoot down as many enemy craft with your LASER-BASE before your time runs out. Excellent sound and graphics. You can play the computer or a human opponent. You both will use the same playing field at the same time!!!
O.S.I. C4P & C8P - Sound 8K Cassette - $7.95

Novastar
You are a lone warrior on a seek and destroy mission. You've been sent to ALDEBERON to destroy 4 CYCLON fuel stations which are protected by a CYCLON fighter squadron. You will have 5 oldstyle nuclear warheads to destroy the stations. At the same time you will have to dogfight the CYCLON fighters!!!
O.S.I. C4P & C8P - Color & Sound 8K Cassette - $6.95

Bunkers
This is a two player game in which you must destroy 5 of your opponents bunkers by shooting over a rugged mountain. You control your muzzle velocity and barrel angle. Great COLOR graphics and sound.
O.S.I. C4P & C8P - Color & Sound 8K Cassette - $7.95

Send $1.00 for catalog

COMP-U-GAMER SOFTWARE
P.O. Box 802
Nevada, Missouri 64772

Table 1

Index	Error Message
0	Next Without For
1	Syntax Error
2	Return Without Gosub
3	Out Of Data
4	Function Call — argument out of range
5	Overflow
6	Out Of Memory
7	Undefined Statement GOTO non-existent line
8	Bad Subscript Subscript greater than dimension
9	Double Dimension
10	Division By Zero
11	Illegal Direct Can't use in immediate mode
12	Type Mismatch
13	Long String
14	String Temporaries
15	Continue Error
16	Undefined Function

floating point and stored in the variable XX. The error index contained in the X register is halved, converted to floating point, and stored in the variable X.

Next a search is made for line 50000. If it is found, the parser pointer is set to the start of line 50000 and the program jumps to the start of the BASIC execution loop. If no line 50000 is found, the normal error message is output and execution is halted.

Notes on 1P Version

Whenever the BREAK key is pressed, the 1P's vectors are reset to the original. The output vector again must be pointed to ON ERRor after every break. This can also be done with

```
POKE 538,34 : POKE 539,2
```

For the 1P version, the error type is contained in the variable X. Table 1 lists the error types. A program can be written to print out the full error descriptions if you have trouble remembering what "T*" means.

Notes on 540 Version

On error can also be set up using

```
POKE 4,64 : POKE 5,2
```

The first command in line 50000 should be PRINT. This scrolls the error message up one line to prevent retriggering ON ERRor. The 540 version does not put the error type into X, but the error type is displayed on the screen at $D741 and $D742. The ON ERRor program could be extended to read these locations and do a table look-up to get the error index.

Contact Mr. Morris at 3200 Washington, Midland, MI 48640. Contact Mr. Lourash at 1220 N. Dennis, Decatur, IL 62522.

Perry Peripherals Repairs KIMs!!
(SYMs and AIMs Too)

- We will Diagnose, Repair, and Completely Test your Single Board Computer
- We Socket all replaced Integrated Circuits
- You receive a 30-day Parts and Labor Warranty
- Your repaired S.B.C. returned via U.P.S. — C.O.D., Cash

Don't delay! Send us your S.B.C. for repair today
Ship To: (Preferably via U.P.S.)

Perry Peripherals
6 Brookhaven Drive
Rocky Point, NY 11778

KIM-1 Replacement Modules

- Exact replacement for MOS/Commodore KIM-1 S.B.C.
- Original KIM-1 firmware — 1K and 4K RAM versions

Replacement KIM-1 Keyboards

- Identical to those on early KIMS — SST switch in top right corner
- Easily installed in later model KIMs

Perry Peripherals is an authorized HDE factory service center.

Perry Peripherals carries a full line of the acclaimed HDE expansion components for you KIM, SYM, and AIM, including RAM boards, Disk Systems, and Software like HDE Disk BASIC V1.1. Yes, we also have diskettes. For more information write to: P.O Box 924, Miller Place, NY 11764, or Phone (516) 744-6462.

QUALITY SOFTWARE FOR TRS-80 COLOR AND OSI

ADVENTURES AND QUEST ALSO FOR SINCLAIR AND VIC-20

ADVENTURES!!!
For TRS-80 COLOR and OSI. These Adventures are written in BASIC, are full featured, fast action, full plotted adventures that take 30-50 hours to play. (Adventures are inter-active fantasies. It's like reading a book except that you are the main character as you give the computer commands like "Look in the Coffin" and "Light the torch.")

Adventures require 16k on TRS80, TRS80 color, and Sinclair. They require 8k on OSI and 13k on Vic-20. Derelict takes 12k on OSI. $14.95 each.

ESCAPE FROM MARS
(by Rodger Olsen)
This ADVENTURE takes place on the RED PLANET. You'll have to explore a Martian city and deal with possibly hostile aliens to survive this one. A good first adventure.

PYRAMID (by Rodger Olsen)
This is our most challenging ADVENTURE. It is a treasure hunt in a pyramid full of problems. Exciting and tough!

TREK ADVENTURE (by Bob Retelle)
This one takes place aboard a familiar starship. The crew has left for good reasons – but they forgot to take you, and now you are in deep trouble.

HAUNTED HOUSE (by Bob Anderson)
It's a real adventure – with ghosts and ghouls and goblins and treasures and problems – but it is for kids. Designed for the 8 to 12 year old population and those who haven't tried Adventure before and want to start out real easy.

DERELICT
(by Rodger Olsen & Bob Anderson)
New winner in the toughest adventure from Aardvark sweepstakes. This one takes place on an alien ship that has been deserted for a thousand years – and is still dangerous!

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.

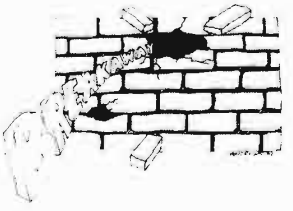
BASIC THAT ZOOOMMS!!
AT LAST AN AFFORDABLE COMPILER FOR OSI AND TRS-80 COLOR MACHINES!!! The compiler allows you to write your programs in easy BASIC and then automatically generates a machine code equivalent that runs 50 to 150 times faster.

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, IF, THEN, RETURN, END, PRINT, STOP, USR (X), PEEK, POKE, *, /, +, -, >, <, =, VARIABLE NAMES A-Z, SUBSCRIPTED VARIABLES, and INTEGER NUMBERS FORM 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)

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 TRS-80, TRS-80 Color, and Sinclair. 13K VIC-20. $14.95 each.



NEW!!
BREAKAWAY – ALL MACHINE CODE – Every computer has some form of BREAKAWAY available. Ours is fast, smooth, has 15 levels of difficulty – and is a bargain!! 16k TRS-80 Color only $9.95.

PROGRAMMERS!
SEE YOUR PROGRAM IN THIS SPACE!!
Aardvark traditionally pays the highest commissions in the industry and gives programs the widest possible coverage. Quality is the keyword. If your program is good and you want it presented by the best, send it to Aardvark.

Please specify system on all orders

ALSO FROM AARDVARK – This is only a partial list of what we carry. We have a lot of other games (particularly for the TRS-80 Color and OSI), business programs, blank tapes and disks and hardware. Send $1.00 for our complete catalog.

AARDVARK - 80
2352 S. Commerce, Walled Lake, MI 48088
(313) 669-3110

Phone Orders Accepted 8:00 a.m. to 4:00 p.m. EST. Mon.-Fri.

TRS-80 COLOR **SINCLAIR** **OSI** **VIC-20**

MICRO™

It's All 1's and 0's

American Gumby Corporation
1009 Mercaptan Road
Epstein-Barr, NJ 11235

New Product Announcement: 7903 MPU with Enhanced Instruction Set

The new 7903 MPU provides a plug-in upgrade for systems currently using the 6502 microprocessor. The 7903 is fully pin-compatible and software upward-compatible with the 6502. Additionally, many of the formerly unused opcodes are used to provide an enhanced instruction set providing features normally found only on much larger systems. The new instructions are listed below.

BBI	Branch on Blinking Indicator
BH	Branch and Hang
BCBF	Branch on Chip Box Full
BPO	Branch on Power Off
BSO	Branch on Sleepy Operator
IIB	Ignore Inquiry and Branch
RPB	Reverse Parity and Branch
BCH	Branch on CPU Halted
BTAD	Branch To Auto-Destruct
JRL	Jump to Random Location
JSP	Jump on Sexy Programmer
FAG	Fold And Go
AI	Add Improper
DO	Divide and Overflow
DC	Divide and Conquer
SRZ	Subtract and Reset to Zero
ARZ	Add and Reset to Zero
XM	Exclusive Maybe
PAII	Prevent All Interrupts and Interrupt
PI	Punch Invalid
RI	Read Invalid
RCSD	Read Card and Scramble Data
RCR	Rewind Card Reader
RASC	Read And Shred Card
BCR	Backspace Card Reader
BCP	Backspace Card Punch
RCI	Read Card and Ignore
RCS	Read Card Sideways
SSJ	Select Stacker and Jam
RP	Read Printer
FSRR	Forms Skip and Run Away
BSP	BackSpace Printer
PBC	Print and Break Chain
TDB	Transfer and Drop Bits
MDB	Move and Drop Bits
MLR	Move and Lose Record
MWC	Move and Wrap Core
MC	Move Continuous
CM	Circulate Memory
WWLR	Write Wrong Length Record
RNR	Read Noise Record
RIRG	Read Inter-Record Gap
REOF	Read End-Of-File .
BST	Backspace and Stretch Tape
RBT	Rewind and Break Tape
MTI	Make Tape Invalid
PMT	Punch Magnetic Tape
PDT	Punch and Delete Tape
MET	Misread and Eat Tape
PTAB	Position Tape Ass-Backwards
STT	Stretch and Tangle Tape
ST	Scratch Tape
SRSD	Seek Record and Scar Disk
RD	Rewind Disk
BD	Backspace Disk
ED	Eject Disk
TD	Throw Disk
LCD	Launch Cartridge Disk
FD	Flip Disk
DF	Disk Feed
UER	Update and Erase Record
CVU	ConVert to Unary
CVS	ConVert to Sesquinary
CRN	Convert to Roman Numerals
WRTC	Wind Real-Time Clock
WWTC	Wind Wrong-Time Clock
PCB	Pause for Coffee Break
SPD	Start and Power Down
PDN	Power Down and Normalize
EBQR	Enable Bi-Quinary Arithmetic
LCC	Load and Clear Core
EROS	Erase Read-Only Store
RWOM	Read Write-Only Memory
WROM	Write Read-Only Memory
FCE	Fill Core with Epoxy
DMPK	Destroy Memory Protect Key
UC	Unwind Core
BPP	Blob Plotter Pen
MPS	Move Pen Somewhere
DPMD	Drop Pen and Mangle Drum
APX	Apply Power and Explode
HCF	Halt and Catch Fire
CCP	Clear Core and Proceed
CCCP	Conditionally Clear Core and Proceed
EIOC	Execute Invalid Op Code
EPI	Execute Programmer Immediate
SPSW	Scramble Program Status Word
ERAF	Execute Relocatable Address Field
EPSW	Execute Program Status Word
EM	EMulate 407
SSN	Set Serial Number
STI	STore Immediate
PSP	Push Stack Pointer

MICRO™

Straightforward Garbage Collection for the Apple

by Cornelis Bongers

This article presents a method of garbage collection that dramatically increases program efficiency by eliminating lengthy delays caused by string processing.

Memory Organizer
requires:

Apple II or Apple II Plus with Applesoft in ROM or language card

Introduction

When processing large amounts of alphanumeric data in Applesoft (or another Microsoft BASIC) by means of strings and string operations, you will be confronted with one of Applesoft's weakest points: garbage collection. A program that runs perfectly may suddenly come to a grinding halt. Depending on the number of active strings in memory, you could be put out of business for a few tenths of a second or half an hour. Consider, for instance, the following simple program:

GARBAGE TEST on 48K Apple with DOS (HIMEM:$9600)

```
10 DIM A$(4600)
20 FOR I = 1 TO 4600
30 PRINT I
40 A$(I) = "$" + STR$(I)
50 NEXT
60 PRINT CHR$(7);" READY
   (AT LAST)"
```

This program runs fine until I becomes equal to 2740. At that point, the regular printing of I stops for 13 minutes, after which it continues again until I becomes 3835. This time you will have to wait for 20 mintues before the printing restarts.

In total, the time needed to finish the program is 84 minutes. Only three minutes (or 3.5%) are spent on the actual execution of the program. The remaining time is consumed by Applesoft's garbage collection routine. A question that may arise at this point is: What is meant by garbage and why is garbage collection necessary? To answer this question, consider the following program segments:

```
1000 GOSUB 5000: REM INPUT
     NAME AND CHECK NAME
1010 B$(I) = A$
1020 .....
.
.
.
5000 INPUT "GIVE NAME OF
     ITEM ";A$
5010 ....CHECK INPUT ....
.
.
.
5200 RETURN
```

When a new item has to be added to the list in B$, the subroutine at line 5000 is called, then asks for a name and checks whether this name satisfies some conditions (not listed here). At each new input the text that has been input is added to the stringpool and a pointer to this text is inserted in (the space reserved for) A$. The variable A$ itself is treated in the same way as a numeric variable and resides therefore in the variable space. There are seven bytes reserved for A$. The first two bytes contain the hex values 41 and 80, which represent the name. The next three bytes form the string descriptor and the last two bytes are unused. The string descriptor contains the length of the text (string) that has been input in A$ and the pointer to the string, in that order. So, if the text 'lens' is input, a snapshot of memory may look as displayed in figure 1.

If the checking in the subroutine is done, control returns to the main program and the assignment at line 1010 is executed. The target of the assignment is an array element. As in the case of simple variables (like A$), room is reserved for each array element. However, only the three bytes of the descriptor are reserved for each element of a string array. At the assignment, the top of free memory (i.e., the start of the stringpool) is decreased by four and the text 'lens' is copied to the area between the 'old' top and the 'new' top. Consequently, the text 'lens' now occurs twice in the stringpool (see figure 2).

Now, if line 5000 is executed later in the program, A$ gets a new value, say 'shutter'. This text is again added to the stringpool in the way described above and the pointer in the descriptor of A$ is set to the new top. The previous value of A$, i.e. 'lens', has now become non-active, since none of the pointers in the descriptors point to it. In other words, the text 'lens' that was related to A$ is of no use anymore and it is therefore called garbage.

Apart from assignment statements, garbage may also be generated if you make use of string expressions. In the first program listed above, where a string expression occurs at line 40, the stringpool will be filled with the strings, $1,$2,..., etc. But between every two strings, there is some garbage. For instance, after executing the I loop 10 times, the stringpool looks like this:

$1010$99$88$77$66$55$44$33$22$11

Only the underlined strings are active because they are referred to by the descriptors in the A$ array. If the I loop has been executed 2740 times, the stringpool occupies all the memory up to the top of the array space, and at that time Applesoft takes action. It knows that garbage is likely to be present in the stringpool and therefore it starts reorganizing the stringpool, eliminating the non-active strings. This reorganization process works as follows:

A complete pass is made through the variable and the array space. In this

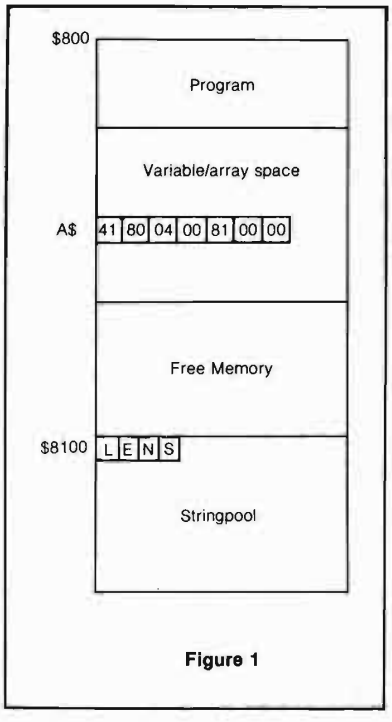


Figure 1

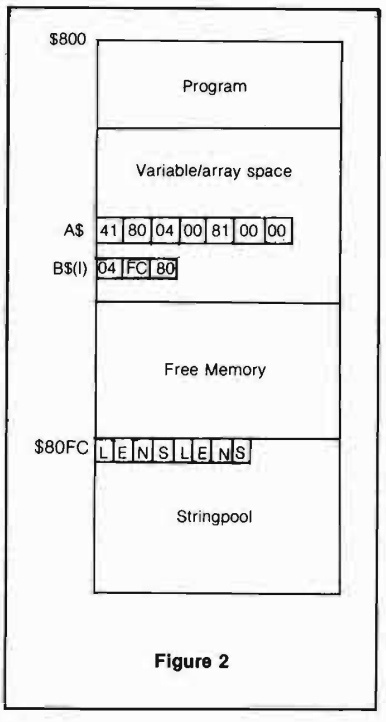


Figure 2

pass all string descriptors are examined with the purpose of finding the pointer to the highest string in the stringpool (i.e., the string nearest to HIMEM). The string that this pointer references is moved up to the top of memory and the descriptor of this string is appropriately adjusted. Next, a second pass is made through the variable and array space; now the next-but-highest pointer is searched for. The string pointed to is then moved to just below the string that was moved up during the previous pass. This process continues until all the active strings reside in a compact block that is located just below HIMEM. Then the pointer to the start of the stringpool is adjusted. Since all the garbage now has been eliminated, there will be sufficient free memory to continue the execution of the program.

The time needed to collect garbage with this method varies approximately quadratically with the number (n) of active strings in memory. This is due to the fact that Applesoft has to make n passes through the variable/array space and each pass consists of n comparisons and some other operations. Consequently, the total time needed is proportional to n*n. Thus if you have an array of 1000 active strings and you fill another array with 1000 strings, garbage collection will not last twice as long, but four times as long.

It is mainly the quadratic character of Applesoft's garbage collection routine that causes the long execution times if there are many active strings in memory (see table 1). Therefore, the best way to cut down execution time is to design a garbage collection method whose execution time varies linearly, rather than quadratically, with the number of active strings. Another way to speed up garbage collection, which may be used in combination with the first, is to clear (erase) all string arrays as soon as their contents are not needed anymore. This effectively reduces the number of active strings and therefore also the time to collect garbage.

The program listed in figure 3 contains a linear garbage collection routine and a selective array eraser.

How It Works

Basically the problem with garbage collection is that there are only pointers from variables and array elements to the strings and not the other way around. Thus when looking in the stringpool one can view the strings, but it is impossible to say to which variable/array element they belong or whether they are active or non-active (garbage). If somehow the strings themselves would indicate whether they are active or non-active and to which variable/array element they belonged, garbage collection would become a simple matter. Just move up all the active strings one by one to the current top of free memory and adjust for each string the pointer in the descriptor. Note that it is necessary to retrieve the length of the string from the descriptor before the move is done, since this length is used at the execution of the move.

We now have the interesting question: is it possible to store (temporarily) the necessary information (mentioned above) in the strings themselves? Of course this must happen under the condition that no extra memory space is used, since lack of memory is just what triggered FRE(0). However, the storage of information will require memory. But let us first consider how much memory exactly is required. Suppose first that all strings in the pool have a length of at least three bytes. By making a pass through the variable and array space, we can successively store the active/non-active marker and the pointer to the descriptor in each string. The insertion of the active/non-active marker does

Table 1: **Execution times (in seconds) of garbage collection routines as a function of the number of active strings in memory.**

No. of Active Strings	Applesoft(1)	'Linear' Method(1)	'Linear' Method(2)
100	1	0.05	0.06
250	5	0.11	0.14
500	19	0.21	0.26
1000	75	0.42	0.52
2000	292	0.84	1.04
3000	655	1.26	1.53
4000	1154	1.68	2.04
5000	1811	2.10	2.55

(1) All strings are of length 3.
(2) One string is of length 2, the others are of length 3.

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.

Figure 3

```
                ;************************
                ;*                      *
                ;*  MEMORY ORGANIZER I  *
                ;*          BY          *
                ;*   CORNELIS BONGERS   *
                ;*                      *
                ;*      ORGANIZER       *
                ;*                      *
                ;*                      *
                ;*                      *
                ;*                      *
                ;*                      *
                ;*                      *
                ;************************
                ;
                        ORG $9000
                        OBJ $800
                ;
                LEN     EPZ $06         ;LENGTH STRING
                STAT    EPZ $07         ;ARRAY/VAR OFFSET
                SDES    EPZ $08         ;POINTER TO STRING
                                         VARIABLE/ARRAY EL.
                RCHK    EPZ $17         ;RESULT MEMORY CHECK
                A1L     EPZ $3C         ;MONITOR MOVE REGISTERS
                A2L     EPZ $3E         ;   "     "       "
                A4L     EPZ $42         ;   "     "       "
                VARS    EPZ $69         ;START VARIABLE SPACE
                ARS     EPZ $6B         ;START ARRAY SPACE
                EARS    EPZ $6D         ;END ARRAY SPACE
                SPOOL   EPZ $6F         ;START STRING POOL
                HIM     EPZ $73         ;HIMEM
                SARA    EPZ $9B         ;POINTER TO START ARRAY
                SPTR    EPZ $9D         ;POINTER TO STRING
                ARP     EPZ $9F         ;POINTER TO NEXT ARRAY
                DO3     EPZ $A1         ;3RD STEP FLAG
                EPROG   EPZ $AF         ;POINTER TO END OF PROGRAM
                SDES1   EPZ $FC         ;POINTER TO STRING DESCRIPTOR
                TOP     EPZ $FE         ;START OF (NEW) STRING POOL
                ;
                ; ** BASIC AND MONITOR ROUTINES **
                ;
                MOVE    EQU $FE2C       ;MONITOR MOVE ROUTINE
                GNAME   EQU $F7D9       ;EVALUATE ARRAY NAME
                NAME    EQU $DFE3       ;EVALUATE VARIABLE
                PLET    EQU $DA63       ;PART OF BASIC'S LET
                CTYP    EQU $DD6A       ;CHECK ON NUMERIC
                FREO    EQU $E2E8       ;PART OF BASIC'S FRE(0)
                ITOO    EQU $E84E       ;INIT MFP TO 0
                CKOL    EQU $DEBB       ;CHECK ON '('
                CKOR    EQU $DEB8       ;CHECK ON ')'
                SYN     EQU $DEC9       ;SYNTAX ERROR
                CHRW    EPZ $B1         ;GET NEXT CHARACTER
                CHOL    EPZ $B7         ;GET CURRENT CHARACTER
                ;
                ; ** TOKENS AND SPECIAL CHARACTERS **
                ;
                FRE     EPZ $D6         ;TOKEN FOR FRE
                MIN     EPZ $C9         ;TOKEN FOR -
                CLEAR   EPZ $BD         ;TOKEN FOR CLEAR
                COM     EPZ $2C         ;COMMA
                BJP     EQU $3F5        ;& VECTOR
                ;
9000            ; ** INITIALIZATION **
9000            ;
9000 A9 13      BEGIN   LDA #START
9002 8D F6 03           STA BJP+1       ;SET & VECTOR
9005 A9 90              LDA /START
9007 8D F7 03           STA BJP+2
900A A9 00              LDA #BEGIN
900C 85 73              STA HIM         ;SET HIMEM
900E A9 90              LDA /BEGIN
9010 85 74              STA HIM+1
9012 60                 RTS
9013            ;
9013            ; ** ERASE ARRAYS **
9013            ;
9013 C9 BD      START   CMP #CLEAR      ;TOKEN FOR CLEAR?
9015 D0 35              BNE START1      ;NO,CHECK ON FRE
9017 20 B1 00   NXAR    JSR CHRW        ;ADVANCE TEXTPOINTER
901A 20 D9 F7           JSR GNAME       ;EVALUATE ARRAY NAME
901D 18                 CLC
901E A0 01              LDY #$01
9020 A2 FE              LDX #$FE
9022 B5 6F      DOAG    LDA EARS+2,X    ;SET UP POINTER TO
9024 95 40              STA A2L+2,X     ;END OF MOVE AREA
9026 B5 9D              LDA SARA+2,X    ;SET UP POINTER TO
```

(Continued)

not lead to problems because all characters have their high bit off when processing strings in the normal way. A string can thus be marked active by setting the first bit of a character in that string. The pointer to the descriptor requires two bytes and these can be substituted for two characters. Thus, if A$ equals 'MARJO', the stringpool contains somewhere the values 4D, 41, 52, 4A, and 4F. By making a pass through the variable/array space, A$ will be encountered and the string in the pool can be adjusted to 23, 0A, D2, 4A, and 4F. As can be seen, the third byte now has its high bit on, whereas the address of the descriptor is stored in the first and second bytes. (It is assumed that A$ resides at $A21). The only problem left is where to store the two characters M and A, which have been replaced by the address of the descriptor. The obvious answer is: store them in the descriptor of A$. This descriptor contains the pointer to the string 'MARJO' or what is left of it, but this pointer is not needed anymore, once the string is located.

Having applied this operation to all strings, a pass through the stringpool is made. We start at HIMEM and search down until a character with its first bit on is encountered. This character must belong to an active string and the address of the descriptor will be stored in the next two bytes. First the length of the string and its two characters are retrieved from the descriptor. Next the two characters are restored in the string and the string is moved up as far as possible. Finally, the new starting position of the string is stored in the descriptor. Then the next string is searched for and this process continues until the start of the (old) stringpool is reached.

The remaining problem is: what to do with strings whose length equals 1 or 2? Clearly, these strings are too short to allow us to store the necessary information in them. Again the answer is not difficult. Just copy the whole string in the descriptor and forget about the string in the pool (it will thus not be marked active). The descriptor occupies three bytes. One of these bytes will be needed to flag that the descriptor itself contains the string. For this purpose the high address byte of the pointer can be used by setting it to FF if a string is stored in the descriptor. Note that the high byte cannot equal FF during normal operations because this would mean that the string resides in the Monitor. If the length of a string equals 2, the remaining two bytes of the descriptor can be used to store the character. If the length equals 1, the string is put in the first byte of the

descriptor and the second byte is put equal to FF. Note that here again use is made of the fact that during normal operations all characters in the stringpool have their high bit off.

If strings with length 1 or 2 are present in the stringpool, it will be necessary to make a second pass through the variable/array space (after the pass through the stringpool). During this pass all strings that were stored in the descriptors are replaced in the stringpool and the pointers to these strings are inserted in the descriptors.

Table 1 shows the performance of the method outlined above. As can be seen, there exists, apart from some small measurement errors, a linear relationship between the number of active strings and the execution time. The fourth column of the table shows that the extra pass, which is necessary if the stringpool contains strings with a length less than 3, increases the execution time by about 20%.

The differences in execution time between Applesoft's garbage collection routine and the 'linear' garbage collection routine are considerable. For instance, if garbage collection is done when, say, 1000 active strings reside in memory (in an array), a speed improvement with a factor of 75/0.52 = 144 can be realized. When doubling the number of active strings, this factor doubles (approximately) too. So, with 4000 active strings, the speed improvement is about a factor of 4*144 = 576.

However, it should be noted that the speed improvement reduces if the (average) length of the strings increases. For instance, if there are 999 active strings of length 15 and one of length 1, the time gain is about a factor of 80. For 1999 active strings of length 15 and one of length 1, it is about a factor of 160.

The Program

The machine language program, listed in figure 3, is linked to BASIC by means of the & statement. The syntax of the statement for this program is

& FRE ([–] digit
[,name of numeric variable])

or

& CLEAR arrayname
[(,arrayname)]

As indicated by the keyword FRE, the first &-line handles the garbage collection. Contrary to Applesoft's

Figure 3 *(Continued)*

```
9028 95 44              STA A4L+2,X         ;TARGET OF MOVE AREA
902A C8                 INY
902B B1 9B              LDA (SARA),Y
902D 75 9D              ADC SARA+2,X        ;SET UP POINTER TO
902F 95 3E              STA A1L+2,X         ;START OF MOVE AREA
9031 E8                 INX
9032 D0 EE              BNE DOAG            ;EXECUTE LOOP TWICE
9034 A0 00              LDY #$00            ;Y MUST BE ZERO ON ENTRY MOVE
9036 20 2C FE           JSR MOVE            ;EXECUTE MOVE
9039 18                 CLC
903A CA                 DEX                 ;X=FF
903B B5 43      DOAG1   LDA A4L+1,X         ;ADJUST POINTER TO
903D E9 00              SBC #$00            ;END ARRAY SPACE
903F 95 6E              STA EARS+1,X
9041 E8                 INX
9042 F0 F7              BEQ DOAG1           ;EXECUTE LOOP TWICE
9044 20 B7 00           JSR CHKL            ;GET CURRENT CHARACTER
9047 C9 2C              CMP #COM            ;IS IT A COMMA?
9049 F0 CC              BEQ NXAR            ;ERASE NEXT ARRAY IF SO
904B 60                 RTS                 ;RETURN TO BASIC


904C                    ;
904C                    ; ** DRIVER FRE ROUTINE **
904C                    ;
904C C9 D6      START1  CMP #FRE            ;TOKEN FOR FRE?
904E F0 03              BEQ NSYN            ;BRANCH IF SO
9050 4C C9 DE           JMP SYN             ;ELSE SYNTAX ERROR
9053 4A         NSYN    LSR                 ;CLEAR FIRST BIT
9054 85 17              STA RCHK            ;SET UP CHECK REGISTER
9056 20 B1 00           JSR CHRW            ;GET NEXT CHAR
9059 20 BB DE           JSR CKOL            ;CHECK '('
905C C9 C9              CMP #MIN            ;MINUS?
905E D0 08              BNE FREAM           ;BRANCH IF NOT
9060 20 21 92           JSR CHECK           ;CHECK MEMORY ON NEG ASCII'S
9063 66 17              ROR RCHK            ;SAVE RESULT CHECK
9065 20 B1 00           JSR CHRW            ;GET DIGIT
9068 29 0F      FREAM   AND #$0F
906A F0 0B              BEQ DOFRE1          ;DO IT ALWAYS IF 0
906C 18                 CLC
906D 65 6E              ADC EARS+1
906F A6 6D              LDX EARS            ;IS EARS+DIGIT*256<SPOOL?
9071 E4 6F              CPX SPOOL
9073 E5 70              SBC SPOOL+1
9075 90 07              BCC NOFRE           ;YES, DON'T DO IT
9077 A5 17      DOFRE1  LDA RCHK            ;CHECK OK?
9079 30 03              BMI NOFRE           ;NO, DON'T DO IT
907B 20 A4 90           JSR DOFRE           ;COLLECT GARBAGE
907E 20 B1 00   NOFRE   JSR CHRW            ;GET NEXT CHARACTER
9081 C9 2C              CMP #COM            ;IS IT A COMMA
9083 D0 1C              BNE HAAK            ;BRANCH IF NOT
9085 20 B1 00           JSR CHRW            ;GET NEXT CHARACTER
9088 20 E3 DF           JSR NAME            ;EVALUATE NAME
908B 85 85              STA $85
908D 84 86              STY $86             ;SAVE POINTER FOR LET
908F 20 6A DD           JSR CTYP            ;MUST NOT BE A STRING
9092 20 E8 E2           JSR FREQ            ;CALCULATE FREE MEMORY
9095 A5 17              LDA RCHK
9097 10 03              BPL NZER            ;BRANCH IF TEST OK
9099 20 4E E8           JSR ITOQ            ;ELSE SET MEM TO Q
909C A5 12      NZER    LDA $12             ;LOAD TYPE VARIABLE
909E 20 63 DA           JSR PLET            ;SIMULATE A LET
90A1 4C B8 DE   HAAK    JMP CKOR            ;RETURN TO BASIC


90A4                    ;
90A4                    ; ** COLLECT GARBAGE **
90A4                    ;
90A4                    ; ** STEP 1: PUT POINTERS IN STRINGS **
90A4                    ;
90A4 85 A1      DOFRE   STA D03             ;INIT STEP 3 FLAG TO <> 0
90A6 20 6E 91           JSR INITS           ;SET POINTERS FOR NOSTR
90A9 20 80 91   GNXST   JSR NOSTR           ;SEARCH NEXT STRING
90AC B0 3B              BCS STEP2           ;BRANCH IF NO MORE STRINGS
90AE E4 AF              CPX EPROG           ;STRING IN PROGRAM?
90B0 E5 B0              SBC EPROG+1
90B2 90 F5              BCC GNXST           ;DON'T PROCESS IT IF SO
90B4 A6 06              LDX LEN
90B6 F0 F1              BEQ GNXST           ;NEGLECT IF LENGTH=0
90B8 E0 03              CPX #$03
90BA 90 14              BCC APART           ;BRANCH IF LENGTH EQUALS 1 OR
2
90BC B1 9D              LDA (SPTR),Y        ;Y=2 ON EXIT NOSTR
90BE 09 80              ORA #$80            ;SET 1ST BIT OF 3RD CHAR
90C0 91 9D      P2C     STA (SPTR),Y
90C2 88                 DEY
90C3 30 E4              BMI GNXST
90C5 B1 9D              LDA (SPTR),Y        ;PUT FIRST 2 CHARS
90C7 C8                 INY                 ;IN FIRST 2 BYTES OF DESC.
90C8 91 FC              STA (SDES1),Y
```

(Continued)

Figure 3 *(Continued)*

```
90CA 88                 DEY
90CB B9 FC 00           LDA SDES1,Y         ;PUT ADDRESS DESC. IN STRING
90CE B0 F0              BCS P2C             ;ALWAYS
90D0 A0 00      APART   LDY #$00
90D2 84 A1              STY D03             ;STEP 3 MUST BE DONE
90D4 B1 9D              LDA (SPTR),Y        ;PUT FIRST CHAR IN
90D6 91 FC              STA (SDES1),Y       ;LENGTH BYTE OF DESC.
90D8 A9 FF              LDA #$FF            ;PUT FF IN 2ND BYTE DESC.
90DA C8                 INY                 ;IF LENGTH EQUALS 1
90DB CA                 DEX
90DC F0 02              BEQ AFF
90DE B1 9D              LDA (SPTR),Y        ;PUT 2ND CHAR IN 2ND BYTE
90E0 91 FC      AFF     STA (SDES1),Y       ;OF DESC. IF LENGTH EQUALS 2
90E2 C8                 INY
90E3 A9 FF              LDA #$FF            ;PUT 3RD BYTE OF DESC.
90E5 91 FC              STA (SDES1),Y       ;EQUAL TO FF
90E7 30 C0              BMI GNXST           ;ALWAYS

90E9            ;
90E9            ; ** STEP 2: MOVE STRINGS UP **
90E9            ;
90E9 20 21 92   STEP2   JSR CHECK           ;SEARCH NEG ASCII'S
90EC 4C F2 90           JMP CON
90EF 20 2E 92   ESTEP2  JSR DEYY            ;CONTINUE SEARCH
90F2 90 3E      CON     BCC STEP3           ;FOUND NONE IF CARRY CLEAR
90F4 A0 02              LDY #$02
90F6 29 7F              AND #$7F            ;CLEAR 1ST BIT OF CHAR
90F8 91 9D              STA (SPTR),Y        ;AND RESTORE CHAR
90FA 88                 DEY
90FB B1 9D              LDA (SPTR),Y        ;NOW GET ADDRESS OF
90FD 85 09              STA SDES+1          ;OF DESCRIPTOR AND SAVE
90FF 88                 DEY                 ;IT IN SDES
9100 B1 9D              LDA (SPTR),Y
9102 85 08              STA SDES
9104 B1 08              LDA (SDES),Y        ;GET LENGTH STRING
9106 85 06              STA LEN             ;AND SAVE IT
9108 C8                 INY
9109 B1 08              LDA (SDES),Y
910B 88                 DEY                 ;GET 1ST CHAR OF STRING
910C 91 9D              STA (SPTR),Y        ;AND RESTORE IT
910E A0 02              LDY #$02
9110 B1 08              LDA (SDES),Y        ;GET 2ND CHAR OF STRING
9112 88                 DEY                 ;AND RESTORE IT
9113 91 9D              STA (SPTR),Y
9115 A5 FE              LDA TOP             ;CARRY IS SET
9117 E5 06              SBC LEN             ;CALCULATE NEW TOP OF
9119 85 FE              STA TOP             ;FREE MEMORY
911B B0 02              BCS NOADJ
911D C6 FF              DEC TOP+1
911F 91 08      NOADJ   STA (SDES),Y        ;STORE ADDRESS STRING
9121 A5 FF              LDA TOP+1           ;IN DESCRIPTOR
9123 C8                 INY
9124 91 08              STA (SDES),Y
9126 A4 06              LDY LEN
9128 88         MOV     DEY
9129 B1 9D              LDA (SPTR),Y        ;MOVE STRING TO NEW
912B 91 FE              STA (TOP),Y         ;LOCATION
912D 98                 TYA
912E D0 F8              BNE MOV
9130 F0 BD              BEQ ESTEP2          ;ALWAYS

9132            ;
9132            ; ** STEP 3: RESTORE STRINGS OF LENGTH 1 OR 2 **
9132            ;
9132 A5 A1      STEP3   LDA D03             ;IS THIS STEP NECESSARY?
9134 D0 08              BNE OMIT            ;BRANCH IF NOT
9136 20 6E 91           JSR INITS           ;INIT FOR NOSTR
9139 20 80 91   GNX2    JSR NOSTR           ;SEARCH NEXT STRING
913C 90 09              BCC NOKL            ;BRANCH IF FOUND
913E A5 FE      OMIT    LDA TOP             ;ALMOST READY NOW
9140 85 6F              STA SPOOL           ;UPDATE STRING POOL POINTER
9142 A5 FF              LDA TOP+1
9144 85 70              STA SPOOL+1
9146 60                 RTS                 ;RETURN TO DRIVER
9147 A2 01      NOKL    LDX #$01            ;NOSTR RETURNS WITH Y=2
9149 B1 FC              LDA (SDES1),Y       ;LENGTH STRING EQUAL TO 1 OR 2 ?

914B C9 FF              CMP #$FF
914D D0 EA              BNE GNX2            ;BRANCH IF NOT
914F 88                 DEY
9150 B1 FC              LDA (SDES1),Y       ;LOAD 2ND BYTE OF DESC.
9152 30 05              BMI NOCH+1          ;BRANCH IN IT IS NOT A CHAR
9154 E8                 INX                 ;INCREMENT LENGTH COUNTER
9155 20 14 92           JSR INS             ;STORE 2ND CHAR IN STRING POOL

9158 24 88      NOCH    BIT $88             ;HIDDEN DEY INSTRUCION
915A B1 FC              LDA (SDES1),Y       ;LOAD 1ST CHARACTER
```

(Continued)

FRE(0), the arguments of the & FRE are significant. One of the reasons to include arguments in the & FRE statement is that the user must be able to prevent a regular Applesoft FRE(0) from happening. This can be done by giving the & FRE(digit) command. If digit = 0 garbage collection will always be done. If digit < > 0 garbage collection will be done only if less than digit*256 free bytes of memory remain. Thus, & FRE(4) leads only to garbage collection if the amount of free memory is less than 1K. In case no garbage collection is done, the execution time of the statement will be about 0.0004 seconds only. It is therefore advised to insert the & FRE (digit) statement frequently in your program (of course with digit < > 0) to make sure Applesoft doesn't get the chance to execute its own FRE(0).

If a variable is specified in the & FRE statement, the amount of free memory that is available after garbage collection will be assigned to it. Thus, & FRE(0,K):PRINT K prints the amount of free memory.

Finally, if a minus sign is specified before the digit, the whole stringpool will be checked on the occurrence of

Remember when...

Grandma used envelopes for paying her bills, and it worked?

Now...

Grandma's system has been converted for your standard **VIC®**. Now it looks like a checkbook and works like a payroll deduction. It reserves your funds for what's due, and tells you what's left to spend.

Includes 4 Routines

on cassette tape...

- *Checks*
- *Deposit*
- *Reconcile*
- *Maintenance*

Check or money order accepted for $17.95 plus $1.50 postage and handling.

RAM/RBC SYSTEMS
P.O. Box 351
Malden, MA 02148

® *Trademark of Commodore*

characters which have their high bit on. As will be clear from the explanation in the previous section, the occurrence of such characters will lead to a terrible crash if an & FRE(0) is forced. However, & FRE(-0,K) leads to garbage collection only if all high bits are off. If characters are found which have their high bit on, this will be flagged by putting K equal to 0.

It will usually not be necessary to use the '-' option since under normal circumstances Applesoft will store all characters with their high bit off. However, in some programs statements of the type: A$ = CHR$(X) are used. For X > = 128 the execution of this statement will store 'characters' in the string pool with their high bit on. Therefore, if & FRE is implemented in a program in which 'high bit on' characters might occur, it is strongly advised to check the stringpool at the first execution of the & FRE statement. If one is sure there are 'no high bit on' characters, it is better to omit the check since it increases execution time.

The second &-line can be used to clear (erase) arrays. For instance, & CLEAR A, A$, A% clears the arrays A, A$, and A%. An array that is mentioned in the & CLEAR statement must have

RGB Color Board for the Apple II Computer **$179.00**

A video board for the Apple II computer that provides RED, GREEN, BLUE, and COMPOSITE SYNC signals that may be used with an RGB video monitor. Displays of color graphics or text are significantly superior in resolution and color quality as compared to the standard composite video display available from the Apple II. The board may be used with the 80-character Videx board, so that both color graphics and 80-column text may be displayed on one RGB monitor. The board plugs into slot 7 of the Apple II. Modifications are not required of the computer or software. A text page may be displayed in any one of eight colors. Output signals are +TTL, Composite Sync is −TTL. *Model NCT-A2*

Barcode Reader for Apple II Computer **$299.00**

Enables you to read 12-digit American UPCA and European EANA barcodes. Includes HP HEDS 3050 Wand with interface to plug into games I/O Port, driver software on DOS 3.3 disk. Complete instructions with same demo. *Model BCR-A2*

RGB 19" Trinitron Video Monitor
Displays 80 x 24 characters. **$1250.00**

RGB input, TTL or analog; Composite video input. Audio input. Switchable Under-Scan Display. Switchable H.V Sync Signal Input for +TTL or −TTL. Perfect for use with IBM micro, Apple III, Apple II with RGB video board, or other Hi-Res video. *Model KX-1901M.*

25" model also available. *Model KX2501M.*

Dealer Inquiries Invited

Video Marketing, Inc.
Box 339
Warrington, PA 18976
(215) 343-3000

Figure 3 *(Continued)*

```
915C 20 14 92          JSR INS              ;AND STORE IT IN POOL
915F 8A                TXA
9160 91 FC             STA (SDES1),Y        ;STORE LENGTH IN DESCRIPTOR
9162 C8                INY
9163 A5 FE             LDA TOP
9165 91 FC             STA (SDES1),Y        ;SAVE POINTER TO STRING
9167 C8                INY                  ;IN DESCRIPTOR
9168 A5 FF             LDA TOP+1
916A 91 FC             STA (SDES1),Y
916C D0 CB             BNE GNX2             ;ALWAYS

916E              ;
916E              ; ** SET UP POINTERS FOR NOSTR **
916E              ;
916E A9 07        INITS   LDA #$07
9170 85 07              STA STAT            ;VARIABLES OFF SET
9172 38                 SEC
9173 A5 69              LDA VARS
9175 E9 07              SBC #$07            ;SET SDES TO START OF
9177 85 08              STA SDES            ;VARIABLE SPACE-7
9179 A5 6A              LDA VARS+1
917B E9 00              SBC #$00
917D 85 09              STA SDES+1
917F 60                 RTS



9180              ;
9180              ; ** SEARCH NEXT STRING **
9180              ;
9180 18           NOSTR   CLC
9181 A5 07              LDA STAT
9183 AA                 TAX
9184 65 08              ADC SDES            ;ADVANCE DESCRIPTOR POINTER
9186 85 08              STA SDES            ;WITH 7 OR 3
9188 90 02              BCC NOIN
918A E6 09              INC SDES+1
918C E0 07        NOIN    CPX #$07          ;ARE WE DEALING WITH ARRAYS?
918E F0 36              BEQ DOVAR           ;BRANCH IF NOT
9190 A6 09        DOAR    LDX SDES+1
9192 A0 00              LDY #$00
9194 E4 A0              CPX ARP+1           ;END ARRAY REACHED?
9196 D0 19              BNE VAR2            ;BRANCH IF NOT
9198 C5 9F              CMP ARP
919A D0 15              BNE VAR2            ;BRANCH IF NOT
919C F0 3E              BEQ NEXTAR          ;YES, DO NEXT ARRAY
919E A0 00        VARSP   LDY #$00
91A0 B1 08              LDA (SDES),Y
91A2 30 DC              BMI NOSTR           ;BRANCH IF NOT A STRING
91A4 C8                 INY
91A5 B1 08              LDA (SDES),Y
91A7 10 D7              BPL NOSTR           ;IDEM
91A9 88                 DEY
91AA A5 08              LDA SDES
91AC 69 02              ADC #$02            ;ADD 2 BYTES TO
91AE 90 01              BCC VAR2            ;COMPENSATE FOR NAME
91B0 E8                 INX
91B1 85 FC        VAR2    STA SDES1         ;SAVE POINTER TO DESCRIPTOR
91B3 86 FD              STX SDES1+1
91B5 B1 FC              LDA (SDES1),Y
91B7 85 06              STA LEN             ;SAVE LENGTH STRING
91B9 C8                 INY
91BA B1 FC              LDA (SDES1),Y
91BC 85 9D              STA SPTR            ;SET POINTER TO STRING
91BE AA                 TAX
91BF C8                 INY
91C0 B1 FC              LDA (SDES1),Y
91C2 85 9E              STA SPTR+1
91C4 18                 CLC                 ;RETURN WITH CARRY CLEAR
91C5 60           RTSS    RTS
91C6 A6 09        DOVAR   LDX SDES+1        ;START ARRAY SPACE REACHED?
91C8 E4 6C              CPX ARS+1
91CA D0 D2              BNE VARSP           ;BRANCH IF NOT
91CC C5 6B              CMP ARS
91CE D0 CE              BNE VARSP
91D0 46 07              LSR STAT          ;PROCESS ARRAYS NOW, PUT STAT= 3

91D2 85 9F              STA ARP             ;SAVE POINTER TO ARRAY IN ARP
91D4 86 A0              STX ARP+1
91D6 A5 9F        NEN     LDA ARP
91D8 A6 A0              LDX ARP+1
91DA A0 00              LDY #$00
91DC E4 6E        NEXTAR  CPX EARS+1        ;END ARRAY SPACE REACHED?
91DE D0 04              BNE ARRY            ;BRANCH IF NOT
91E0 C5 6D              CMP EARS
91E2 F0 E1              BEQ RTSS            ;RTS WITH CARRY SET IF SO
91E4 85 08        ARRY    STA SDES
91E6 86 09              STX SDES+1
```

(Continued)

Figure 3 *(Continued)*

```
91E8 B1 08              LDA (SDES),Y        ;GET 1ST CHAR OF ARRAY NAME
91EA AA                 TAX                 ;SAVE IN X
91EB C8                 INY
91EC B1 08              LDA (SDES),Y        ;GET 2ND CHAR OF ARRAY NAME
91EE 48                 PHA                 ;SAVE ON STACK
91EF C8                 INY
91F0 B1 08              LDA (SDES),Y        ;ADD LENGTH ARRAY TO ARP
91F2 65 9F              ADC ARP
91F4 85 9F              STA ARP
91F6 C8                 INY
91F7 B1 08              LDA (SDES),Y
91F9 65 A0              ADC ARP+1           ;ARP POINTS TO
91FB 85 A0              STA ARP+1           ;NEXT ARRAY
91FD 68                 PLA                 ;GET 2ND CHAR OF NAME
91FE 10 D6              BPL NEN             ;BRANCH IF NOT A STRING ARRAY
9200 8A                 TXA                 ;GET 1ST CHAR OF NAME
9201 30 D3              BMI NEN             ;BRANCH IF NOT A STRING ARRAY
9203 C8                 INY
9204 B1 08              LDA (SDES),Y        ;GET # OF DIMS
9206 0A                 ASL                 ;MULTIPLY BY 2
9207 69 05              ADC #$05            ;ADDOVERHEAD
9209 65 08              ADC SDES            ;ADD SDES
920B 85 08              STA SDES            ;SDES POINTS TO FIRST DESC.
920D 90 81              BCC DOAR            ;OF ARRAY
920F E6 09              INC SDES+1
9211 4C 90 91           JMP DOAR


9214            ;
9214            ;** DEREMENT TOP AND SAVE CHARACTER **
9214            ;
9214 A4 FE      INS     LDY TOP             ;HIGH BYTE DECR. NECESSARY?
9216 D0 02              BNE OK              ;BRANCH IF NOT
9218 C6 FF              DEC TOP+1
921A C6 FE      OK      DEC TOP
921C A0 00              LDY #$00
921E 91 FE              STA (TOP),Y         ;SAVE CHARACTER
9220 60                 RTS



9221            ;
9221            ; ** CHECK/SEARCH NEGATIVE ASCII'S IN STRING POOL
9221            ;
9221 A2 01      CHECK   LDX #$01
9223 B5 73      CHNM    LDA HIM,X           ;COPY HIMEM IN TOP AND IN SPTR

9225 95 FE              STA TOP,X
9227 95 9D              STA SPTR,X
9229 CA                 DEX
922A F0 F7              BEQ CHNM
922C A0 00              LDY #$00
922E A5 9E      DEYY    LDA SPTR+1
9230 C5 70              CMP SPOOL+1         ;IS SPTR(HIGH)<SPOOL(HIGH)?
9232 90 2B              BCC RTS2            ;RETURN WITH CARRY CLEAR IF SO

9234 C6 9E              DEC SPTR+1          ;SET SPTR FOR SEARCH
9236 88         DEYY1   DEY
9237 B1 9D              LDA (SPTR),Y
9239 30 05              BMI FOUND           ;FOUND ONE
923B 98                 TYA
923C D0 F8              BNE DEYY1
923E F0 EE              BEQ DEYY            ;ALWAYS
9240 48         FOUND   PHA                 ;SAVE BYTE FOR A WHILE
9241 A6 9E              LDX SPTR+1
9243 98                 TYA
9244 18                 CLC
9245 65 9D              ADC SPTR            ;SPTR=SPTR+Y
9247 90 01              BCC NOINCR
9249 E8                 INX
924A E4 70      NOINCR  CPX SPOOL+1         ;IS SPTR<SPOOL?
924C 90 10              BCC RTS1            ;RETURN WITH CARRY CLEAR IF SO

924E D0 04              BNE SBC2
9250 C5 6F              CMP SPOOL
9252 90 0A              BCC RTS1            ;IDEM
9254 E9 02      SBC2    SBC #$02            ;SPTR=SPTR-2
9256 85 9D              STA SPTR
9259 B0 01              BCS NODECR
925A CA                 DEX
925B 86 9E      NODECR  STX SPTR+1
925D 38                 SEC                 ;RETURN WITH CARRY SET
925E 68         RTS1    PLA                 ;RESTORE BYTE
925F 60         RTS2    RTS
9260            ;
9260            ; ** END OF PROGRAM **
9260            ;
9260                    END
```

been dimensioned earlier in the program, else an OUT OF DATA error will be generated.

& CLEAR can be used to reduce garbage collection time still further by timely clearing of string arrays. In addition, it can be used to clear numerical working arrays. This will especially be of use if a program consists of several subroutines, each of which require differently dimensioned array space to do data manipulations. By DIMing the working arrays on entry of the subroutine and by clearing them on exit, one prevents that memory from becoming littered with unused arrays. This reduces array access time and leads to fewer 'OUT OF MEMORY' problems.

Finally, & CLEAR can be used to initialize an array to zero, for instance by & CLEAR A: DIM A(20,20). These statements execute about 80 times faster than the usual zero-assignment within a double loop.

Installing the Program

The program has been developed with the excellent BIG MAC assembler, recently released by *Call —A.P.P.L.E.* (for more information see: *Call —A.P.P.L.E.*, Vol. IV, Number 7, page 37). *Editor's note: Figure 3 was reassembled by LISA for purposes of uniformity.*

The machine-language program starts at $9000 and has a length of $260 bytes. After assembling the text file and storing it to disk, the program can be installed by : BRUN programname. This command executes an initialization routine that sets HIMEM to $9000 and installs the & vector. If you want to BRUN the routine from within an Applesoft program, the BRUN command should be inserted at the first line of the program, and must be followed by a CLEAR command. For example:

```
10 PRINT "BRUN program name": CLEAR: REM control D behind first quotes.
```

The program makes use of some Applesoft routines in ROM. If the RAM version of Applesoft is being used, the relevant subroutine calls have to be adjusted.

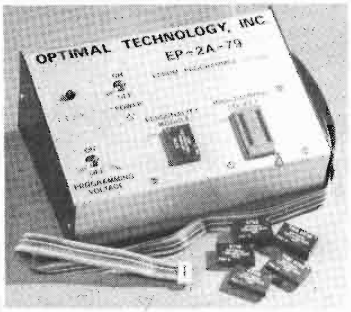
Contact the author at Erasmus University, P.O. Box 1738, 3000 DR Rotterdam, The Netherlands.

MICRO

Model EP-2A-79

EPROM Programmer

North Star
Apple
S-100
SS-50
STD-Bus
Atari
Pet
Kim-1



TRS-80
H-8
H-89
Ohio Scientific
SWTP
Aim-65
Sym-1

Three years in the field with unsurpassed performance. Software is available for the EP-2A-79 for most all of the microcomputers including the popular CP/M, FLEX, HDOS operating systems. Write or call for specific hardware/software interfacing. Driver packages available for F-8, 6800, 6809, 8080,8085, Z-80, 1802, 6502 and 2650 based systems.

EP-2A-79 115V 50/60 HZ $169.00

Personality Modules

PM-0	TMS 2708	$17.00	PM-5	2716, 2758	$17.00
PM-1	2704, 2708	17.00	PM-5E	2816	35.00
PM-2	2732	33.00	PM-8	MCM68764	35.00
PM-2A	2732A	33.00	PM-9	2764	35.00
PM-3	TMS 2716	17.00	SA-64-2	TMS 2564	39.00
PM-4	TMS 2532	33.00	SA-64-3	2764	39.00

Optimal Technology, Inc.

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

IS THIS YOU?

You know you saw their ad in one of those magazines stacked in the corner... but which one? You are ready to buy their hard/soft Thing-a-magig, but how are you going to find them? What can you do?

THE COMPUTERIST'S DIRECTORY YELLOW PAGES is the one national reference to companies [large and small] and individuals producing and selling the hardware, software, services and supplies that you need. Extensive indexing and cross referencing by product makes it easy to find what you are looking for in the personal and small business field.

THE COMPUTERIST'S DIRECTORY WHITE PAGES list Individuals, Clubs, Associations and Bulletin boards including network I.D. Numbers [Source, CompuServe etc] and a short description of their interests and projects. Find people and clubs who share your interests. Form local and national networks. And if your computer nomenclature is rusty... Just check with the Glossary included in each issue. All from the Computerist's Directory White Pages!

THE COMPUTERIST'S DIRECTORY is published twice each year. A one year subscription costs just $10.00 and includes a free listing in the White Pages and two issues of the Directory, one in January and one in July. It's the best investment you'll make this year!

YOU ONLY HAD TO LOOK IN ONE PLACE!

the Computerist's Directory

The National Phone Book of Computing

PO BOX 405
FORESTVILLE, CA 95436

(707) 887-1857

DON'T WASTE TIME AND MONEY SIMPLY BECAUSE YOUR APPLE . . .

S.D.S. INVITES YOU TO FREE YOURSELF & YOUR COMPUTER!

DOUBLETIME PRINTER™

IT'S LIKE GETTING ANOTHER APPLE FOR ONLY $150!

Stop losing time and money while you or your staff wait for the printer. Doubletime Printer uses special interrupt driven software to print files INDEPENDENTLY of the program currently being run.

Unlike some buffered printer interface cards, Doubletime Printer is an integrated hardware/software package that offers more than just a limited print buffer.

With Doubletime Printer, you can print multiple copies, prioritize printing schedules, do formatting of listings and text, all independently of the program being run in your Apple at the time. Doubletime Printer uses your diskette as a buffer, and as such has a much greater capacity than any RAM buffer card.

To find out more about Doubletime Printer, see your local Apple dealer, or call or write Soutwestern Data Systems for a free catalog which includes a complete description of Doubletime Printer and over 20 other programs for the Apple.

southwestern data systems™

P.O. BOX 582 • SANTEE, CA 92071
TELEPHONE: 714/562-3670

OSI Extended I/O Processor

by Michael J. Keryan

More than thirty new functions can be added to an OSI C1P by redefining the input, output, and control-C routines. Most of the new functions are activated by either control character keyboard input, or escape sequences placed in BASIC.

Extended I/O
requires:
OSI C1P with 8K
May be modified for other OSI machines.

"Cursor Control for the C1P," by Kerry Lourash (May, 1981 MICRO), added nine utility functions to the input and output routines. I have pieced together the desirable features of most of these smaller programs, and added a number of new ones, such as automatic line number generation. In all, over thirty routines are now available for use during keyboard input, screen output, etc. User-supplied software/hardware additions for a printer, bell, and bug-free garbage collection are also supported. An improved monitor program is included, which can be called at any time. All the constants — screen parameters, subroutine vectors, and flags — were put into tables, rather than imbedded into machine code, making changes relatively easy. The program was originally written for a C2-8P, but the version described here is for a C1P with 8K of memory. The 2K program is ROMable, assuming all the references to the high byte of subroutines ($18 through $1F) are translated to higher memory.

The Video Screen

Several screen parameters are stored in page zero memory, as shown in figure 1 and table 1. There are no restrictions on screen size or video memory location; 32, 64, or non-standard line widths can be supported, as well as video memory at locations other than $Dxxx. Figure 1 shows the window starting near the top of the screen and the flags and monitor fields (described later) near the bottom, but all locations can be modified. During initialization, the parameters are copied from tables within the program (default locations) to lower memory. The parameters can be changed by POKEing into pages zero and two, but the default values will be re-established on each warm start. Therefore, if the default values do not suit you, change them in the upper memory tables.

Cursor Movement

The cursor position is stored in locations $00E0 (low byte) and $00E1 (high byte). The cursor movement functions print the character under the cursor, move the cursor, and print the cursor symbol (stored in location $00E9) at the new position. No other output to the CRT or printer is affected. The following control characters will cause non-destructive cursor movement to any screen location:

Up one line	—	control-U ($15)
Down one line	—	control-D ($04)
Left one space	—	control-L ($0C)
Right one space	—	control-R ($12)
Right eight spaces	—	control-I ($09)

Use of these cursor movements can put the cursor outside an active window. The following movement controls keep the cursor within an active window:

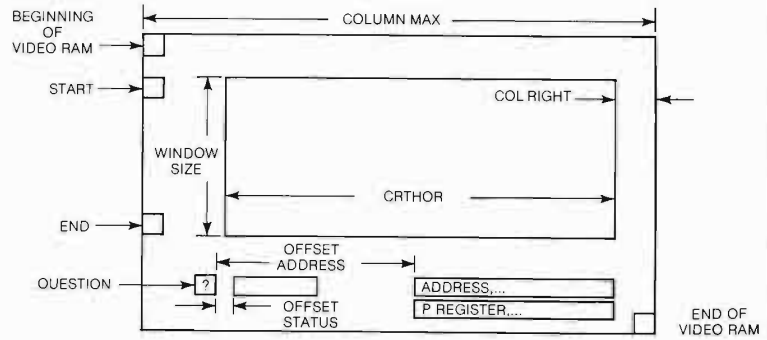
Return to the left of a line
control-Q ($11)

Home cursor (to bottom of window)
control-B ($02)

Backspace (like control L, but stays in margins)
control-H ($08)

Move cursor (to a preset location)
control-N ($0E)

***Figure 1:* CRT Screen Parameters**



COLMAX – 1 = COLUMN MAX – 1
CRTHOR + 1 = CRTHOR + 1
COL L – R = (COLUMN MAX) – (CRTHOR) – (COL RIGHT) + 2
OFFSET – P REGISTER = COLUMN MAX – 14

Control-N will move the cursor to the location stored in $02DA (low byte) and $02DB (high byte). It is now set for the top left corner of the screen. Note that if the preset location is outside the window, control-N will cause the cursor to leave the window.

Window Controls

Active window boundaries are stored in START: $00E2, $00E3, and END: $00E4, $00E5. All CRT output, scrolling, etc., will be maintained within these boundaries. An alternate window is stored in START2: $02D6, $02D7, and END2: $02D8, $02D9. The two windows could be equivalent, partially overlapping, or completely separate.

The two windows can be switched by pressing control-W ($17). In addition to toggling the windows, the cursor will be homed in the new active window.

The window boundaries can be changed by POKEing into the appropriate locations, but are easily changed by using the control-X ($18) key. To use control-X, first place the cursor anywhere on the desired line by using control-U or control-D, then press control-X. You will be prompted for another key with a question mark (at location $00E6, $00E7) and a beep (if this function is implemented), until either a T (for top of window) or a B (for bottom) is pressed. Control-X will only change boundaries of the active window; to change the other window's boundaries, first use control-W.

If the cursor is placed above the window, it will naturally move down into (and be trapped in) the window. If the cursor is placed below the bottom boundary, however, it will not move by itself from that line. This can be used for a one-line non-scrolling window, but a two-line window is the minimum required to give readable text.

Scroll Controls

If the cursor is placed near the top of the window, it will move down the screen as lines of text are output. No scrolling will occur until the cursor attempts to move down when at the bottom of the window; the whole window will then scroll upward and the home line will be blanked. An upward scroll can be forced at any time by pressing control-Y ($19); similarly, a downward scroll is forced by control-Z ($1A). These functions control only the location of the text, which is moved up or down on the screen; they do not move the cursor, which remains stationary. The scrolling functions are useful in editing and in game programs.

Clear Controls

To erase the entire screen, press either control-T ($14) or ESCAPE ($1B). To erase only the active window, press RUBOUT ($7F); this will also home the cursor in the window.

Edit Text

Text can be entered by typing it in as usual, or by placing the cursor anywhere on the screen and pressing control-E ($05). This causes whatever is under the cursor to be entered into BASIC; it has the same effect as typing the character. The cursor is then indexed one space to the right.

When entering a line of text, characters can be deleted with shift O ($5F); this moves the cursor one space backwards, deletes the character from BASIC, and erases it from the CRT. The function of shift P ($64) is not changed; it scratches from BASIC the line being worked on, but does not erase the line from the CRT.

To summarize, text is entered by typing characters (or spaces) or by using control-E over text. Text can be deleted by typing spaces over text when using control-E or with shift O. Text is not changed by using cursor controls; these are used only to position the cursor to allow use of a combination of control-E, character input, or space input.

Autoline

To facilitate easy entry of text, an automatic line entering system can be invoked by inputting control-A ($01). Control-A toggles the autoline mode off or on at any time. It can also be changed by POKEing the status flag. When the autoline mode is on, an A will appear near the bottom of the screen. You then enter a carriage return to activate autoline.

When the system is initialized, the starting line number will be 100 and the increment is 10, resulting in lines numbered 100, 110, 120, ..., 9990. The line number and increment can be changed at any time by POKEing locations $02D0 and $02D1 (line number) and $02D2 (increment). These are packed BCD numbers, four bits per digit. The default values will be re-established on warm start.

When the autoline mode is on, the input routine looks at both the character being entered and the last character. If the last character was a carriage return, you are now at the beginning of a new line, possibly in

Table 1: **Parameter Location and Values (for C1P)**

PARAMETER	LOCATION	DEFAULT LOCATION	DEFAULT VALUE
CURSOR-LO	$00E0	$1C00	A0
CURSOR-HI	$00E1	$1C01	D0
START1-LO	$00E2	$1C02	A0
START1-HI	$00E3	$1C03	D0
END1-LO	$00E4	$1C04	C0
END1-HI	$00E5	$1C05	D2
QUESTION-LO	$00E6	$1C06	C5
QUESTION-HI	$00E7	$1C07	D3
OK SYMBOL	$00E8	$1C08	E5
CURSOR SYMBOL	$00E9	$1C09	A4
COLUMN MAX	$00EA	$1C0A	20
COLMAX-1	$00EB	$1C0B	1F
COL L-R	$00EC	$1C0C	07
COL RIGHT	$00ED	$1C0D	01
CRTHOR+1	$00EE	$1C0E	1A
STATUS FLAGS	$00EF	$1C0F	82
CONTROL C FLAG	$0212	$1C36	00
AUTOLINE-LO	$02D0	$1C10	90
AUTOLINE-HI	$02D1	$1C11	00
AUTOLINE INCREMENT	$02D2	$1C12	10
LINES/PAGE-CRT	$02D3	$1C13	12
LINES/PAGE-PRINTER	$02D4	$1C14	30
START2-LO	$02D6	$1C16	01
START2-HI	$02D7	$1C17	D3
END2-LO	$02D8	$1C18	80
END2-HI	$02D9	$1C19	D3
MOVE CURSOR-LO	$02DA	$1C1A	A5
MOVE CURSOR-HI	$02DB	$1C1B	D0
OFFSET-STATUS	$02DC	$1C1C	00
OFFSET-ADDRESS	$02DD	$1C1D	08
OFFSET-P REGISTER	$02DE	$1C1E	12

need of a new line number. Entering any character other than a space, a control character, a number from 0-9, a shift-O, or a rubout, will automatically generate a new line number before the key is entered. These exceptions allow certain things to be done without getting a line number put on it: immediate mode commands are invoked by first typing a space, then the command; new line numbers can be inserted between or over existing lines; and all cursor and editing commands can be used. The autoline mode can simply be toggled off by using control-A.

Flag Changes

To change a status flag, use control-F (\$06). You will then get a prompt. You must then enter the flag number (from 1 to 8), followed by either a 0 (for off) or 1 (for on). The flag code numbers are:

Flag Number	Code	Description
1	H	Hard copy (printer) mode
2	C	CRT output mode
3	I	Intermittent output (paging) mode
4	T	Trace mode
5	S	Step mode
6	A	Autoline mode
7	M	Monitor save mode
8	E	Extended I/O mode (all functions)

After the flag number and status is entered, the status of all flags will be displayed near the bottom of the screen (these can be erased by escape or control-T). The status can also be changed at any time (e.g., during execution of a BASIC program) by POKEing bits into location \$00EF; the flag number corresponds to the bit number. Note that if the E flag is cleared, you can get back into the extended I/O mode by POKEing a number greater or equal to 128 (\$80) into \$00EF, or a warm start.

CRT and Hardcopy Flags

When these flags are set to 1, a corresponding output to the screen or printer will be created. These flags are independent. To get printed output, a user-supplied printer subroutine must be included: change the NOP's at \$1EF7 to JSR \$YYXX (20 XX YY), where \$YYXX is the address of your subroutine. Prior to this subroutine call, 16 page-zero locations (\$00EX) are freed for additional use by the print routine, and are restored before returning to the CRT output.

Print Window

At any time, a control-P (\$10) from the keyboard will cause the entire active window to be output to the printer, character by character. The H flag need not be set. The CRT display is not affected.

Intermittent Output

If the I flag is set, the number of lines output to the CRT/printer are counted and stored in locations \$02F6/\$02D5. These are compared to constants stored in locations \$02D3/\$02D4. If the line count is equal to the preset page size, the computer will prompt you and wait for a keyboard entry before continuing. This will allow you to copy (or read) CRT text before it scrolls off, or change to a new sheet of paper on the printer. These counts are independent; both are reset to zero on warm start.

Stop/Restart Output

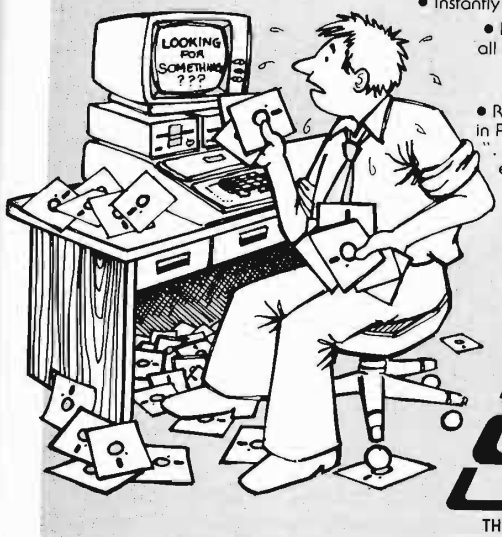
In addition to the above intermittent output mode, a program or listing can be stopped at any time by pressing control-S (\$13) and then restarted by control-R (\$12). These commands are functional only during output. In many cases, the control S/R sequence is preferred over control C/CONT since no extraneous output is printed.

Step and Trace Modes

If the Step mode is invoked by setting the S flag, only one line of BASIC code will be executed during RUN. You will then be prompted for a keyboard entry, after which the next line will be executed, and so on.

TAKE CHARGE OF YOUR GROWING COLLECTION OF DISK-BASED SOFTWARE!

• A professional database, dedicated to your disk catalog! • Consolidates up to 1,200 catalog entries into a single file!
• Instantly retrieves any file name, sorts in 4 seconds!
• Produces well organized reports of all your programs, nicely formatted to your printer for easy insertion into a notebook, etc!
• Reviewed by InfoWorld as Excellent in Performance and Ease of Use, and "...delivers on all its claims... an elegant system... well worth it."



EASY TO LEARN & USE!
Ask your local Apple dealer about Disk Library or call or write SDS for a free catalog.

disk Library™
THE SOFTWARE MANAGEMENT SYSTEM

southwestern data systems™
P.O. BOX 582 • SANTEE, CA 92071 • TELEPHONE: 714/562-3670

If the Trace mode is invoked by setting the T flag, the BASIC line number will be printed when that line is executed. The output will then be a mixture of line numbers with the normal program output. The program cannot be LISTed while in T mode.

The Step and Trace modes are independent, but for most purposes, are used together for debugging programs. The control-C flag (at location $0212) must be cleared (enabled) to activate either Step or Trace: this is done on warm start.

View Tape

Pressing control-V ($16) will cause entry into the cassette view mode, where BASIC tapes can be read and displayed on the CRT, but are not entered. To exit this mode, enter a space. This routine uses the old I/O vectors to eliminate accidental control character routine activation during viewing.

Bell

An audible prompt is used in several of the above routines. This bell function is also used when a control-G ($07) is either input or output. $07 is output if you attempt to enter more than 71 characters on a line. As an additional feature, the bell is also sounded once after the 64th character, just like a typewriter to warn you that the end of the line is near. To use the Bell feature, the user must supply a subroutine at location $1CEC and the appropriate hardware modifications. (See MICRO, July 1981, "A Typewriter Bell for Your Microcomputer.")

Carriage Return on BASIC Input

With OSI computers, if you respond to an input statement with only a carriage return, you will be kicked out of your program into the immediate mode. You can usually jump back in with a CONT statement, but this is frustrating. On most large computers, such a response is legal. This feature has been added to the input routine. A carriage return will be accepted as a zero for numeric inputs, such as INPUT A, or as a space ($20) for string inputs, such as INPUT A$.

Other Jumps

An input of $1D will cause a jump to the menu ($FF00). This duplicates the function of the Break (Reset) key, and makes it easy to jump there from inside a BASIC program. Inputs of $1C, $1E, or $1F are not used. You can add your own functions by adding your vectors to the tables located at $1800-$183F.

Escape Sequence on Output

Most of the functions are accessed by entering a control character ($01-$1F) from the keyboard, either in immediate mode, or in response to an INPUT statement. These functions can also be accessed on output, either in immediate mode, or by a BASIC program. An escape sequence is used. The escape code ($1B, decimal 27) is output, followed by the control code. For example, to toggle windows, execute:

```
PRINT CHR$(27);CHR$(23);
```

The last semicolon is used to keep the display from scrolling. To output the graphic character for $1B, output two consecutive escapes:

```
PRINT CHR$(27);CHR$(27);
```

Of course, not all functions are suitable for use during a BASIC run, but many are extremely useful, including cursor movements, scrolling, window toggles, screen clear, bell, print, etc. A summary of control functions is shown in table 2.

New Monitor

An improved machine-language monitor routine is accessed by input of control-K ($0B). This monitor is significantly better than OSI's minimal monitor, but not as versatile as commercial monitors. The advantage of this monitor is that it can be called at any time: in immediate mode, in the middle of a BASIC program, or by a JSR machine-language call.

Once the monitor is entered, data appears at the bottom of the screen, as shown in figure 1. The screen locations of this data are set by constants stored at $00E6 (low byte) and $00E7 (high byte), and offsets $02DD and $02DE. There are eight fields shown:

L — Location (four character address)
H — Hexadecimal data stored in L
C — ASCII character stored in L
S — Stack pointer
P — Processor status register (flags)
A — Accumulator
X — X register
Y — Y register

The "cursor" in the monitor mode is controlled by the keys "," and "."; these keys were chosen because the

Table 2: **Summary of Control Key Functions**

CONTROL KEY	HEX	DECIMAL	FUNCTION	LOCATION
@	00	0	NONE-NULL	$185C
A	01	1	AUTOLINE TOGGLE	$1DF5
B	02	2	BOTTOM CURSOR (HOME)	$19F7
C	03	3	NONE-CONT C	$185C
D	04	4	DOWN CURSOR	$193D
E	05	5	EDIT	$187E
F	06	6	FLAG CHANGE	$1DD0
G	07	7	BELL	$1CEC
H	08	8	BACKSPACE CURSOR	$1905
I	09	9	INCREMENT CURSOR 8 SPACES	$1924
J	0A	10	NONE-LINE FEED	$185C
K	0B	11	MONITOR	$1A48
L	0C	12	LEFT CURSOR	$18F2
M	0D	13	NONE-CARR. RETURN	$185C
N	0E	14	MOVE CURSOR	$1946
O	0F	15	NONE-CONT O	$185C
P	10	16	PRINT WINDOW	$1EA3
Q	11	17	RETURN CURSOR	$1919
R	12	18	RIGHT CURSOR / RESTART	$190E
S	13	19	STOP OUTPUT	$185C
T	14	20	CLEAR SCREEN	$1CA0
U	15	21	UP CURSOR	$192D
V	16	22	VIEW TAPE	$1885
W	17	23	WINDOW TOGGLE	$18A8
X	18	24	SET WINDOW	$18BC
Y	19	25	SCROLL UP	$1894
Z	1A	26	SCROLL DOWN	$189E
ESC	1B	27	CLEAR SCREEN	$1CA0
-	1C	28	--	$185C
-	1D	29	JMP TO $FF00 (MENU)	$1C7A
-	1E	30	--	$185C
-	1F	31	--	$185C

symbols for the left arrow and right arrow appear on these keys. The "," will move the cursor left, the "." will move it right. The cursor actually changes the lower case letters l, h, c, etc., to the upper case letter to be changed. Any field is changed by typing new data into it. The C field will allow any character (except "," and ".") to be entered; the other seven fields allow only hexadecimal (0-9, A-F) characters.

Machine-language programs can thus be entered, or memory reviewed or changed one byte at a time. The space bar is used to step forward through memory; the carriage return key is used to step backwards. To return to where you were before you entered the monitor, type R.

To jump to a subroutine (whose location is shown in L), type J; if the subroutine executes correctly and is terminated by an RTS ($60), control will return to the monitor. All flags and registers (S, P, A, X, and Y) will be changed to what was shown on the screen just before the jump occurred. When returning to the monitor, the contents of S, P, A, X, and Y shown on the screen will reflect their status at the time of return. No provisions are made for single step, trace, trap, etc.

When the monitor mode is entered, several things happen. All flags and registers are saved, and the P field is initialized to $04 (ignore interrupts and clear decimal mode). The S field is adjusted to prevent change to the stack. If the P register is changed, it will automatically be restored on return. However, if the stack is disturbed, you may run into problems when returning, unless the original page one ($01XX) was saved. If the M flag of $00EF is set, the first three pages of memory — page zero (BASIC constants and routines), page one (the stack), and page two (BASIC and Extended I/O constants) — are saved in the top three quarters of screen memory ($D000-$D2FF). This will allow you to use these lower memory locations for your machine-language programs. They will be restored from the screen memory when exiting the monitor mode (R). If the M flag is clear, these three pages will not be saved. Leave the M flag cleared if you merely want to examine or change a few memory locations, or if you don't want the screen display disturbed.

Garbage Collector

A bug in OSI's BASIC-in-ROM may cause your program to bomb if you make extensive use of dimensioned strings. Provisions have been made to allow you to add a foolproof machine-language garbage collection routine. This routine will be called through the revised control-C routine if fewer than 512 bytes of free memory are available; this will keep OSI's defective routine from being called. To use this function, insert $20 XX YY at $1D72, where $YYXX is the location of your new garbage collection routine. In addition, the approximate number of free pages can be monitored at any time by PEEKing at $02F8. This can be used in lieu of FRE(X); never call FRE(X) when using dimensioned strings, as this will force a fatal garbage collection by the defective routine.

Initialization

First cold start, then Break-M, load the tape containing the Extended I/O routines, Break-M, then type .1D1FG. The initialization routine will then be run. The input, output, and control-C vectors are pointed to new routines. The warm start and OK routines are replaced by new ones. Tables are copied from within the program to page zero and page two, where they are used by the new routines. The memory size is adjusted to keep BASIC from overwriting the new routines. The stack is adjusted to prevent an OM error after a warm start, then a message is written to the screen.

***Table 3:* Hex Dump of Complete Program**

$1800

```
5C F5 57 5C 3D 7E D0 EC 05 24 5C 48 F2 5C 46 5C
A3 19 0E 5C A0 2D 85 A8 BC 94 9E A0 5C 7A 5C 5C
18 1D 19 18 19 18 1D 1C 19 19 18 1A 18 18 19 18
1E 19 19 18 1C 19 18 18 18 18 18 1C 18 1C 18 18
A9 AD 8D 07 02 A9 8D 8D 0A 02 A9 60 8D 0D 02 A5
E3 8D 09 02 8D 0C 02 A5 E2 8D 0B 02 60 A9 20 48
20 40 18 A4 E4 A6 E5 68 20 0A 02 EE 0B 02 D0 03
EE 0C 02 CC 0B 02 D0 F0 EC 0C 02 D0 EB 60 AD 01
02 8D 02 02 60 20 F4 FF 20 83 1C 20 7D 1C AD 03
02 D0 F5 60 20 94 19 20 A0 19 20 89 19 60 20 94
19 20 52 1E 20 89 19 60 A2 03 B5 E2 48 BD D6 02
95 E2 68 9D D6 02 CA 10 F1 4C 57 19 20 6C 19 48
A5 E1 48 20 D9 1C C9 54 D0 09 68 85 E3 68 85 E2
18 90 0A C9 42 D0 EC 68 85 E5 68 85 E4 60 20 5D
18 20 CF 19 A5 E5 85 E1 A5 E4 20 6E 19 85 E0 18
90 61 20 94 19 A5 EC 48 A9 00 85 EC 20 7B 19 68
85 EC 18 90 4E 20 94 19 20 74 19 18 90 45 20 94
19 E6 E0 D0 3E E6 E1 D0 3A 20 94 19 20 6C 19 85
E0 18 90 2F A2 08 20 0E 19 CA D0 FA 60 20 94 19
A5 E0 38 E5 EA 85 E0 B0 1A C6 E1 D0 16 20 94 19
20 5D 19 18 90 0D 20 94 19 AD DA 02 85 E0 AD DB
02 85 E1 20 89 19 60 20 94 19 4C E4 18 A5 E0 18
65 EA 85 E0 90 02 E6 E1 EA EA EA 60 A5 E0 05 EB
38 E5 EE 60 20 6C 19 C5 E0 D0 0B A5 E0 38 E5 EC
85 E0 B0 02 C6 E1 C6 E0 60 A0 00 B1 E0 8D 01 02
A5 E9 D0 03 AD 01 02 A0 00 91 E0 60 A5 E8 D0 F7
20 40 18 18 65 EA 90 03 EE 09 02 8D 08 02 A6 E4
A4 E5 20 07 02 EE 08 02 D0 03 EE 09 02 EE 0B 02
D0 03 EE 0C 02 EC 0B 02 D0 E8 CC 0C 02 D0 E3 A4
EB A9 20 91 E4 88 10 FB 60 AD 02 02 C9 20 B0 F8
AA BD 00 18 8D F1 02 BD 20 18 8D F2 02 6C F1 02
EE D5 02 EA EA EA 18 90 1B A9 20 20 97 19 20 2D
```

Table 3 *(continued)*

$1A00

```
1A 18 90 10 C6 0E A9 20 8D 01 02 20 97 19 20 74
19 20 90 19 4C F6 1F EA EA EA 8D 01 02 20 97 19
E6 E0 A5 E0 05 EB 38 E5 ED C5 E0 D0 17 EE F6 02
20 6C 19 85 E0 C5 E4 A5 E1 E5 E5 B0 04 4C 40 19
EA 20 A0 19 20 89 19 60 08 48 8A 48 98 48 BA CA
CA 86 43 24 EF 50 1C A2 00 B5 00 9D 00 D0 CA D0
F8 BD 00 01 9D 00 D1 CA D0 E7 BD 00 02 9D 00 D2
CA D0 F7 A5 E6 18 6D DD 02 85 49 A2 00 8A 20 98
1A A2 04 86 44 6D DE 02 20 98 1A A5 E7 85 4A 20
A8 1A 20 22 1B 18 90 F7 95 4B 69 06 95 4C 69 04
95 4D 69 04 95 4E 60 EA A2 07 B4 4B BD 1A 1B 91
49 CA 10 F6 A6 42 B4 4B BD 1A 1B 29 DF 91 49 A2
07 B5 40 20 FE 1A B4 4B 20 0F 1B CA E0 02 D0 F1
A0 00 B1 40 A4 4D C8 91 49 20 FE 1A A4 4C 20 0F
1B A5 41 20 FE 1A A4 4B 20 0F 1B A5 40 20 FE 1A
20 0F 1B 60 29 0F F8 18 69 90 69 40 D8 60 48 4A
4A 4A 4A 20 F4 1A 85 53 68 20 F4 1A 85 54 60 C8
A5 53 91 49 C8 A5 54 91 49 60 6C 68 63 73 70 61
78 79 20 83 1C C9 2C D0 09 C6 42 10 04 A9 07 85
42 60 C9 2E D0 0D E6 42 A6 42 E0 08 90 04 A9 00
85 42 60 A6 42 E0 02 D0 05 A2 00 81 40 60 C9 0D
D0 09 A5 40 D0 02 C6 41 C6 40 60 C9 20 D0 07 E6
40 D0 02 E6 41 60 C9 4A D0 27 BA 86 48 A6 43 9A
A6 46 A4 47 A5 44 48 A5 45 28 20 8E 1B 08 85 45
68 85 44 84 47 86 46 BA 86 43 A6 48 9A 60 6C 40
00 C9 52 D0 2E 68 68 AD 00 D0 C9 4C D0 1E EA EA
A2 00 BD 00 D0 95 00 CA D0 F8 BD 00 D1 9D 00 01
CA D0 F7 BD 00 D2 9D 00 02 CA D0 F7 68 A8 68 AA
68 28 60 48 20 93 FE 10 02 68 60 A8 68 98 A6 42
D0 0E A2 03 06 40 26 41 CA 10 F9 05 40 85 40 60
E0 01 D0 0F 85 48 A2 00 A1 40 0A 0A 0A 05 48
81 40 60 16 40 16 40 16 40 16 40 15 40 95 40 60
```

***Listing 1:* BASIC Program to Print a Hex Dump**

```
100 POKE239,131
105 PRINT:PRINT"$1800"
110 FORI=6144TO6655STEP16
120 PRINT
130 FORJ=0TO15
140 K=I+J
150 A=PEEK(K):POKE85,A
160 POKE11,247:POKE12,28
170 X=USR(0):C=PEEK(83):D=PEEK(84)
180 PRINTCHR$(C);CHR$(D);" ";
190 NEXTJ,I
200 POKE239,130
```

Odds and Ends

A subroutine that will decode a byte into two ASCII characters is located at $1CF7. Place the byte to be decoded into $0055. A JSR $1CF7 will leave the high nibble character in $0053, the low one in $0054. An example of this routine is shown in listing 1. The simple program generated the hexadecimal dumps of table 3. Lines 100 and 200 turned the printer on and off. Line 160 set the USR vector to $1CF7.

A dump of the entire 2K program is shown in table 3; the underlined bytes are those that will require changing if the program is relocated. Here are the locations that will require changing if your OSI computer is not a C1P:

Location	Function	C1P Location (low, high)
$1C7E	Old Output Routine	6 9 F F
$1C81	Old Output + 3	6C FF
$1C84	Old Input	BA FF
$1D5D	Old Control-C Routine	9B FF

However, you *must* have a support ROM (or EPROM) containing indirect vectors for these routines, which vector through page two of memory.

Due to its length, the assembly listing could not be reproduced here. For a copy of the 40-page listing, send $5 to the author. (Sorry, I cannot provide copies on tape.)

The control keys can be redefined any way you see fit, by changing the pointers shown in table 2; these are stored at the beginning of the program ($1800-$183F). You may want to eliminate some functions (such as printer routines) and add others. You may want to let some keys generate predefined strings that can be entered into BASIC, such as DATA, or FOR I=1TO, etc. For hints on how to do this, study the autoline code. You may want to make some changes. I have yet to use a program that didn't need a few alterations.

Michael Keryan has a Master of Science in Chemical Engineering. His interest is in hardware projects; he built an OSI system a few years ago and added a number of extensions, including two printers, music generators, and a real time clock. Contact Mr. Keryan at 713 Locust Drive, Tallmadge, OH 44278.

Table 3 *(continued)*

$1C00

```
A5 D0 A0 D0 C0 D2 C5 D3 E5 A4 20 1F 07 01 1A 82
90 00 10 12 30 00 01 D3 80 D3 A5 D0 00 08 12 25
A9 A2 A0 1D 20 C3 A8 60 EA 4C 1F 1D 4C 9A 1E FE
1E 5E 1F 42 1D A9 00 8D 12 02 A9 00 8D F6 02 A9
AA 8D F4 02 A2 0F BD 00 1C 95 E0 CA 10 F8 A2 0F
BD 10 1C 9D D0 02 CA 10 F7 A9 18 85 86 A9 00 8D
D5 02 EA EA A2 05 BD 29 1C 95 00 CA 10 F8 A2 05
BD 2F 1C 9D 18 02 CA 10 F7 60 4C 00 FF 4C 69 FF
4C 6C FF 4C BA FF EA EA 48 8A 48 A2 0F BD E0 02
48 B5 E0 9D E0 02 68 95 E0 CA 10 F1 68 AA 68 60
48 98 48 A0 00 A9 20 99 00 D7 99 00 D6 99 00 D5
99 00 D4 99 00 D3 99 00 D2 99 00 D1 99 00 D0 C8
D0 E5 68 A8 68 60 20 D0 1C C9 0A B0 F9 48 68 60
20 D9 1C 20 93 FE 30 F8 60 20 EC 1C A2 00 A9 3F
81 E6 20 83 1C 48 A9 20 81 E6 68 60 48 EA EA EA
EA EA EA EA EA 68 60 A5 55 4C FE 1A 48 20 16 1D
C9 BE D0 10 A9 F7 20 18 1D C9 7F D0 07 20 D9 1C
C9 12 D0 F9 68 60 A9 FE 8D 00 DF AD 00 DF 60 20
A0 1C AD F4 02 C9 AA F0 00 A9 00 8D D5 02 8D F6
02 EA EA EA 20 35 1C EA EA 20 20 1C A2 FE 9A 4C
74 A2 AD 12 02 D0 1E A5 EF 29 10 F0 03 20 D9 1C
A5 EF 29 08 F0 03 20 5A B9 20 66 1D 4C 9B FF EA
EA EA EA EA EA 60 A5 82 38 E5 80 8D F8 02 C9 02
B0 03 EA EA EA 60 A2 08 A9 01 8D F0 02 8A 18 6D
DC 02 A8 AD F0 02 25 EF D0 04 A9 20 D0 03 BD 99
1D 91 E6 0E F0 02 CA D0 E4 60 45 4D 41 53 54 49
43 48 45 78 74 65 6E 64 65 64 20 49 2F 4F 20 50
72 6F 63 65 73 73 6F 72 0D 0A 43 31 50 20 56 65
72 73 69 6F 6E 20 63 2E 4D 4B 20 31 39 38 31 00
20 C6 1C AA A9 01 CA F0 03 0A D0 FA 48 20 C6 1C
D0 0A 68 49 FF 25 EF 85 EF 18 90 05 68 05 EF 85
EF 20 76 1D 60 A9 20 45 EF 85 EF 4C 76 1D AD 02
```

Table 3 *(continued)*

$1E00

```
02 C9 5F F0 44 C9 7F F0 40 C9 21 90 3C C9 3A B0
04 C9 30 B0 34 18 F8 AD D0 02 6D D2 02 8D D0 02
48 AD D1 02 69 00 8D D1 02 48 D8 A2 00 4A 4A 4A
4A 20 4A 1E 68 29 0F 20 4A 1E 68 48 4A 4A 4A 4A
20 4A 1E 68 29 0F 20 4A 1E 60 09 30 95 13 E8 4C
1A 1A 20 40 18 A5 E5 8D 09 02 8D 0C 02 A5 E4 05
EB 8D 0B 02 38 E5 EA B0 03 CE 09 02 8D 08 02 A6
E2 A4 E3 20 07 02 CE 08 02 D0 03 CE 09 02 CE 0B
02 D0 03 CE 0C 02 EC 0B 02 D0 E8 CC 0C 02 D0 E3
A4 EB A9 20 91 E2 88 10 FB 60 A9 0D 20 5E 1F 4C
9C 19 EA 48 8A 48 98 48 A5 E0 48 A5 E1 48 EA EA
EA EA A5 E2 85 E0 A5 E3 85 E1 A0 00 A9 0D 20 F4
1E A6 EA B1 E0 20 F4 1E E6 E0 D0 02 E6 E1 CA D0
F2 A9 0D 20 F4 1E A5 E1 C5 E5 90 E5 A5 E0 C5 E4
90 DF F0 DD EA EA EA 68 85 E1 68 85 E0 68 A8
68 AA 68 60 20 88 1C EA EA EA 20 88 1C 60 A5 EF
30 03 4C 83 1C 20 83 1C 8D 02 02 98 48 8A 48 A9
40 C5 0E D0 03 20 EC 1C A9 0D CD F3 02 D0 1B CD
02 02 D0 08 A9 20 20 4C 1E 18 90 09 A5 EF 29 20
F0 08 20 FE 1D 8A A8 68 98 48 AD 02 02 C9 7F D0
03 20 DE 18 20 D9 19 EA EA EA EA EA EA EA EA EA
AD 02 02 8D F3 02 68 AA 68 A8 AD 02 02 60 48 8D
02 02 A5 EF 30 04 68 4C 7D 1C 8A 48 98 48 20 FC
1C A5 EF 29 01 F0 1C A5 EF 29 04 F0 10 AD D5 02
CD D4 02 90 08 20 D9 1C A9 00 8D D5 02 AD 02 02
20 F4 1E A5 EF 29 02 F0 5F A5 EF 29 04 F0 10 AD
F6 02 CD D3 02 90 08 20 D9 1C A9 00 8D F6 02 AD
02 02 C9 0A D0 03 4C F0 19 C9 0D D0 03 4C F9 19
C9 5F D0 03 4C 04 1A C9 07 D0 03 20 EC 1C C9 1B
D0 11 AD F5 02 D0 07 A9 01 8D F5 02 D0 1A A9 00
8D F5 02 AD F5 02 F0 0A 20 D9 19 A9 00 8D F5 02
F0 06 AD 02 02 20 1A 1A 68 A8 68 AA 68 4C 80 1C
```

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

O.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!

VIDEX:			
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 AS10	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

WORD-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
IOS:			
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:			
Quickscreen	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" Soft 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.
REQUIRES 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!
REQUIRES 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.
REQUIRES 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

Delete on the OSI

by Earl Morris and Yasuo Morishita

This utility will show the ROM BASIC user how to delete blocks of lines, as well as single lines, with just a few keystrokes.

DELETE
requires:
OSI C1P

This article describes a routine to allow users of OSI ROM BASIC to delete multiple lines. Normally only a single line of BASIC can be deleted by typing in the line number followed by a carriage return. This can be tedious if a large block of lines must be removed. This often happens, for example, when programs are merged or a utility program is run with another program also in memory. The "DELETE" program creates a USR routine which is called by

Z = USR (first line)(last line)

All of the lines of BASIC with line numbers inside the specified range are then deleted.

When OSI ROM BASIC is called on to delete a single line, two major routines are used. The code at $A2A2 finds the line to be deleted, and then shrinks the program by the number of bytes found in the offending line. Another routine at $A31C is responsible for refixing the pointers that rechain each line to the next. Unfortunately these routines are not written as subroutines and thus cannot be used by "outside" programs.

However, the DELETE program copies these routines from ROM into RAM and creates the needed subroutine. The main line DELETE pro-

BASIC Program to Set Up USR Delete Function

```
10 REM  BASIC LINE DELETE
12 REM FORMAT :  Z=USR(START LINE #)(END LINE #)
14 M=565:REM START ADDRESS=$0235 RELOCATABLE
16 A=INT(M/256):POKE12,A:POKE11,M-A*256
18 N=64:FORX=MTOM+N-1:READJ:POKEX,J:NEXT
20 A=41634:M=M+N:N=68:GOSUB28:REM DELETE=$A2A2
22 A=INT(M/256):B=M-256*A:POKEM-13,A:POKEM-14,B
24 A=41756:M=M+N:N=47:GOSUB28:REM REBUILD  =$A31C
26 POKEM+15,96:END: REM "RTS"
28 FORX=0TON-1:J=PEEK(A+X):POKEM+X,J:NEXT:RETURN
30 DATA32,8
32 DATA180,32,173,170,32,49,184,165,175,133
34 DATA48,165,174,133,49,32,50,164
36 DATA176,27,160,1,177,170,240,26,160,3
38 DATA177,170,133,18,136,177,170,133,17
40 DATA166,48,165,49,228,17,229,18
42 DATA144,5,32,125,2,240,219,169,146,160
44 DATA161,32,195,168,76,25,163
```

Source Code for Main Delete Program

```
                        ;DELETE
                        ;BY MORRIS & MORISHITA
                        ;
                        ;ASSEMBLY LANGUAGE LISTING
                        ;
                                ORG $235

                        ;
0235 20 08 B4                   JSR $B408           ;1ST ARGUMENT TO BINARY INTO $
11,12 (START)
0238 20 AD AA                   JSR $AAAD           ;GET 2ND ARGUMENT (LAST LINE # )

023B 20 31 B8                   JSR $B831           ;CONVERT TO BINARY
023E A5 AF                      LDA $AF
0240 85 30                      STA $30             ;STORE FINAL LINE # IN $30,31
0242 A5 AE                      LDA $AE
0244 85 31                      STA $31
0246 20 32 A4           LBLA    JSR $A432           ;FIND ADDRESS OF BASIC LINE
0249 B0 1B                      BCS LBLB            ;BRANCH IF FOUND, OTHERWISE UP
                                                     DATE POINTER AT $11,12
024B A0 01                      LDY #$01
024D B1 AA                      LDA ($AA),Y         ;LOOK AT POINTER TO NEXT LINE
024F F0 1A                      BEQ LBLC            ;IF NULL MUST BE END OF PROGRAM
                                                     SO QUIT
0251 A0 03                      LDY #$03
0253 B1 AA                      LDA ($AA),Y         ;GET NEXT LINE # HI BYTE
0255 85 12                      STA $12
0257 88                         DEY
0258 B1 AA                      LDA ($AA),Y         ;GET NEXT LINE # LO
025A 85 11                      STA $11
025C A6 30                      LDX $30             ;LOAD X,A WITH FINAL LINE
025E A5 31                      LDA $31
0260 E4 11                      CPX $11             ;COMPARE TO CURRENT LINE
0262 E5 12                      SBC $12
0264 90 05                      BCC LBLC            ;QUIT IF BEYOND FINAL LINE
0266 20 75 02           LBLB    JSR $0275
0269 F0 DB                      BEQ LBLA            ;ALWAYS BRANCH
026B A9 92              LBLC    LDA #$92
026D A0 A1                      LDY #$A1            ;$A192 IS ADDRESS OF "OK"
026F 20 C3 A8                   JSR $A8C3           ;PRINT "OK"
0272 4C 19 A3                   JMP $A319           ;GO BACK TO BASIC
0275                    ;
0275                            END
```

gram accepts the first line to be deleted and calls the copied ROM routine to do the work. Then the line pointers are used to find the line number of the next BASIC line. This is checked for end of program, and checked to see if it exceeds the upper limit for deleting. Then the copied routines are called again and the process is repeated until completed. Lines are still deleted one at a time, but the computer, rather than your busy fingers, is doing the work.

The BASIC program listed here will create the DELETE program on page two below the start of BASIC program space. This memory is normally unused in OSI machines. If you are using this space, then the delete program can be relocated by changing the value of "M" in line 14. Line 16 sets up the USR vector, and line 18 builds the main program from the DATA statements. Line 20 moves the "memory close" routine from ROM. Line 22 calculates an absolute JSR address and POKEs it into the main program. Line 24 copies the rechaining routine from ROM and line 26 adds an "RTS" to convert it to a subroutine.

After running the BASIC program, it can delete itself with:

```
Z = USR (10)(44)
```

Note that the USR function now requires two arguments and will give an "SN" error if both are not present. Everything is deleted by Z=USR(1)(-1) which of course is the same as a NEW command. The form Z=USR(A)(B) is also helpful to figure out which lines to omit.

For those readers interested in how the program works, the source code for the main program is listed with comments. The code is relocatable with the exception of the JSR at $026E. This is a jump to the copied ROM routines. The BASIC set-up program automatically fixes this absolute address.

Earl Morris may be contacted at 3200 Washington, Midland, Michigan 48640. Contact Yasuo Morishita at 950 Beau Drive #106, Des Plains, Illinois 60016.

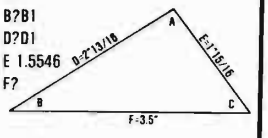
MICRO™

THE TRIANGULATOR

Solve triangles on your Apple* and get rid of your calculator and pencil!!

This program will solve any right or oblique triangle. Results can be printed or used on further calculations. Previous results are recalled with a simple letter/number pair and can be added or subtracted from each other or from new data. For example, the height of the triangle below is calclulated as follows:

```
A?          A?
B?          B?B1
C?          D?D1
D?2"13/16   E 1.5546
E?1"15/16   F?
F?3.5
```



Entry can be fraction, decimal, or DMS. Results are rounded to 4 places. Requires 48k Apple II+, DOS 3.2 or 3.3, parallel printer.

Send $39.95 + $1.50 postage and handling to:

Arrow Data Systems
1224 E. Harmont
Phoenix, AZ 85020

*Apple is a registered trademark of Apple Computer, Inc.

Dealer inquires invited. (602) 997-6638

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!
A FULL 8K OPERATING SYSTEM source listing add — $100.00 / $ 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
CHARGEX VISA

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
master charge

MICRO™

Apple Slices

By Tim Osborne

This month's Apple Slices shows you how to access directly any sector on an unprotected Disk II 16-sector diskette. And I'll discuss further how to pass an Applesoft arithmetic expression from BASIC to machine language (see last month's column (50:59)).

The DOS RWTS Subroutine

This method uses the DOS subroutine RWTS, which stands for Read or Write a Track or Sector. You may find this method referred to as physical I/O because it deals with physical data-access locations. It can be handy in error trapping, fixing clobbered diskettes, customizing catalogs, file and space management, accessing disks from machine language, and transferring files between different operating systems.

To connect to RWTS the user program should JSR to the RWTS vector located at $3D9. Upon entry to RWTS the A + Y registers must contain the address of the IOB (Input/Output control Block). The IOB is a table of parameters that describes to RWTS what operation it is being called to perform. For a breakdown of the IOB, see page 95 of the Apple II DOS Manual. The IOB in turn contains the address of the DCT (Device Characteristics Table) in bytes seven and eight. The DCT provides physical and timing information to the RWTS subroutine.

The Example

I have written a handy subroutine, AMPERRWTS, which allows the user to call RWTS from BASIC using the following syntax:

1 . READ; &R(T,S,B,D)
&R(T,S,B)

2. WRITE; &W(T,S,B,D)
&W(T,S,B)

T = A valid Applesoft arithmetic expression representing a track number (0-34).

S = A valid Applesoft arithmetic expression representing a sector number (0-15).

***Listing 1:* AMPERRWTS**

```
0800                1  ;*********************************
0800                2  ;*                               *
0800                3  ;*         APPLE SLICES          *
0800                4  ;*     ACCESSING RWTS FROM       *
0800                5  ;*           BASIC               *
0800                6  ;*                               *
0800                7  ;*         TIM   OSBORN          *
0800                8  ;*                               *
0800                9  ;*********************************
0800               10  ;
0800               11  ;       APPLESOFT ENTRY POINTS
0800               12  ;
00B1               13  CHRGET    EPZ $B1             ;INCREMENT TXTPTR AND GET CHAR
                                                        ACTER
00B7               14  CHRGOT    EPZ $B7             ;CHRGET - INCREMENT
DEBB               15  CHKOPN    EQU $DEBB           ;CHECK FOR "(" @ TXTPTR
DEBE               16  CHKCOM    EQU $DEBE           ;CK. FOR "," @ TXTPTR
D995               17  DATA      EQU $D995           ;ADVANCE TXTPTR TO END OF STAT
                                                        EMENT
DD67               18  FRMNUM    EQU $DD67           ;EVALUATE EXPRESSION @ TXTPTR
                                                        + PUT IN FAC
E6FB               19  CONINT    EQU $E6FB           ;MODIFY FAC TO 8 BIT INTEGER
E752               20  GETADR    EQU $E752           ;MODIFY FAC TO 16 BIT INTEGER
DEC9               21  SYNERR    EQU $DEC9           ;PRINT SYNTAX ERROR MESSAGE
03F5               22  AMPERV    EQU $3F5            ;AMPERSAND VECTOR ADDRESS
0800               23  ;
0800               24  ;       MONITOR ENTRY POINT
0800               25  ;
FDB3               26  XAM       EQU $FDB3           ;HEX DUMP ROUTINE
0800               27  ;
0800               28  ;       DOS ENTRY POINT VECTOR
0800               29  ;
03D9               30  RWTS      EQU $03D9           ;READ-WRITE-TRACK-SECTOR
0800               31  ;
0800               32  ;       ZERO PAGE EQUATES
0800               33  ;
003C               34  A1L       EPZ $3C             ;2-BYTE MONITOR PARAMETER
003E               35  A2L       EPZ $3E             ;2-BYTE MONITOR PARAMETER
0050               36  LINNUM    EPZ $50
0073               37  HIMEM     EPZ $73
0800               38  ;
0800               39  ;
0800               40  ;       OTHER EQUATES
0800               41  ;
0001               42  READ      EPZ $01               AD OPERATION CODE
0002               43  WRITE     EPZ $02             ;WRITE OPERATION CODE
002C               44  COMMA     EPZ $2C
0052               45  R         EPZ $52
0057               46  W         EPZ $57
0058               47  X         EPZ $58
0800               48  ;
0800               49  ;
8000               50            ORG $8000
8000               51            OBJ $800
8000               52  ;SETVEC SETS HIMEM AND AMPERSAND VECTOR
8000               53  ;
8000 A9 0F         54  SETVEC    LDA #ENTRY
8002 8D F6 03      55            STA AMPERV+1
8005 85 73         56            STA HIMEM
8007 A9 80         57            LDA /ENTRY
8009 8D F7 03      58            STA AMPERV+2
800C 85 74         59            STA HIMEM+1
800E 60            60            RTS
800F               61  ;
800F               62  ;
800F               63  ;
800F               64  ;       MAIN PROGRAM
800F               65  ;
800F C9 52         66  ENTRY     CMP #R              ;READ REQUEST?
8011 D0 08         67            BNE WRITE?
8013 A9 01         68            LDA #READ           ;YES LD + SV READ
8015 8D D7 80      69            STA IBCMD           ;OPERATION CODE IN IOB
8018 4C 59 80      70            JMP RWTSCALL
801B C9 57         71  WRITE?    CMP #W              ;WRITE REQUEST?
801D D0 08         72            BNE XAM?            ;NO, CONTINUE
801F A9 02         73            LDA #WRITE          ;YES, LD + SV WRITE
8021 8D D7 80      74            STA IBCMD           ;OPERATION CODE IN IOB
8024 4C 59 80      75            JMP RWTSCALL
8027 C9 58         76  XAM?      CMP #X              ;MEMORY XAM REQUEST?
8029 F0 03         77            BEQ XAMINE          ;YES
802B 4C C9 DE      78            JMP SYNERR          ;NO, MUST BE ERROR
802E 20 B1 00      79  XAMINE    JSR CHRGET          ;GET NEXT CHARACTER AND
8031 20 BB DE      80            JSR CHKOPN          ;CHECK FOR "("
8034 20 67 DD      81            JSR FRMNUM          ;GET BEGINNING ADDRESS
8037 20 52 E7      82            JSR GETADR          ;CONVERT TO INTEGER
803A A5 50         83            LDA LINNUM
803C 85 3C         84            STA A1L             ;AND STORE FOR XAM
803E A5 51         85            LDA LINNUM+1
8040 85 3D         86            STA A1L+1
```

Listing 1 *(Continued)*

```
8042 20 BE DE    87              JSR CHKCOM          ;CHECK FOR ","
8045 20 67 DD    88              JSR FRMNUM          ;GET ENDING ADDRESS
8048 20 52 E7    89              JSR GETADR          ;CONVERT TO INTEGER
804B A5 50       90              LDA LINNUM
804D 85 3E       91              STA A2L             ;AND STORE FOR XAM
804F A5 51       92              LDA LINNUM+1
8051 85 3F       93              STA A2L+1
8053 20 B3 FD    94              JSR XAM
8056 4C 95 D9    95              JMP DATA            ;ADVANCE TXTPTR + RETURN TO BASIC
8059             96  ;
8059             97  ;
8059 20 B1 00    98  RWTSCALL JSR CHRGET          ;GET NEXT CHARACTER AND
805C 20 BB DE    99              JSR CHKOPN
805F 20 67 DD   100              JSR FRMNUM          ;GET TRACK NUMBER
8062 20 FB E6   101              JSR CONINT          ;CONVERT TO 1 BYTE INTEGER
8065 E0 23      102              CPX #$23            ;TRACK LESS THAN 35?
8067 30 03      103              BMI RWTS1           ;YES, CONTINUE
8069 4C C9 DE   104              JMP SYNERR          ;NO, DISPLAY MESSAGE
806C 8A         105  RWTS1      TXA
806D 8D CF 80   106              STA IBTRK           ;AND STORE IN PARMLIST
8070 20 BE DE   107              JSR CHKCOM          ;CHECK FOR ","
8073 20 67 DD   108              JSR FRMNUM          ;GET SECTOR NUMBER
8076 20 FB E6   109              JSR CONINT
8079 E0 10      110              CPX #$10            ;SECTOR LESS THAN 16?
807B 30 03      111              BMI RWTS2           ;YES, CONTINUE
807D 4C C9 DE   112              JMP SYNERR          ;NO, DISPLAY MESSAGE
8080 8A         113  RWTS2      TXA
8081 8D D0 80   114              STA IBSECT          ;AND STORE IN PARMLIST
8084 20 BE DE   115              JSR CHKCOM          ;CHECK FOR ","
8087 20 67 DD   116              JSR FRMNUM          ;GET BUFFER ADDRESS
808A 20 52 E7   117              JSR GETADR          ;CONVERT TO TWO BYTE INTEGER
808D A5 50      118              LDA LINNUM
808F 8D D3 80   119              STA IBBUFP          ;AND STORE IN PARMLIST
8092 C8         120              INY
8093 A5 51      121              LDA LINNUM+1
8095 8D D4 80   122              STA IBBUFP+1
8098 20 B7 00   123              JSR CHRGOT          ;LOAD ACCUM WITH (TXTPTR)
809B C9 2C      124              CMP #COMMA          ;IS IT A COMMA
809D D0 13      125              BNE NODRIVE         ;NO, DRIVE NOT SPECIFIED
809F 20 B1 00   126              JSR CHRGET          ;YES , ADVANCE TXTPTR
80A2 20 67 DD   127              JSR FRMNUM          ;GET DRIVE NUMBER
80A5 20 FB E6   128              JSR CONINT          ;AND CONVERT TO INTEGER
80A8 8A         129              TXA                 ;TRANSFER DRIVE # TO ACCUM
80A9 F0 04      130              BEQ RWTS3           ;SHOULD NOT BE = ZERO
80AB E0 03      131              CPX #$03            ;DRIVE NO. < 3?
80AD 30 05      132              BMI YESDRV          ;YES, CONTINUE
80AF 4C C9 DE   133  RWTS3      JMP SYNERR          ;NO, DISPLAY ERROR MESSAGE
80B2 A9 01      134  NODRIVE    LDA #$01            ;DEFAULT TO DRIVE #1
80B4 8D CD 80   135  YESDRV     STA IBDRVN          ;STORE IN PARMLIST
80B7 A9 DC      136              LDA #DCT            ;GET DCT ADDRESS(LOW)
80B9 8D D1 80   137              STA IBDCTP          ;AND STORE IN PARMLIST
80BC A9 80      138              LDA /DCT            ;DCT(HIGH)
80BE 8D D2 80   139              STA IBDCTP+1
80C1 A9 80      140              LDA /IOB            ;SET UP DCT ADDRESS PARAMETERS
80C3 A0 CB      141              LDY #IOB            ;FOR RWTS
80C5 20 D9 03   142              JSR RWTS
80C8 4C 95 D9   143              JMP DATA            ;ADVANCE TXTPTR TO END OF STAT
                                                      EMENT
80CB            144  ;                            ;AND RETURN TO BASIC
80CB            145  ;
80CB            146  ;INTERNAL STORAGE AREA
80CB            147  ;
80CB            148  ;
80CB            149  ;*****************************
80CB            150  ;*                           *
80CB            151  ;* INPUT/OUTPUT CONTROL BLK *
80CB            152  ;*                           *
80CB            153  ;*****************************
80CB            154  ;
80CB            155  IOB       EQU *
80CB            156  IBTYPE    DFS $01,$01         ;TYPE OF IOB
80CC            157  IBSLOT    DFS $01,$60         ;SLOT NUMBER
80CD            158  IBDRVN    DFS $01             ;DRIVE NUMBER
80CE            159  IBVOL     DFS $01,$00         ;VOLUME NO. 0=WILDCARD
80CF            160  IBTRK     DFS $01             ;TRACK NO.
80D0            161  IBSECT    DFS $01             ;SECTOR NO.
80D1            162  IBDCTP    DFS $02             ;POINTER TO DCT
80D3            163  IBBUFP    DFS $02             ;POINTER TO BUFFER
80D5            164  IBDUMY    DFS $02,$00         ;UNUSED
80D7            165  IBCMD     DFS $01             ;0=NULL,1=READ,2=WRITE,4=FORMAT
80D8            166  IBSTAT    DFS $01,$00       ;ERROR CODE (FOLLOWING CODES APPLY)
80D9            167  ;                             ;$10=WRITE PROTECT
80D9            168  ;                             ;$20=VOLUME MISMATCH
80D9            169  ;                             ;$40=DRIVE ERROR
80D9            170  ;                             ;$80=READ ERROR
80D9            171  IBSMOD    DFS $01,$00         ;ACTUAL VOLUME NO.
80DA            172  IOBPSN    DFS $01,$60         ;PREVIOUS SLOT NO.
80DB            173  IOBPDN    DFS $01,$01         ;PREVIOUS DRIVE NO.
80DC            174  ;
80DC            175  *********************************
80DC            176  *                               *
80DC            177  * DEVICE CHARACTERISTICS TABLE *
80DC            178  *                               *
80DC            179  *********************************
80DC            180  ;
80DC            181  DCT       EQU *
80DC            182  DEVTPC    DFS $01,$00         ;DEVICE TYPE CODE
80DD            183  PPTC      DFS $01,$01         ;PHASES PER TRACK
80DE            184  MONTC1    DFS $01,$EF         ;TIME COUNT
80DF            185  MONTC2    DFS $01,$D8         ;TIME COUNT
80E0            186  ;
80E0            187  ;
80E0            188              END
```

B= A valid Applesoft arithmetic expression representing the starting address of a 256-byte buffer.

D= A valid Applesoft arithmetic expression representing a drive number (1-2). The D parameter is optional and the program will default to a value of 1 if it is not coded in the &R or &W statement.

I have included for convenience a routine to call the monitor routine XAM directly from BASIC. This routine can be used to examine a hex dump of the RWTS buffer, or any range of memory in the Apple II. The syntax is as follows:

& X[B,E]

B= A valid Applesoft arithmetic expression representing the beginning of the range of memory to be dumped.

E= A valid Applesoft arithmetic expression representing the ending addresss of the range of memory to be dumped.

When XAM is dumping a range of memory, it may have more than a screen's worth of data, in which case you may wish to use the CTRL-S key to stop-list the dump. Any key will restart the dump. When the dump is completed, control returns back to the user program or the "]" prompt if this command is issued from the immediate mode.

How AMPERRWTS Works

AMPERRWTS uses the following subroutines, which are essential to its performance:

1. CHRGET $B1

Increments TXTPTR, loads the accumulator with the value TXTPTR is pointing at.

2. CHRGOT $B7

Same as CHRGET except CHRGOT does not increment the TXTPTR.

3. CHKOPN $DEBB

Checks at TXTPTR for an open parenthesis "(". Displays a syntax error if not found. Post-increments the TXTPTR through CHRGET.

4. CHKCOM $DEBE

Checks at TXTPTR for a comma. Displays a syntax error if not found. Post-increments the TXTPTR through CHRGET.

5. DATA $D995

Advances the TXTPTR to the end of the current Applesoft statement.

6. FRMNUM $DD67

Evaluates the arithmetic expressions @TXTPTR, and places the result in the FAC (floating point accumulator).

7. CONINT $E6FB

Evaluates the FAC and places the result in the X-register as an 8-bit X integer (0-255).

8. GETADR $E752

Evaluates the FAC and places the result into LINNUM ($50-$51) as a 16-bit integer (0-65535).

9. SYNERR $DEC9

Displays a syntax error message.

To install AMPERRWTS the user must BRUN the binary object file. This sets up the ampersand vector for Applesoft ($3F5). When the "&" is encountered, Applesoft will pass control to ENTRY at $800F. ENTRY first looks for an R, W, or X in the accumulator. If an R is found, a 1 is placed in IBCMD (the IOB command parameter), then a jump to the internal routine RWTSCALL is performed. The parameters are: 0 = null command — position head and start drive; 1 = read specified sector; 2 = write specified sector; 4 = format disk. For more information see page 97 of the DOS manual. If a W is found, a 2 is placed in IOBCMD and a JMP to RWTSCALL is performed. If an X is found, the internal routine XAMINE is JMPed to. If the accumulator is not equal to an R, W, or X, then a syntax error message is displayed.

RWTSCALL is the heart of AMPERRWTS. It evaluates the parameters passed between the left and right parentheses for the &R and &W commands, and sets up the IOB to make a proper call to RWTS. First, RWTSCALL advances the TXTPTR through a JSR to CHRGET. Then RWTSCALL JSRs to CHKOPN because we now expect, as defined in the above syntax diagrams, that we will find a left parenthesis ("("). If "(" is found, the TXTPTR will be advanced to point to what is expected to be an arithmetic expression that will result in a value between 0 and 34 (track number). This expression is evaluated through a JSR to FRMNUM, which places the result in the FAC. A JSR to CONINT converts the FAC to an 8-bit value and places it in the X-register. RWTSCALL then checks the X-register to make sure it is less than 35 (CPX #$23). If so, it is placed in IBTRK. Otherwise a syntax error message will be displayed. The next step is to JSR to CHKCOM to verify that a comma separates the T and S expressions. If the comma is not found, a syntax message is displayed. If the comma is found, the expression representing the S parameter is evaluated. This evaluation is done exactly as is the T expression, except a syntax error is displayed and processing ends if the expression results in a value greater than 15. The valid result is placed in IBSECT, the IOB sector parameter.

The next step is to JSR to CHKCOM, looking for a comma between the S and B expressions. The B expression is evaluated first, like the T and S expressions by a JSR to FRMNUM, but since it is a 16-bit value, we must use GETADR to translate the FAC to an integer. This places the result in LINNUM ($50-$51). Once in LINNUM, RWTSCALL moves the buffer address to IBBUFP, the IOB buffer address parameter. The D (drive number) expression is optional. If the routine finds a comma, it evaluates the D expression through a sequence of: 1. JSR CHRGET; 2. JSR FRMNUM; 3. JSR CONINT. If the drive number does not equal 1 or 2, a syntax error message is displayed and processing stops, unless it is stored at IBDRVN (the IOB drive number parameter). If RWTSCALL does not find a comma following the B parameter, then it defaults to a value of 1.

Next we must load the DCT address into IBDCTP (IOB DCT pointer parameter). Now we're ready to JSR to RWTS, which will load the requested sector into the 256-byte area starting at the address pointed to by IBBUFP. If we issued an &R command and wish to see a hex dump of the requested sector, we can issue an X & X command. The format should be & X[B,B + 255]; where B is equal to the B parameter in the &R command.

Having &R and &W around allows us to examine and zap any sector on a 16-sector Disk II diskette. One use of these subroutines is given in listing 3, "UNDELETE".

Listing 2

```
1 BFR = 8192: PRINT  CHR$ (4)"BLOAD AMPERRWTS.CODE,A$8000"
5  HOME : PRINT "V.T.O.C. (TRACK 17, SECT. 0)
10  & R(17,0,BFR)
15  GOSUB 300
20  FOR J = 1 TO 15
25  HOME : PRINT "CATALOG SECTOR ";J;" (TRACK 17,SECT. ";16 - J;")"
27  & R( PEEK (BFR + 1), PEEK (BFR + 2),BFR)
30  GOSUB 300
40  NEXT
50  END
300  FOR I = 1 TO 500: NEXT
305  & X(BFR,BFR + 191)
310  GET A$
320  HOME
330  & X(BFR + 192,BFR + 255)
340  GET A$
350  RETURN
```

***Listing 3:* UNDELETE**

```
1  PRINT  CHR$ (4);"BRUN AMPERRWTS.CODE,A$8000"
2  HOME
3 DT = 32: REM SEE FIGURE #3
5  PRINT "DRIVE #? ";: GET D$: PRINT D$
6  IF D$ < "1" OR D$ > "2" THEN  GOTO 5
7 D =  VAL (D$)
8  PRINT
10 BFR = 8192
20  & R(17,0,BFR,D): REM READ VTOC
30 T =  PEEK (BFR + 1):S =  PEEK (BFR + 2)
35  IF T = 0 AND S = 0 THEN  END
40  & R(T,S,BFR,D): REM  READ CATALOG
50  FOR J = 0 TO 210 STEP 35
55 TRK =  PEEK (BFR + 11 + J)
57  IF TRK = 0 THEN  END
60  IF TRK <  > 255 THEN  GOTO 100
70  FOR I = 0 TO 28
80  PRINT  CHR$ ( PEEK (BFR + 14 + J + I));
85  NEXT : REM  (I)
90  PRINT : PRINT "UNDELETE (Y) OR (N)? ";: GET A$: PRINT A$
91  PRINT
92  IF A$ <  > "Y" GOTO 98
94  POKE (BFR + 11 + J), PEEK (BFR + 11 + DT + J)
96  POKE BFR + DT + 11 + J,160
97  & W(T,S,BFR,D): GOTO 100
98  IF A$ <  > "N" THEN  GOTO 90
100  NEXT : REM  (J)
110  GOTO 30
```

How Undeleting is Possible

Whenever we delete a file from a diskette, the system replaces the track number of the first track/sector list (TRK in figure 1 and listing 3) with $FF (255). Then it places the track number in byte $1E (30) of the name filed in the file's directory entry (DT in figure 1 and listing 3). Track 17 sector 0 (VTOC) contains the track and sector number of the first directory entry in byte offsets 1 and 2 respectively. The directories also have a link to the next

Software Catalog

Name: **Darkstar**
System: Sinclair ZX81, Apple II
Memory: 16K - ZX81
48K RAM - Apple with DOS 3.3
Language: BASIC
Description: Solves darkroom problems relating to print density, magnification, lens opening, neutral density, color balance, filter factors, black-and-white paper type, black-and-white paper grade; variable contrast filters; black-and-white film development. Answers fully compensated for reciprocity and enlarger type. For both black-and-white and color darkroom application. Documentation offers many examples of use.
Price: $99.95 - Sinclair ZX81
$129.95 for Apple II
$550.00 for complete Sinclair package
Includes tape or disk and full documentation. Sinclair package also includes 12" black-and-white monitor and tape deck as well as Sinclair ZX81 with 16K RAM.
Author: Bob Nadler
Available:
F/22 Press
P.O. Box 141
Leonia, NJ 07605

Name: **Hardisk Accounting Series**
System: Apple II or Apple III
Memory: 64K, 128K
Language: UCSD Pascal
Hardware: 5-Megabyte hard disk
Description: *The Hardisk Accounting Series* from Great Plains Software is a menu-driven, double-entry accounting system for small business management and accounting. Each module of the *Hardisk Accounting Series* is interactive and includes complete audit trails. The program is written in UCSD Pascal and currently runs on the Apple II and Apple III with a hard disk. Special features include a password security system, flexible formatting for reports, menu-driven file utilities, budget tracking, and a management information system. Documentation includes cross-reference screen displays and step-by-step instructions. With the extensive data entry prompts and error checking, users will find the *Hardisk Accounting Series* totally comprehensive and very easy to operate and understand.
Price: $395.00/module for GL, AR, AP, Payroll
$595.00 for Inventory, Purchase Order Entry, Sales Order Entry, Job Costing.
Available:
Great Plains Software

Name: **OGI Fig-FORTH**
System: Apple II
Memory: 48K
Language: Machine Language and fig-FORTH
Hardware: Disk II
Description: This is an implementation of the FORTH programming language as defined by the Forth Interest Group (Fig). This high-level compiled language runs much faster than BASIC and requires less development and debugging time due to its structure and interaction with the user. This is a complete system including a screen disk containing a line editor, screen editor, assembler, decompiler, utilities, and more.
Price: $40.00
Includes diskette.
Author: Hal Clark
Available:
On-Going Ideas
RD #1, Box 810
Starksboro, VT 05487

Name: **A2-PBI Pinball**
System: Apple II or Apple II Plus
Memory: 48K
Language: Machine
Description: The ultimate arcade simulation program, which recreates the look and sound of a real pinball table down to the finest detail. *Pinball* offers ten user-selectable modes of play, and allows the user to create and save up to 100 custom modes of his/her own design. Forty parameters, each user-adjustable, control the characteristics of the game.
Price: $29.95
Includes 16-page adjustment manual, instruction card, high-score label.
Author: Bruce Artwick
Available:
Sublogic Communications Corporation
713 Edgebrook Drive
Champaign, IL 61820

Name: **Personal PEARL™**
System: CP/M
Memory: 56K
Language: Pascal
Hardware: 280 or 8080 at present
Description: *Personal PEARL* lets any user create his own custom program library. Designed for non-technical users as well as those with technical expertise. Combines forms generator, report generator, program generator and database manager. May be used with WordStar and SuperCalc.
Price: $295.00
Includes user manual.
Available:
Relational Systems Int'l. Corp.
P.O. Box 13850
Salem, OR 97309

Name: **Firebug**
System: Apple II or Apple II Plus
Memory: 48K
Language: Assembly Language
Hardware: Disk drive
Description: *Firebug* is a video game requiring fast manual response and quick thinking. It has high scoring potential and challenging maze situations from which to escape. *Firebug* includes colorful graphics and sound effects to heighten the player's involvement.
Price: $24.95
Includes disk, catalog, documentation.
Author: Silas Warner
Available:
Local dealers and distributors, nationwide and in Europe, and MUSE Software.

Name: **GRAPHVICS**
System: VIC-20
Memory: 3K or 8K Expander
Language: Assembly Language
Description: Adds 18 commands to VIC BASIC for creating both hi-resolution and multicolor objects on the VIC-20. Provides two screens: normal text screen and the graphics screen. Swap between the two with a function key. Press another function key to save the graphics screen to tape or diskette. All commands are available to BASIC programs.
Price: $25.00 ($30.00 foreign)
Includes manual and sample programs.
Author: Roy C. Wainwright
Available:
Abacus Software
P.O. Box 7211
Grand Rapids, MI 49510
(616) 241-5510

Name: **Atari Version -The Shattered Alliance**
System: Atari 800 (400 if enough memory)
Memory: 40K with BASICs cartridge
Language: BASIC
Hardware: Monitor, one disk drive
Description: Players have the choice of playing any of three battle scenarios on the fantasy world of Osgorth. Or they can simulate battles of ancient armies. There are a variety of units in the fantasy scenarios, including centaurs, elves, unicorns, dwarfs, lizardmen, and many more. Each is rated for strength, speed, and morale. The game moves at a quick pace due to Strategic Simulations' proprietary Rapidfire Movement system. Two-player and solitaire versions are included.
Price: $39.95
Includes diskette, rulebook, and data cards.
Author: John Lyon
Available:
Strategic Simulations Inc.
465 Fairchild Dr.
Suite 108
Mountain View, CA 94043
(415) 964-1353

Software Catalog
(continued)

Name: **Professional Investment System**
System: OS65U
Memory: 48K
Language: BASIC
Hardware: Ohio Scientific C-2 or C-3 Series
Description: This is an information management system for use by professional financial counseling and investment firms. It is fully menu driven and provides a variety of timely reports, as well as complete and up-to-date portfolios. The basic breakdowns of this system are Market Classifications, Stock/Bond Information, Portfolio, Transactions, and System Information.
Price: $1,500.00
Includes program disk and user's manual.
Available:
Electronic Information Systems, Inc.
P.O. Box 5893
Athens, GA 30604
(404) 353-2858

Name: **Labyrinth**
System: Apple II and Apple II Plus
Memory: 48K
Language: Assembly
Description: Beneath the City of Euqubud on the famed river Ippississim lie Prince Julian's mines — a labyrinth of hundreds of miles of tunnels and caves which was once the richest source of diamonds in the world. The mines are closed now. The yield became too meager and the cost too great, or so they said. Many men believe otherwise, and rumors abound of mysterious and terrifying creatures of the dark caverns which chased Prince Julian's company from the mines and now jealously guard their riches. Many courageous adventurers have ventured back into the deep seeking the fortune they believe to be there, but none has returned. They learned too late the terrifying secret of the labyrinth which ensures the doom of even the best prepared explorer: the walls of the mine are in constant motion, exposing entryways and sealing off exits, as its ghastly guardians render useless both map and compass with their evil engineering.
Price: $29.95
Includes software package.
Author: Scott Schram
Available:
Broderbund Software
1938 Fourth Street
San Rafael, CA 94901
(415) 456-6424
or your local computer store

Name: **CASDUP**
System: Atari 400/800
Memory: Less than 2K
Language: Assembly
Hardware: Cassette recorder
Description: This cassette program will duplicate all cassette-based BASIC, data, and machine-language files. If your Atari computer can read a tape, *CASDUP* will copy it.
Price: $20.00
Includes cassette tape with detailed 21-page instruction manual.
Author: Eric Verheiden
Available:
VERVAN Software
10072 Balsa Street
Cucamonga, CA 91730

Name: **Problem Solving in Everyday Math**
System: Apple II or Apple II Plus with Applesoft in ROM
Memory: 48K
Language: BASIC
Hardware: One disk drive, monitor, or TV
Description: This new process-oriented program takes a step-by-step approach to analyzing practical everyday mathematical problems. The diskette subjects are: how to solve problems; solving addition and multiplication problems; solving subtraction and division problems; other problem-solving processes such as reasoning without numbers, estimating solutions, and analyzing multiple-step problems.
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

OHIO SCIENTIFIC
THE WIZARD'S CITY — search for gold in the dungeons beneath the Wizard's city or in the surrounding forest. A dynamic adventure allowing progress in strength and experience. All OSI — cassette $12.95, disk $15.95.
OSI HARDWARE 15% OFF RETAIL PRICES!
GALACTIC EMPIRE — a strategy game of interstellar conquest and negotiation. Compete to discover, conquer, and rule an empire with the computer or 1-2 other players. C4P, C8P cassette $12.95, disk $15.95.
AIR TRAFFIC ADVENTURE — a real time air traffic simulation. C4P, C8P disks $15.95.
Plus S-FORTH, PACKMAN, CRAZY BOMBER, ADVENTURE, TOUCH TYPING, INTELLIGENT TERMINAL and more. Send for our free catalog including photos and complete descriptions.
(312) 259-3150
Aurora Software Associates
37 S. Mitchell
Arlington Heights
Illinois 60005

What would you give to have your Apple II able to configure to any peripheral?
SOFTECH MICROSYSTEMS

Software Catalog

(continued)

Name: **EDITRIX™ 1.0**
System: Apple II Plus
Memory: 48K
Language: Applesoft in ROM
Hardware: At least one disk drive with DOS 3.3 and one of the following printers: Anadex 9500/9501/9000/9001; Centronics 739/122/350/351; Epson MX-100/MX-80/MX-70; IDS 440G/445G/460G/560G; ITOH 8510; MPI 88G; NEC 8023; Okidata 82A/83A; Silentype
And one of the following parallel interface cards: Apple Standard/Centronics; CCS 7728; Epson APL; Grappler; Mountain CPS; Prometheus PRT-1/Versacard; SSS-AIO; TYMAC

Description: A new screen-oriented text editor featuring advanced text editing with ease of learning and use. To be used with Data Transforms' Graphtrix 1.3 to form the most powerful text editing/graphics screen dump system available.

Price: $75.00
Includes complete readable instructional manual and free updates as required.

Author: Steve Boker

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

Name: **K-DOS**
System: Atari 800
Memory: 24K minimum RAM
Language: Machine
Hardware: Disk

Description: This command-driven *K-DOS*™ is not only a more powerful and convenient DOS for the Atari 800, but it is completely compatible with the Atari 2.0S and other related software. In addition, *K-DOS* supports the Atari 850 handler which allows the use of printers and modems. *K-DOS* features a machine-language monitor which allows examination and alteration of memory in hexadecimal and displays ATASCII representation; interception of the break instruction does not crash the system, but takes the user back into *K-DOS*; new, powerful commands reserve and erase memory and may be executed when the BASIC or assembler cartridge is in control; *K-DOS* allows the user to create his own commands.

Price: $89.95
Includes 40-page handbook, disk, and pocket command summary card.

Author: Marcus Watts

Available:
K-Byte
1705 Austin
P.O. Box 456
Troy, Michigan 48099
(313) 524-9878

Name: **Financial Analysis Package**
System: Apple II or IBM Personal Computer
Memory: Apple II - 48K bytes RAM memory
IBM - 64K
Hardware: Apple II - Disk II disk controller and at least one Disk II disk drive.
IBM - 80-column video monitor, and a printer.

Description: The Execuware *Financial Analysis Package* provides sophisticated analysis for financial executives in determining whether to lease or buy, figuring loan and lease payment schedules, analyzing capital budgeting alternatives and determining depreciation schedules based on the Economic Recovery Act of 1981. The *Financial Analysis Package* is capable of performing nine versatile functions: 1. Loan Amortization Schedule; 2. Lease Amortization Schedule; 3. Depreciation Schedules; 4. Net Present Value Schedule; 5. Present Value of an Amount; 6. Internal Rate of Return; 7. Lease *versus* Buy Analysis; 8. Variable Rate Loan Schedule; and 9. Variable Payment Lease Schedule.

Price: $274.95
Includes easy to follow step-by-step instruction manual and diskette.

Author: Execuware™ Microcomputer Software Division of Aeronca, Inc.

Available:
Apple and IBM Personal Computer dealers

Name: **Korel The Robot**
System: Apple II
Memory: 64K
Language: Pascal

Description: *Korel The Robot* implements a Pascal-like compiler/debugger environment in which to learn and explore structured programming. By programming Korel (in an easy to master language) to avoid walls, escape mazes, etc., he will help you break down learning walls. A course disk is available which contains all the problems and solutions in the book and can be used to help develop a classroom curriculum or as a personal learning tool.

Price: $242.00
Includes book, Korel Simulator, course disks, and user's manual.

Author: Richard E. Pattis

Available:
Cybertronics Int'l. Inc.
999 Mt. Kemble Ave,
Morristown, NJ 07960

Name: **Jellyfish**
System: Apple II or Apple II Plus
Memory: 48K
Language: Machine (Assembly)

Description: Exciting undersea action! You must recover lost nuclear waste cannisters while avoiding schools of mutant jellyfish and octopuses. Can be played by one or two players and features high-resolution graphics.

Price: $29.95
Includes complete instructions

Author: Mike Burek

Available:
Sirius Software

Name: **TCS™ Total Inventory**
System: CP/M-compatible
Memory: 56K CP/M, 51K TPA, BDOS location: CC00
Language: MBASIC, DBOS, Agent, Machine Language

Description: *TCS™ Total Inventory* offers unique features such as the ability to handle multiple locations and departments. Reports can be keyed by department, location, or vendor with emphasis on management-oriented reporting for close control of stock and sales. When invoicing, the system automatically interrogates inventory for stock levels, item description, and pricing. Used as an interface with *TCS™ Total Receivables*..

Price includes software, manual, documenation, utilities, sample data.

Available:
TCS Software, Inc.
International Dealer Network

Name: **CP/M Fast Disk**
System: Apple II
Memory: 48K
Language: Assembly
Hardware: Legend 64K or 128KDE Card

Description: Add memory to CP/M and use every bit of RAM on the Legend 64K and/or 128KDE card(s). The *CP/M Fast Disk* allows you to read and write to an emulated disk and eliminates the need to wait for motor speed, track search and other time-consuming mechanical delays. As Legend cards are added, up to 512K of memory can be accessed.

Price: $69.95
Includes disk and manual.

Author: E.S. Tobin

Available:
Legend Industries, Ltd.
2220 Scott Lake Rd.
Pontiac, MI 48054
(313) 674-0953

Name: **Easi/Arima**
System: Apple II
Memory: 48K
Language: Applesoft
Hardware: One or more disk drives and a printer

Description: A forecasting package for stock and commodity traders, which automates the approach developed by Box and Jenkins so that no statistical experience is necessary. It is usually able to forecast stock and commodity prices within 2% over a one- to five-day horizon. Reads Compu-Trac databases directly and includes programs to create, correct, print, and transform its own databases. Right- or left-hand software keypads are also part of the package.

Price: $300.00
Includes one year of maintenance and updates.

Author: Eric Weiss, Ph.D.

Available:
The Winchendon Group
P.O. Box 10114
Alexandria, VA 22310

MICRO™

MICRObits

Deadline for MICRObits: 20th of second month before publication; i.e., August 20th for October issue. Send typewritten copy (40-word limit) with $25.00 per insertion. (Subscribers: first ad at $10.00.)

6800/6809 Software

Includes compatible single-user, multi-user and network-operating systems, compilers, accounting and word processing packages. Free catalog.

Software Dynamics
2111 W. Crescent, Sta. G
Anaheim, CA 92801

Kids Can Touch

A Child's Guide to the Apple II Plus Computer. A primer for all ages 8 to 88. Adult supervision not required. Forty-eight pages packed with BASIC programming, how computers work, and their history. $4.95 plus $1.00 shipping.

Kids Can Touch
24 Beechwood Road
Summit, NJ 07901

Scientific Calculator

OSI 8K program with easy entry, input work sheet display, totals in hex and dec. Hex and dollar modes selectable. 8K tape $15. Send SASE for data sheet and free utility program listing.

Harry Hawkins
Box 4432
Burton, SC 29902

Dynamite PET/CBM Accessories!

Write-protect switches/indicators for 2040/4040 disk drives. Real world software at low cost. 2114 RAM adapter (replaces obsolete 6550s) and 4K memory expansion for "old" 8K PETs. Hundreds of satisfied customers. Write for *free* catalog!

Optimized Data Systems
Dept. M, Box 595
Placentia, CA 92670

New! The Aerobics Master

Here's a day-by-day diary for exercisers designed by a runner. Tracks progress of a variety of exercise activities. 48K, DOS 3.3 and ROM Applesoft required. Warranteed upon registration. SASE for more information. Introductory price $22.95 - includes shipping.

Free Lance Ink
1806 Wickham
Royal Oak, MI 48073

(Continued on page 117)

What would you give to have TURTLEGRAPHICS, with automatic scaling, and four graphic modes, including HIRES and LORES, on your Apple II?

SOFTECH MICROSYSTEMS

MICRO™ Hardware Catalog

Name: **Apple Computer Case**
Description: A carrying and storage case designed for the Apple II and two disk drives, that may remain connected to a monitor and/or printer with the cover closed or removed completely. The case features a foam-padded interior, non-metalic hold-down strap, removable locking cover, no-slip bumpers and sturdy ABS plastic end-cap construction.
Price: $64.00
Includes UPS delivery to 48 states.
Available:
Fiberbilt
601 West 26th St.
New York, NY 10001

Name: **Fully Extended Wirewrap Prototype Board**
System: Apple II
Description: Size: 2.8" × 10.7" 2-layer pc. Capacity: up to 57 16-pin or 12 40-pin sockets or any combination in between. Carefully designed layout for minimum electrical noise, low impedance, and maximum versatility in layout of ICs, capacitors, and other discretes.
Price: $45.00 (CA residents add 6% tax)
Includes extensive wirewrap technique documentation.
Available:
Spectrum Systems
P.O. Box 2262
Santa Barbara, CA 93120

Name: **Apple Crate™**
System: Apple
Description: *Apple Crate* is a lightweight case for the Apple computer and disk drives.
Price: $92.00
Includes computer case with accessory pocket and case for two disk drives.
Available:
ABCOM Corporation
16005 Sherman Way
Suite 105
Van Nuys, CA 91409
or your local computer store

Name: **Micro View**
System: Almost any 8-bit microcomputer
Description: If you program, engineer, test, or repair microprocessor systems, you can debug fast, thoroughly, and with ease using this new invention. Features include: extensive 256-LED display; shows activity in real time; shows address, data, I/O, program flow, and timing information; traces instabilities; gives telescopic and microscopic views; works at full CPU speed; soldered-in CPU OK; easy to learn and easy to use; based on principle of visual pattern recognition; supports most micros; quick setup; rugged; portable; as general purpose as an oscilloscope. Also good for teaching. Lowest cost tool in its class.
Price: $995.00
Includes complete instrument, personality pack, manual, Micro Chart.
Available:
Micro Logic Corp.
P.O. Box 174
100 2nd St., Dept. MC
Hackensack, NJ 07602
(201) 342-6518

Name: **Full-View 80 Display Card**
System: Atari 800
Memory: 32K and up
Language: Letter Perfect (80-Column Version), BASIC, Machine Language
Description: The *Full-View 80* provides 80-column capability for the Atari 800 with upper and lower case characters, while retaining the normal Atari 40-column/graphics mode. Switching is accomplished *via* an on-board soft-switch. The *Full-View 80* fits in slot 3. With the 32K *Memory Plus* in slot 2 the Atari 800 becomes a powerful 80-column, 48K wordprocessor.
Price: $349.00
Includes all features - no extras.
Available:
Bit 3 Computer Corporation
8120 Penn Ave. S. #548
Minneapolis, MN 55431

Name: **APX-800 APPLE-VERTER**
System: Apple II
Description: VHF RF modulator (Ch. 7-10 tunable). Designed to mount inside the Apple II computer. Installs in seconds. Operates above normal computer harmonics. Exhibits unusually high stability. Can be used with other computer systems with power sources between 8 and 24 VDC.
Price: $29.75
Includes direct connect 10' antenna cable.
Available:
ATV Research
13th and Broadway
Dakota City, NE 68731

Name: **Leggs**
Hardware: MX-80 Printer
Description: *Leggs*™ is a new 'stand' for the Epson MX-80 printer allowing the paper to be put underneath it. *Leggs* installs in seconds in existing tapered holes and requires no tools or changes to the printer. Room for three-inches of paper under the printer. Made of clear acrylic plastic — cheaper than any other stand you can buy.
Price: $15.00
Includes set of four legs.
Available:
Argus, Inc.
Box 9777
Baltimore, MD 21204
or your local computer store

Name: **Cool Stack™ Printer Pal™**
System: Apple II
Description: The *Cool Stack-Sentry II* offers the features of locking, cooling, easy tilt action access, and efficient organization and storage for the Apple II computer system, all in one compact unit. The precision all-steel construction is designed for optimum strength and durability. An attractive textured finish is color matched to the Apple II. The *Printer Pal* stores and feeds printer paper from underneath the printer, offering space-saving convenience and efficiency. Different models are available for other printers.
Price: $175.00 Cool Stack-Sentry II complete.
$29.95 Printer Pal-Model P80 complete with paper support brackets.
Available:
FMJ, Inc.
P.O. Box 5281
Torrance, CA 90510
(213) 325-1900

Name: **AI13 Analog Input System**
System: Apple II
Description: The *AI13* is a high-performance, 12-bit Analog-to-Digital Input System including software. The *AI13* gives the Apple the ability to make precision voltage measurements. The hardware is distinguished by the following: 16 separate input channels, 8 software-selectable voltage ranges from ±5 volts down to 0-100 millivolts, 20 microsecond conversion time, 12-bit (0.024%) resolution.
Price: $550.00
Includes cabling, software diskette, and comprehensive manual.
Available:
Interactive Structures Inc.
146 Montgomery Ave.
Bala Cynwyd, PA 19004

Name: **Starwriter F-10 Wheel Printer**
Description: High-performance Daisy Wheel printer. Uses industry-standard ribbon cartridges. Standard parallel or RS-232-C interfaces and, by jumper selection, ET X/ACT, X-ON/X-OFF protocols provide maximum flexibility and installation ease. Has extensive built-in wordprocessor functions. Low noise. Choice of friction feed or bi-directional tractor feed. Universal power supply. 30.8 lbs.
Price: $1995.00
Available:
Leading Edge Products
225 Turnpike St.
Canton, MA 02021

MICRO™

MICRObits

(Continued from page 115)

TRS-80 Color Computer Game

MUNCH-O is probably the most challenging arcade-type maze game available for the color computer, with two mazes, nine changing color schemes, sound, and at certain levels, a maze with invisible walls. Minimum 4K. Joystick required. $12.95 cassette.

Mike-Ro Products
P.O. Box 21
Lake Orion, MI 48035

TRS-80 Color Computer

Expand your 4K system to 16K for better color graphics. Full instructions/documentation in each kit. Price of kit is $29.95. Compare at $99.00. Allow two to three weeks for delivery. Check/money orders OK. $3.00 postage/handling charge extra.

Dick Williams
Whispering Pines
Lane 2-1
Derry, NH 03038
(603) 432-3634

OSI Super Defender

Super Defender is an all machine-code game just like the arcades. The mountains roll by as your scanner shows what's comming at you. Protect your humanoids from being snatched by the landing crafts. $14.95 for tape or 5¼" disk.

DMP Systems
319 Hampton Blvd.
Rochester, NY 14612

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 So. General McMullen Dr.
San Antonio, TX 78237

Apple Education

Physics: 11 diskettes, 75 programs — $200. May be ordered separately. *Happy Face* (four word games) — $15; *Dinosaurs* — $15; *Aquarium* — $25; *Christian Education* — $15. Above programs have extensive hi-res graphics. *Peachy Writer* text editor — $24.95; *Grade Reporter* — $19.95. VISA/MC. Free catalog.

Cross Educational Software
Box 1536 M
Ruston, LA 71270
(318) 255-8921

MICRO

What would you give to develop programs for the IBM PC, TRS 80 Model II, T.I. 99/4 Home Computer, and Xerox 820 on your Apple II?

SOFTECH MICROSYSTEMS

MICRO™

Dr. William R. Dial
438 Roslyn Avenue
Akron, OH 44320

6809 Bibliography

48. 80 Microcomputing, Issue No. 27 (March, 1982)

Cook, Douglas R., "Colormon," pg. 212-213.
A monitor for the 6809-based TRS-80 Color Computer.

Wood, James W., "You Light Up My Life," pg. 330.
Teach the physics of light with this program for interference patterns using the TRS-80 Color Computer.

49. CSRA Computer Club Newsletter (March, 1982)

Gresham, Jim, "Color Computer Ramblings," pg. 3.
Miscellaneous information on the TRS-80 Color Computer including some important ROM addresses.

50. Interface Age 7, Issue 4 (April, 1982)

O'Connor, Patrick and Leah, "Game Corner," pg. 22, 149.
Breakout, a game for the 6809-based TRS-80 Color Computer; discussion and listing.

51. MICRO No. 47 (April, 1982)

Capouch, Brian, "Structured Programming in BASIC09," pg. 45-49.
BASIC09 is a programming language available for 6800/6809-based OS-9 operating systems.

Tenny, Ralph, "Extensions to the C-Bug Monitor," pg. 51-55.
Two valuable debugging functions are added to CBUG, a monitor for the 6809-based TRS-80 Color Computer.

Walker, Gregory and Whiteside, Tom, "Multiprecision Addition — A Comparison of 6809 and 6502 Programming," pg. 57-59.
A comparison using 32-bit addition routines to demonstrate several advantages of programming the MC6809 over the 6502.

Puckett, Dale, "FLEX: An Operating System for the 6809," pg. 61-65.
FLEX is a widely supported operating system for 6800- and 6809-based microcomputers. Its history, features and applications are discussed.

Wright, Loren, "PET Vet," pg. 71-72.
A discussion of the SuperPET, a micro based on both the 6809 and 6502 processors.

Staff, "6809 Microprocessor," pg. 121-122.
A reference sheet for the 6809 microprocessor, with pinouts, programming model, indexed/indirect codes, instruction codes, and a table of branched instruction codes.

52. Commodore Magazine 3, No. 1 (February, 1982)

Staff, "Accessing the SuperPET RS-232 Port," pg. 58-59.
Procedure for the 6809/6502-based SuperPET micro.

53. Compute! 3, No. 11 (November, 1981)

Wilkinson, Terry, "SuperPET's Super Software," pg. 28-36.
Discussion of software for the 6809-based SuperPET.

54. Interface Age 6, No. 10 (October, 1981)

Doonan, Dennis, "MC6809 Cookbook," pg. 152.
"How-To" for the 6809.

55. CSRA Computer Club Newsletter (April, 1982)

Gresham, Jim, "Color Computer Ramblings," pg. 1-3.
Miscellaneous information on the 6809-based Color Computer of Radio Shack.

56. Personal Computer World 5, No. 4 (April, 1982)

Kewney, Guy, "16-Bit Battle Takes Shape," pg. 60-61.
Discussion and comparison of various microprocessors including the 6809, 68000, 8088, 8086, etc.

57. BYTE 7, No. 3 (March, 1982)

Barden, William Jr., "Build a Half-Year Clock for the Color Computer," pg. 100-122.
A hardware article for users of the Radio Shack Color Computer based on the 6809.

58. Personal Computer World 5, No. 5 (May, 1982)

Staff, "Hitachi Peach," pg. 104-108.
A detailed description of the 6809-based Hitachi MB-6890 Personal Computer Basic Master Level 3 micro, also now known as the Hitachi Peach.

59. BYTE 7, No. 5 (May, 1982)

Kocher, Christopher P. and Keith, Michael, "Six Personal Computers from Japan," pg. 61-102.
Information is given on several 6809-based micros including the Canon CX-1, the Hitachi MB-6890, and the Fujitsu FM-8 (two 6809 processors).

Barden, William Jr., "Ports of Entry and Soft Breezes for the Color Computer and Model III," pg. 162-198.
A hardware article for the Radio Shack Color Computer including an anemometer and other remote sensing projects.

60. MICRO No. 48 (May, 1982)

Walker, Gregory and Whiteside, Tom, "Memory Moves with the 6502 and 6809," pg. 19-22.
Advantages of the 6809's direct page addressing and 16-bit registers are shown in a comparison of 6502 and 6809 memory moves.

Dial, Wm. R., "6809 Bibliography," pg. 99.
Some 17 references to the literature on the 6809 microprocessor are cited.

61. BYTE 7, No. 4 (April, 1982)

Harrington, John, "A Simple Multiprocessor Implementation," pg. 464-471.
Multiprocessing with 6809-based systems.

Field, Tim, "Easy Entry Program for Radio Shack's Color Computer," pg. 482-487.
A short BASIC program that will help enter machine-language programs for the 6809-based Radio Shack Color Computer.

62. Commodore Magazine (April/May, 1982)

Staff, "Commodore News: SuperPET," pg. 17-18.
Questions and answers related to the 6809-based SuperPET.

Staff, "SuperPET *vs.* IBM," pg. 21-22.
A close-up look at these competing microcomputers.

63. 80 Microcomputing No. 29 (May, 1982)

Commander, Jake, "Spiromania," pg. 88-96.
Use your 6809-based Color Computer to run a graphics program combining a constantly changing angle with a constantly changing radius to form a spirograph.

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

Please send coupon to: **Apple Upgrade Dept.**
P.O. Box 27179, San Diego, CA 92128
(800) 225-0419 (800) 225-1592 in Calif.

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.
My proof of purchase is
☐ invoice ☐ receipt ☐ disk label ☐ other

Signature_____
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.

MICRO™

Dr. William R. Dial
438 Roslyn Avenue
Akron, OH 44320

6502 Bibliography

1. Atari Computer Enthusiasts (March, 1982)

Anon., "Tape Handling," pg. 9.

Discussion of the protocol for tape handling on the Atari 810 Tape Drive, including mark and space frequencies, 600 baud rate, data files, etc.

2. Microcomputer Printout 3, No. 4 (March, 1982)

Butterfield, Jim, "Disk Doctor," pg. 64-67.

Two useful disk routines for the CBM micros. One is for checking the veracity of a particular disk and the other for changing your drive number with recourse to a soldering iron.

3. MICRO No. 46 (March, 1982)

Neils, Jody, "KIM Bouncy Keypad Cure," pg. 77-80.

A 94-byte program to eliminate the annoying key-bounce and prolong the life of the KIM-1 keypad.

4. Commodore Magazine (February, 1982)

Tomczyk, Michael S., "The VIC Magician," pg. 39-51.

The VIC-20 as a super calculator, a light pen drawing program, and other routines for the VIC.

5. Microcomputing 6, No. 3 (March, 1982)

Multer, Kent A., "Freedom from Text Editor Tyranny," pg. 72-78.

A friendly word processor for the Atari, 'Runoff,' including an interface for the Epson MX-80 printer.

6. RTTY Journal 30, No. 3 (March, 1982)

Hammon, George, "Apple Computers and RTTY," pg. 4-6.

How to put the Apple on an amateur radioteletype, transmitting and receiving at 110 baud ASCII or at 60, 67, 75, or 100 WPM Baudot.

7. RTTY Journal 30, No. 3 (March, 1982)

Johnson, Thomas C., "Oscar Pathfinder," pg. 46-51.

A colorful way to track amateur radio satellites using the Apple, in Applesoft BASIC.

8. Apple Gram 4, No. 3 (March, 1982)

David, Jill, "Using Your Pascal Editor as a Word Processor," pg. 23-30.

The Apple Pascal program RJUSFY may be used with your text files to make your Pascal system work like a word processor.

9. Creative Computing 8, No. 3 (March, 1982)

Lubar, David, "Table Lookup," pg. 160-166.

A 6502 machine-language routine to handle lookup tables, with examples for the Apple.

10. Byte 7, No. 3 (March, 1982)

Starbuck, Bill, "Epson MX-80 Print Control for the Apple," pg. 166-170.

A program to set up your Apple for the various typing modes.

Microcomputer Information Resources

MICRO wants to make sure our readers are aware of the excellent sources of microcomputer bibliographic information that are available. *Microcomputer Index* is a periodical that provides a subject index for a cross-section of popular microcomputer magazines. It includes abstracts. Published by Microcomputer Information Services, 2646 El Camino Real #247, Santa Clara, CA 95051, *Microcomputer Index* has put more than 10,000 articles, indexed and abstracted from 23 periodicals, on line with Lockheed's DIALOG service. Probably the best single source of bibliographic information in book form about articles published in microcomputer magazines is *The Index*. Compiled by W.H. Wallace and published by Missouri Indexing, Inc. (P.O. Box 301, St. Ann, MO 63074), *The Index* is not only comprehensive, but so well organized that using it is a pleasure. Another helpful publication is *Micro ... Publications In Review* (Vogeler Publishing Inc., P.O. Box 489, Arlington Heights, IL 60006). This periodical reproduces the tables of contents of the latest issues of the major microcomputer magazines, and provides a subject index.

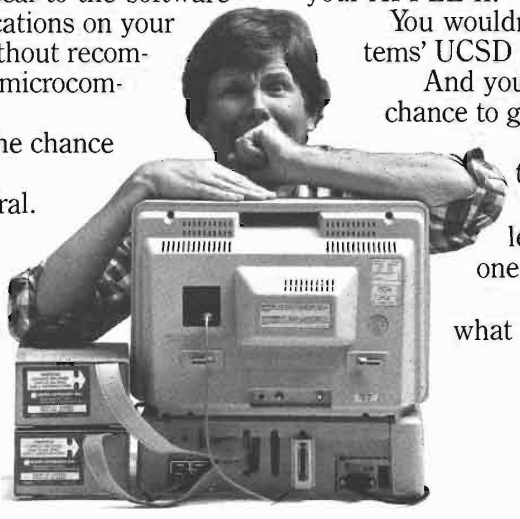
MICRO

What would you do if you missed out on this offer?

You'd have missed out on a chance to upgrade from Apple Pascal to the software that lets you write applications on your Apple II that will run without recompilation on virtually *any* microcomputer. Period.

You'd have missed the chance to have your APPLE II configure to any peripheral.

You wouldn't have TURTLEGRAPHICS, with automatic scaling and four graphic modes, including HIRES and LORES.



You wouldn't have unlimited UNITS on your APPLE II.

You wouldn't have SofTech Microsystems' UCSD p-System. For only $295.

And you'd never again have the chance to get it at that price.

Think about it. But don't think too long. Because you've only got one month left to take advantage of this one-time-only offer.

And if you miss it this time, what will you do?

SOFTECH MICROSYSTEMS

Please send coupon to: **Apple Upgrade Dept.**
P.O. Box 27179, San Diego, CA 92128
(800) 225-0419 (800) 225-1592 in Calif.

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.
My proof of purchase is ☐ invoice ☐ receipt ☐ disk label ☐ other

Signature_____
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.

AARDVARK — THE ADVENTURE PLACE

ADVENTURES FOR OSI, TRS-80, TRS-80 COLOR, SINCLAIR, PET, VIC-20

ADVENTURES — Adventures are a unique form of computer game. They let you spend 30 to 70 hours exploring and conquering a world you have never seen before. There is little or no luck in Adventuring. The rewards are for creative thinking, courage, and wise gambling — not fast reflexes.

In Adventuring, the computer speaks and listens to plain English. No prior knowledge of computers, special controls, or games is required so everyone enjoys them—even people who do not like computers.

Except for Quest, itself unique among Adventure games, Adventures are non-graphic. Adventures are more like a novel than a comic book or arcade game. It is like reading a particular exciting book where you are the main character.

All of the Adventures in this ad are in Basic. They are full featured, fully plotted adventures that will take a minimum of thirty hours (in several sittings) to play.

Adventuring requires 16k on Sinclair, TRS-80, and TRS-80 Color. They require 8k on OSI and 13k on VIC-20. Sinclair requires extended BASIC.

TREK ADVENTURE by Bob Retelle — This one takes place aboard a familiar starship and is a must for trekkies. The problem is a familiar one — The ship is in a "decaying orbit" (the Captain never could learn to park!) and the engines are out (You would think that in all those years, they would have learned to build some that didn't die once a week). Your options are to start the engine, save the ship, get off the ship, or die. Good Luck.

Authors note to players — I wrote this one with a concordance in hand. It is very accurate — and a lot of fun. It was nice to wander around the ship instead of watching it on T.V.

CIRCLE WORLD by Bob Anderson — The Alien culture has built a huge world in the shape of a ring circling their sun. They left behind some strange creatures and a lot of advanced technology. Unfortunately, the world is headed for destruction and it is your job to save it before it plunges into the sun!

Editors note to players — In keeping with the large scale of Circle World, the author wrote a very large adventure. It has a lot of rooms and a lot of objects in them. It is a very convoluted, very complex adventure. One of our largest. Not available on OSI.

HAUNTED HOUSE by Bob Anderson — This one is for the kids. The house has ghosts, goblins, vampires and treasures — and problems designed for the 8 to 13 year old. This is a real adventure and does require some thinking and problem solving — but only for kids.

Authors note to players — This one was fun to write. The vocabulary and characters were designed for younger players and lots of things happen when they give the computer commands. This one teaches logical thought, mapping skills, and creativity while keeping their interest.

DERELICT by Rodger Olsen and Bob Anderson — For Wealth and Glory, you have to ransack a thousand year old space ship. You'll have to learn to speak their language and operate the machinery they left behind. The hardest problem of all is to live through it.

Authors note to players — This adventure is the new winner in the "Toughest Adventure at Aardvark Sweepstakes". Our most difficult problem in writing the adventure was to keep it logical and realistic. There are no irrational traps and sudden senseless deaths in Derelict. This ship was designed to be perfectly safe for its' builders. It just happens to be deadly to alien invaders like you.

NUCLEAR SUB by Bob Retelle — You start at the bottom of the ocean in a wrecked Nuclear Sub. There is literally no way to go but up. Save the ship, raise her, or get out of her before she blows or start WWIII.

Editors note to players — This was actually plotted by Rodger Olsen, Bob Retelle, and someone you don't know — Three of the nastiest minds in adventure writing. It is devious, wicked, and kills you often. The TRS-80 Color version has nice sound and special effects.

EARTHQUAKE by Bob Anderson and Rodger Olsen — A second kids adventure. You are trapped in a shopping center during an earthquake. There is a way out, but you need help. To save yourself, you have to be a hero and save others first.

Authors note to players — This one feels good. Not only is it designed for the younger set (see note on Haunted House), but it also plays nicely. Instead of killing, you have to save lives to win this one. The player must help others first if he/she is to survive — I like that.

PYRAMID by Rodger Olsen — This is one of our toughest Adventures. Average time through the Pyramid is 50 to 70 hours. The old boys who built this Pyramid did not mean for it to be ransacked by people like you.

Authors note to players — This is a very entertaining and very tough adventure. I left clues everywhere but came up with some ingenous problems. This one has captivated people so much that I get calls daily from as far away as New Zealand and France from bleary eyed people who are stuck in the Pyramid and desperate for more clues.

QUEST by Bob Retelle and Rodger Olsen — THIS IS DIFFERENT FROM ALL THE OTHER GAMES OF ADVENTURE!!!! It is played on a computer generated map of Alesia. You lead a small band of adventurers on a mission to conquer the Citadel of Moorlock. You have to build an army and then arm and feed them by combat, bargaining, exploration of ruins and temples, and outright banditry. The game takes 2 to 5 hours to play and is different each time. The TRS-80 Color version has nice visual effects and sound. Not available on OSI. This is the most popular game we have ever published.

MARS by Rodger Olsen — Your ship crashed on the Red Planet and you have to get home. You will have to explore a Martian city, repair your ship and deal with possibly hostile aliens to get home again.

Authors note to players — This is highly recommended as a first adventure. It is in no way simple—playing time normally runs from 30 to 50 hours — but it is constructed in a more "open" manner to let you try out adventuring and get used to the game before you hit the really tough problems.

ADVENTURE WRITING/DEATHSHIP by Rodger Olsen — This is a data sheet showing how we do it. It is about 14 pages of detailed instructions how to write your own adventures. It contains the entire text of Deathship. Data sheet - $3.95. NOTE: Owners of OSI, TRS-80, TRS-80 Color, and Vic 20 computers can also get Deathship on tape for an additional $5.00.

PRICE AND AVAILABILITY:

All adventures are $14.95 on tape except Earthquake and Haunted House which are $9.95. Disk versions are available on OSI and TRS-80 Color for $2.00 additional.

Please specify system on all orders

ALSO FROM AARDVARK — This is only a partial list of what we carry. We have a lot of other games (particularly for the TRS-80 Color and OSI), business programs, blank tapes and disks and hardware. Send $1.00 for our complete catalog.

AARDVARK - 80
2352 S. Commerce, Walled Lake, MI 48088
(313) 669-3110

Phone Orders Accepted 8:00 a.m. to 4:00 p.m. EST. Mon.-Fri.

TRS-80 COLOR **SINCLAIR** **OSI** **VIC-20**

6522 (VIA)

6522 — Versatile Interface Adaptor

— Manufactured by MOS, Synertek, Rockwell
— As its name indicates, it can be used in a wide variety of microcomputer interface applications, including:

keyboard
cassette
disk drive controller
teletype
printer
IEEE-488 controller

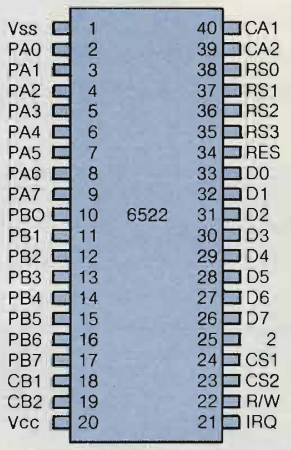
— Commercial computers include one or more of these devices, including PET/CBM, AIM 65, and SYM-1. Many commercial expansion boards include a 6522.

— Features:

- two 8-bit bidirectional ports with handshaking
- two internal timers
- serial/parallel shift register
- input data latching

— Because of its general purpose design, the 6522 can be used with most microcomputers including the 65XX and 68XX families.

6522 Pinout

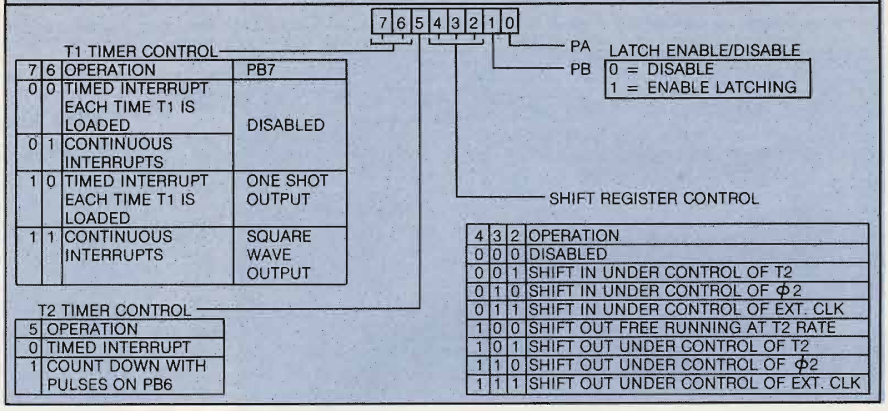


6522 Registers

Register Number		Register Designation	Description	
			Write	Read
0	$00	ORB/IRB	Output register B	Input register B
1	$01	ORA/IRA	Output register A	Input register A
2	$02	DDRB	Data direction register B	
3	$03	DDRA	Data direction register A	
4	$04	T1C-L	T1 low-order latches	T1 low-order counter
5	$05	T1C-H	T1 high-order counter	
6	$06	T1L-L	T1 low-order latches	
7	$07	T1L-H	T1 high-order latches	
8	$08	T2C-L	T2 low-order latches	T2 low-order latches
9	$09	T2C-H	T2 high-order counter	
10	$0A	SR	Shift register	
11	$0B	ACR	Auxilliary	
12	$0C	PCR	Peripheral control register	
13	$0D	IFR	Interrupt flag register	
14	$0E	IER	Interrupt enable register	
15	$0F	ORA/IRA	Same as register 1, except no handshaking	

6522 (VIA)

Reg 11 ($0B) — Auxiliary Control Register (ACR)

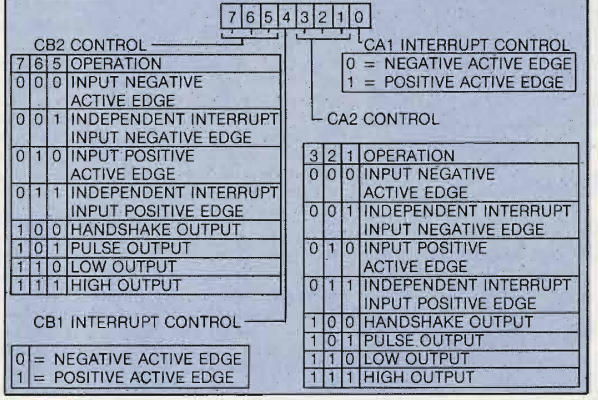


7	6	OPERATION	PB7
0	0	TIMED INTERRUPT EACH TIME T1 IS LOADED	DISABLED
0	1	CONTINUOUS INTERRUPTS	DISABLED
1	0	TIMED INTERRUPT EACH TIME T1 IS LOADED	ONE SHOT OUTPUT
1	1	CONTINUOUS INTERRUPTS	SQUARE WAVE OUTPUT

5	OPERATION
0	TIMED INTERRUPT
1	COUNT DOWN WITH PULSES ON PB6

4	3	2	OPERATION
0	0	0	DISABLED
0	0	1	SHIFT IN UNDER CONTROL OF T2
0	1	0	SHIFT IN UNDER CONTROL OF Φ2
0	1	1	SHIFT IN UNDER CONTROL OF EXT. CLK
1	0	0	SHIFT OUT FREE RUNNING AT T2 RATE
1	0	1	SHIFT OUT UNDER CONTROL OF T2
1	1	0	SHIFT OUT UNDER CONTROL OF Φ2
1	1	1	SHIFT OUT UNDER CONTROL OF EXT. CLK

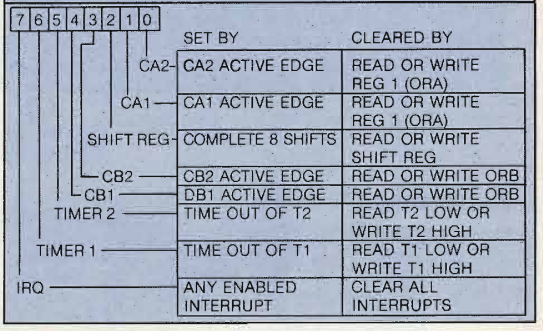
Reg 12 ($0C) — Peripheral Control Register (PCR)



7	6	5	OPERATION
0	0	0	INPUT NEGATIVE ACTIVE EDGE
0	0	1	INDEPENDENT INTERRUPT INPUT NEGATIVE EDGE
0	1	0	INPUT POSITIVE ACTIVE EDGE
0	1	1	INDEPENDENT INTERRUPT INPUT POSITIVE EDGE
1	0	0	HANDSHAKE OUTPUT
1	0	1	PULSE OUTPUT
1	1	0	LOW OUTPUT
1	1	1	HIGH OUTPUT

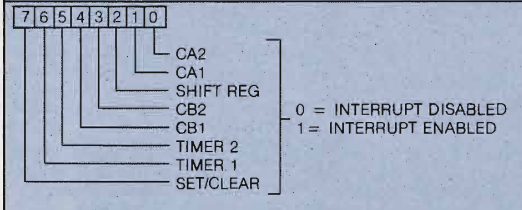
3	2	1	OPERATION
0	0	0	INPUT NEGATIVE ACTIVE EDGE
0	0	1	INDEPENDENT INTERRUPT INPUT NEGATIVE EDGE
0	1	0	INPUT POSITIVE ACTIVE EDGE
0	1	1	INDEPENDENT INTERRUPT INPUT POSITIVE EDGE
1	0	0	HANDSHAKE OUTPUT
1	0	1	PULSE OUTPUT
1	1	0	LOW OUTPUT
1	1	1	HIGH OUTPUT

Reg 13 ($0D) — Interrupt Flag Register (IFR)



	SET BY	CLEARED BY
CA2	CA2 ACTIVE EDGE	READ OR WRITE REG 1 (ORA)
CA1	CA1 ACTIVE EDGE	READ OR WRITE REG 1 (ORA)
SHIFT REG	COMPLETE 8 SHIFTS	READ OR WRITE SHIFT REG
CB2	CB2 ACTIVE EDGE	READ OR WRITE ORB
CB1	DB1 ACTIVE EDGE	READ OR WRITE ORB
TIMER 2	TIME OUT OF T2	READ T2 LOW OR WRITE T2 HIGH
TIMER 1	TIME OUT OF T1	READ T1 LOW OR WRITE T1 HIGH
IRQ	ANY ENABLED INTERRUPT	CLEAR ALL INTERRUPTS

Reg 14 ($0E) — Interrupt Enable Register (IER)



NOTES:

1. IF BIT 7 IS A "0", THEN EACH "1" IN BITS 0-6 DISABLES THE CORRESPONDING INTERRUPT.
2. IF BIT 7 IS A "1", THEN EACH "1" IN BITS 0-6 ENABLES THE CORRESPONDING INTERRUPT.
3. IF A READ OF THIS REGISTER IS DONE, BIT 7 WILL BE "0" AND ALL OTHER BITS WILL REFLECT THEIR ENABLE/DISABLE STATE.

Programming Example

A simple GETKEY routine — an example of using the VIA's control registers.

```
INIT    LDA   #% 0111 1111
        STA   IER          disable all interrupts
        LDA   PCR
        AND   #% 1111 1110 enable CA1 — negative
                           active edge
        STA   PCR
        LDA   ACR
        ORA   #% 0000 0001 latch on CA1
        STA   ACR
        LDA   #0           set all port A lines
        STA   DDRA         for input
KWAIT   JSR   KGET         loop until key pressed
        BCS   KWAIT        null rejected
        .
        .
KGET    SEC
        LDA   IER
        BIT   # 0000 0010  CA1 set?
        BEQ   RETURN
        LDA   IRA          (register 2, with
                           handshaking)
        CLC
RETURN  RTS
```

Announcing ....A New, Authoritative Guide to

The Most Important Book Ever Published for the Apple.

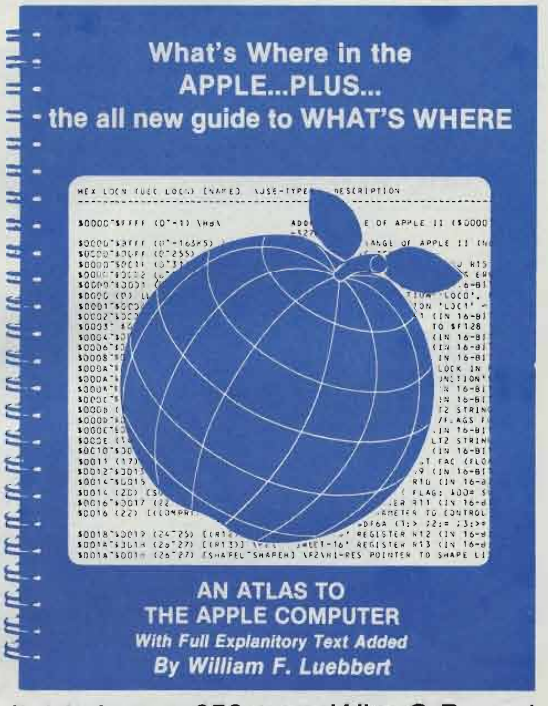
What's Where in the APPLE...PLUS...the all new guide to WHAT'S WHERE

is William F. Luebbert's Revised Edition of the famous Apple Atlas. The original *What's Where in the APPLE?* provided more information on the Apple's memory than was available anywhere else. Now the *Revised Edition* shows you how to use this valuable data.

What's Where in the APPLE...PLUS... the all new guide to WHAT'S WHERE

- 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.
- Explains how effectively to use the information contained in the original *What's Where in the Apple?*
- Enables you to move easily between BASIC and Machine Language.
- Guides you through the inner workings and hidden mechanisms of the Apple.

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



The Atlas and THE ALL NEW GUIDE are available in one 256 page Wire-O-Bound book for only **$24.95**

If you own the original What's Where in the Apple you will want **THE GUIDE** to complement your edition. This 128 page, Wire-O-Bound version contains all new material to be used with the memory map and atlas for **$9.95**

Ask for it at your computer store

Use the Coupon to Order Direct from MICRO
or
Call Toll Free Today 1-800-345-8112
(In PA 1-800-662-2444)

Please send me: M-8-82

_____ What's Where in the APPLE... Plus... the all new guide to WHAT'S WHERE @ $24.95

_____ THE GUIDE @ $ 9.95

Add $2.00 surface shipping for each copy, Massachusetts residents add 5% sales tax.

Total Enclosed $ _____

☐ Check ☐ VISA ☐ MasterCard Acct # _____

Expires _____

Name _____

Address _____

City State Zip

MICRO INK 34 Chelmsford St., P.O. Box 6502, Chelmsford, MA 01824

83-345

WE MAY HAVE ALREADY BUILT THE BOARD YOU ARE STARTING TO DESIGN!

While you may not have heard of us before, you certainly know our customers: Fortune 500 companies, Universities and Government Agencies. Since 1976 we have been providing high quality microcomputer products, ranging from expansion boards, to stand-alone controllers, to complete systems. Before you start your next project, consider how easy it might be to use some of our products.

FLEXI PLUS

A multi-function controller which handles:
- **8" and mini diskettes, double-sided, double-density**
- **RS-232 Communications with programmable features**
- **IEEE-488 Instrumentation Bus fully implemented**
- **6809E Microprocessor, up to 56K RAM, ROM, EPROM**
- **Parallel/Serial I/O, Cassettes and TTY Interfaces**

This versatile controller may be used as an expansion board for any 6502 or 6809 system; as a stand-alone controller; or, as the basis of a complete microcomputer system.

MICRO PLUS

A video-oriented controller which includes:
- **Programmable display up to 132 columns by 30 rows**
- **Programmable character sets in EPROM and RAM**
- **Character and Bit-Mapped Graphics**
- **ASCII Keyboard and Light Pen Interfaces**
- **RS-232 Programmable Communications Interface**
- **6502 Microprocessor, up to 7K RAM, 2K EPROM**

This video controller may be used to expand almost any 6502- or 6809-based system; or as a stand-alone intelligent terminal; or, as the basis for a complete 6502-based computer system.

DRAM PLUS

A multi-purpose expansion board which features:
- **Up to 40K RAM memory with a memory manager**
- **Up to 16K ROM or EPROM memory**
- **EPROM Programmer for 2516, 2716, 2532 and 2732**
- **Multiple parallel/serial I/O ports and timers/counters**
- **Prototyping area for custom circuits**

This memory-oriented expansion board permits addressing of memory on 4K boundaries, supports swapping of sections of memory, and works with most 6502- or 6809-based systems.

FOCUS

An Industrial quality system which features:
- **Two mini disk drives, double-sided, double-density for over 640K bytes of on-line storage**
- **Commercial quality keyboard with numeric pad**
- **Upper/lower case ASCII with programmable characters and display formats, plus bit-mapped graphics**
- **High-resolution video monitor with green phosphor**
- **Heavy-duty aluminum case for desk or rack mounting**
- **RS-232 Communications built in; second optional**
- **IEEE-488 may be added to existing boards**
- **Includes 48K RAM, 4K EPROM, 6809 microprocessor**

This extremely versatile system may be used for system development, for developing stand-alone products, as an end-user system, as an in-house business computer, as a word processor, and much more. Software is available from a number of suppliers and includes compiled BASIC, PASCAL, FORTH and many complete application packages.

$3495.00

If you have a requirement which involves 6502- or 6809-based products, join the growing number of OEMs and System Integration Houses who look to us first. For additional information and our current product literature, please contact us at 617/256-3649 or TELEX 955318 INTL DIV.

THE COMPUTERIST®

34 Chelmsford Street
Chelmsford, MA 01824

HDE Software? YOU BET!

PUT YOUR AIM • SYM • KIM OR OMNI 65 TO WORK WITH

- **DATA FOREMAN $89.95**

  General purpose data entry and retrieval system. Provides record keeping, sorting, searching, and user defined report writing.

  SOME COMMENTS FROM USERS -

  "Every time I use Data Foreman I find more applications for it."

  "Since purchasing both Data Foreman and your Inventory System, we've finally brought our inventory problem under control."

  "Your customer support is terrific, keep up the good work."

- **ALSO AVAILABLE -**

  INVENTORY MANAGEMENT
  950.00

  CHECKING ACCOUNT MANAGEMENT
  $129.95

AVAILABLE FROM:

KEYSTONE
DATA CONSULTANTS, INC.
CUSTOMER SERVICE
P.O. BOX 606
YORK, PA 17405

717-848-5666

MEMBER -
YORK AREA CHAMBER OF COMMERCE
NATIONAL FEDERATION INDEPENDENT BUSINESSES

Advertiser's Index

Aardvark Technical Services, Ltd. . . . 88, 122
ABM Products . . . 83
Advanced Logic Systems . . . 92
Andromeda, Inc. . . . 26
Anthro-Digital Software . . . 62
Apple Tree Electronics . . . 20
Ark Computing . . . 56, 65
Arrow Data Systems . . . 107
Aurora Software Associates . . . 113
Beagle Brothers . . . 47
Byte Microsystems . . . 62
Chandler Microsystems . . . 42
Comp-U-Gamer . . . 86
CompuTech . . . 65
Computer Case Co. . . . 66
Computerists Directory . . . 70, 98
The Computerist, Inc. . . . 126
Computer Mail Order . . . 37
Computer Science Engineering . . . 56
Computer Systems Associates . . . 40, 41
Connecticut Information Systems Co. . . . 105
Datamost . . . 38, 55
Data Transforms, Inc. . . . 30
Decision Systems . . . 65
D&N Micro Products, Inc. . . . 25
Eastern House Software . . . 19
ESD Labs Co., Ltd. . . . 9
Execom Corp. . . . 66
Genesis Information Systems Inc. . . . 60
Gimix, Inc. . . . 1
Hudson Digital Electronics Inc. . . . 48
Huntington Computing . . . BC
Interesting Software . . . 36
Keystone Data Consultants . . . 127
Leading Edge . . . IBC
MICRObits (Classifieds) . . . 115, 117
MICRO INK . . . 125
Micromed Instruments . . . 42
Micro Motion . . . 70
Microsoft Consumer Products . . . IFC
Micro Ware Distributing Inc. . . . 51
Microware Systems Corp . . . 32
Modular Systems . . . 56
MP Computer Services . . . 47
Olympic Sales Co. . . . 42
OPTMAL Technology Inc. . . . 98
Perry Peripherals . . . 87
Personal Computer Products . . . 4
Pretzelland Software . . . 83
Progressive Computing . . . 107
Quentin Research . . . 6
RAM/RBC . . . 95
RC Electronics . . . 59
Samauri Software . . . 15
Sensible Software . . . 69
SGC . . . 61
S&H Software . . . 68
Skyles Electric Works . . . 16
Small Systems Engineering . . . 21
Softech Microsystems . . . 111, 113, 115, 117, 119, 121
Softside Publishing . . . 84
Southeastern Microsystems . . . 52
Southwestern Data Systems . . . 10, 31, 98, 101
Stellation Two . . . 66
Sublogic Communications Corp . . . 2
Synergetic Solutions . . . 47
Versa Computing, Inc. . . . 22
Video Marketing . . . 96
John Wiley & Sons . . . 70
Zenith Data . . . 73

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

September 68000 Feature

- **The 68000's Instruction Set —** The first installment in a series offering detailed descriptions of the 68000's instructions. Part I contains a brief introduction to this chip.
- **An MC68000 Overview —** A discussion by Motorola of the 68000's registers and addressing modes.
- **The 68000 and the Personal Computer —** A look at how owners of 6502-based machines can benefit from the 68000.

Other Highlights

BASIC AIDS

Compress for the Apple
Auto Save for the PET
Amper POS — An Applesoft Position Function
Formatting Output on the Atari

HARDWARE

Delayed Reset on the OSI
Atari AID Conversion
Superimposing TV Pix on the PET

Plus our on-going columns and departments.

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

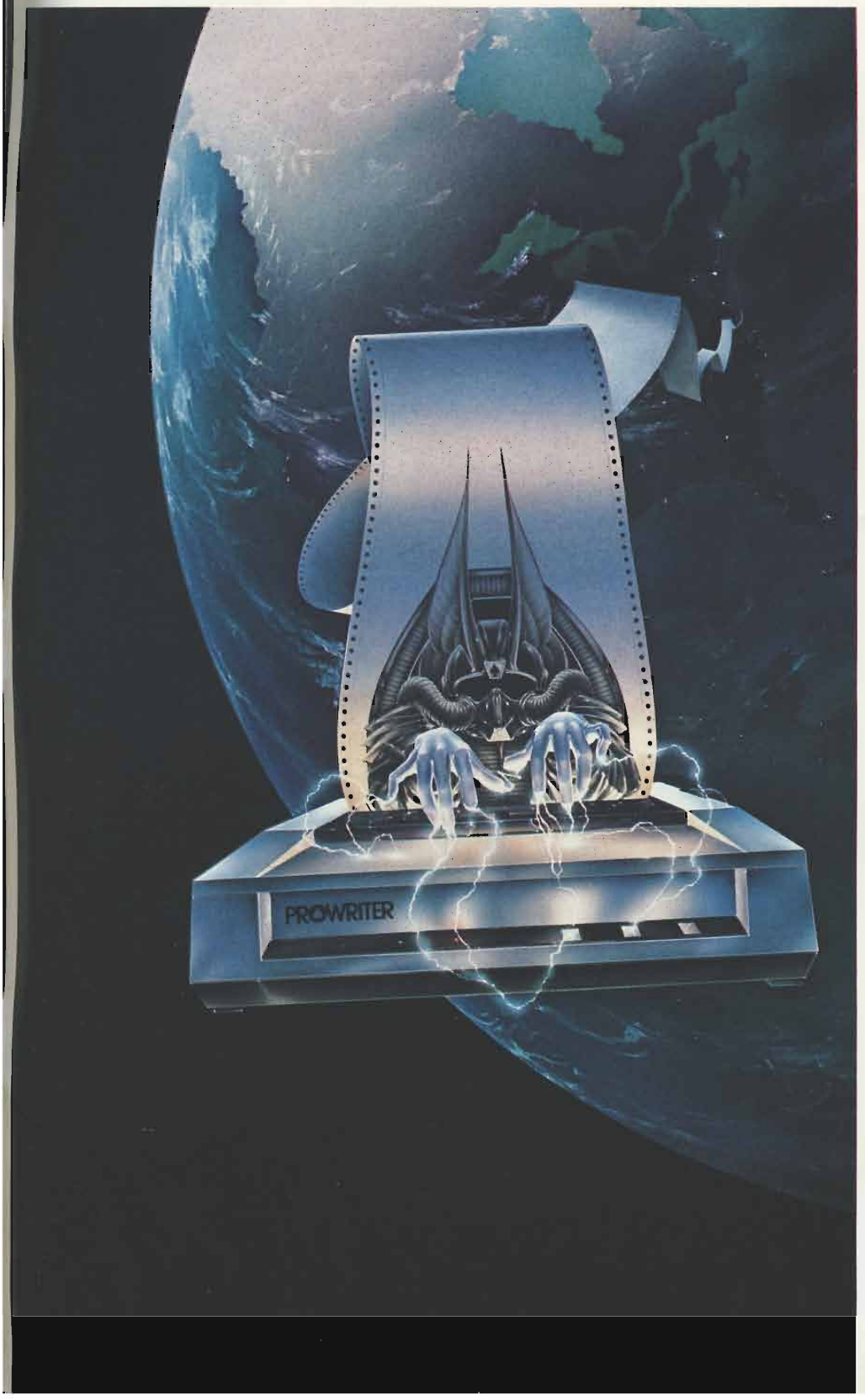
* Airmail subscriptions accepted for only 1 year.
For U.S. and Canadian 2-year rates, multiply by 2.

Job Title: _____

Type of Business/Industry: _____

THE PROWRITER COMETH.

(And It Cometh On Like Gangbusters.)



Evolution.

It's inevitable. An eternal verity.

Just when you think you've got it knocked, and you're resting on your laurels, somebody comes along and makes a dinosaur out of you.

Witness what happened to the Centronics printer when the Epson MX-80 came along in 1981.

And now, witness what's happening to the MX-80 as the ProWriter cometh to be the foremost printer of the decade.

SPEED
MX-80: 80 cps, for 46 full lines per minute throughput.
PROWRITER: 120 cps, for 63 full lines per minute throughput.

GRAPHICS
MX-80: Block graphics standard, fine for things like bar graphs.
PROWRITER: High-resolution graphics features, fine for bar graphs, smooth curves, thin lines, intricate details, etc.

PRINTING
MX-80: Dot matrix business quality.
PROWRITER: Dot matrix correspondence quality, with incremental printing capability standard.

FEED
MX-80: Tractor feed standard; optional friction-feed kit for about $75 extra.
PROWRITER: Both tractor and friction feed standard.

INTERFACE
MX-80: Parallel interface standard; optional serial interface for about $75 extra.
PROWRITER: Parallel and serial interface standard.

PRICE
Heh, heh.

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

NEW ITEMS

#8800	Adam & Eve Paddles (Tech Designs)	$33.89
#3259	Apple Mechanic (Beagle Bros.)	$24.99
#8660	Assembler Teacher (Compu Works)	$38.19
#1077	Bandits (Sirius)	$29.69
#3109	Business Bookkeeping System (Dakin 5)	$333.69
#1118	Cannonball Blitz (On-Line)	$29.69
#1457	Firebug (Muse)	$21.19
#2563	First Class Mail (Continental)	$63.69
#745	Firmware Rom (Novation)	$24.59
#8207	Format Rom (Soft Control)	$42.39
#1456	Frazzle (Muse)	$21.19
#1207	Global Program Line Editor (Synergis)	$33.19
#4403	Gold Rush (Sentient)	$29.69
#8002	Graftrax Plus (Epson)	$60.00
#1409	Graphics Combo Pkg. (Sublogic)	$101.79
#1078	Jellyfish (Sirius)	$25.39
#1079	Kabul Spy (Sirius)	$29.69
#1148	Lafpak (On-Line)	$29.69
#703	The Manager (Omega)	$29.69
#1111	Mouskattack (On-Line)	$35.09
#3302	P F S: Graph (Software)	$106.19
#7558	Personal Finance Master (Spectrum)	$63.69
#1018	Pursuit of Graf Spee (Strategic Sim.)	$50.89
#1164	Pie Writer - Multi (Hayden)	$127.39
#1163	Pie Writer - Standard (Hayden)	$127.39
#9487	The Programmer (Advanced Operating)	$420.69
#224	Pseudo Disk - Pascal (Saturn Sys.)	$33.09
#2183	Queen of Phobos (Phoenix)	$29.69
#221	64K Ramboard (Saturn Systems)	$361.00
#1681	S.A.G.A. #1-Adventureland (Adventure International)	$25.39
#9581	Slide Show (C & H Video)	$42.39
#8208	Sort Rom (Soft Control)	$33.89
#8700	Space Tablet 3-Axis (MCS)	$335.69
#8701	Space Tablet 4-Axis (MCS)	$446.19
#8640	Speed Read Plus (Optimized)	$50.89
#4657	Swordthrust #6-Eternal Curse (CE Soft)	$25.39
#4658	Swordthrust #7-Hall of Alchemie (CE)	$25.39
#2265	Visifactory (Micro Lab)	$63.69

SPECIAL

With any order over $49.00 (COD's excluded) receive FREE a copy of the program Mail Ordering (on disk for Apple). This program will allow you to make up mail orders with just a few keystrokes. Exclusively available from Huntington Computing. Ask for stock number 998.

ADVENTURE INTERNATIONAL

#1664	Adventures # 1-2-3	$34.99
#1665	Adventures # 4-5-6	$34.99
#1666	Adventures # 7-8-9	$34.99
#1677	Adventures # 10-11-12	$33.89
#1668	Apple Spice	$26.89
#1671	Poker Tournament	$17.79
#1681	S.A.G.A. #1 Adventureland	$25.39
#1682	S.A.G.A. #2 Pirate Adventure	$25.39
#1678	Stone of Sisyphus	$25.39

AVALON HILL

#2050	B-1 Nuclear Bomber (Cassette)	$12.99
#2057	Computer Acquire (Cassette)	$16.99
#2063	Computer Stocks & Bonds	$13.59
#2056	Conflict 2500 (Cassette)	$12.69
#2060	Empire of the Overmind (Cassette)	$25.49
#2059	Empire of the Overmind	$24.99
#2055	Lords of Karma (Cassette)	$16.99
#2058	Major League Baseball	$25.49
#2051	Midway Campaign (Cassette)	$12.69
#2052	North Atlantic Convoy (Cassette)	$12.69
#2053	Nuke War (Cassette)	$12.69
#2054	Planet Miners (Cassette)	$12.69
#2062	Tanktics (Cassette)	$20.39
#2061	Tanktics	$34.59

AVANT-GARDE

#7003	Chambers of Xenobia	$13.49
#7000	Creativity Tool Box	$38.19
#7005	Five Great Games	$25.39
#7004	Five More Great Games	$25.39
#7008	Race for Midnight	$25.39
#7006	Sentence Diagramming	$21.19
#7010	Sounds & Scrolling	$16.89
#7011	Super Draw & Write	$15.79
#7009	Super Shape Draw	$16.89
#7015	Word Scrambler	$16.89
#7016	Zero Gravity Pinball	$25.39

Softlights

By Fred Huntington

It's an exciting time around the Huntington household this month. We're very proud to announce the birth of our seven-pound five-ounce baby boy, Dale, born on June 6, 1982 in Visalia, California. Baby and Mama are doing just great and Melody (our three-year-old) loves him and calls him "My baby."

The other big news is that I have resigned my position as school principal so that I may devote full time to Huntington Computing.

We're instituting lots of changes to improve efficiency and speed. Our goal is to get everything out of the door within twenty-four hours with no backorders.

To celebrate the excitement, we're offering the following specials: (Always an excuse to run a sale.)

#9010 **Wurst of Huntington**
List $19.99, **now $9.99**

#4403 **Gold Rush (Sentient)**
List $34.95, **now $25.69**

#1967 **Arcade Machine (Broderbund)**
List $44.95, **now $34.49**

#1970 **Chop Lifter (Broderbund)**
List $34.95, **now $25.69**

#4002 **Verbatim Datalife Disks**
(w/plastic box & hub rings)
List $45.00, **now $25.99**

#9140 **Great Grandma Huntington T-Shirt - $5.00**

GREAT GRANDMA SAYS

Great Grandma Huntington once told me about a computer the Russians invented that was so smart it defected to the West!

Great Grandma also said, "Please buy little Freddy's Wurst of Huntington, because he personally gets two dollars for every one he sells. And, it's good. Just read the review in October 1981 issue of Softalk.

Watch next month's Softlights for the winners of the Great Grandma Huntington contest. There were some fantastic entries.

BRODERBUND

#1960	Alien Typhoon	$21.19
#1963	Apple Panic	$25.39
#1967	Arcade Machine	$39.49
#1966	David's Midnight Magic	$30.69
#1952	Galactic Empire	$22.99
#1954	Galactic Revolution	$22.99
#1953	Galactic Trader	$22.99
#1951	Galaxy Wars	$22.99
#2000	Genetic Drift	$25.39
#1957	Golden Mountain	$16.89
#1950	Hyper Head-on	$21.99
#1962	Labynnth	$25.39
#1961	Payroll	$335.69
#1968	Star Blazer	$27.89
#1956	Tank Command	$12.99
#1955	Tawala's Last Redoubt	$25.39
#1959	Track Attack	$26.89

MISCELLANEOUS

#8800	Adam & Eve Paddles (Tech Designs)	$33.89
#7890	Apple-cillin II (XPS)	$33.89
#3000	Bookkeeper (Delta)	$76.39
#9700	Castles of Darkness (Logical)	$30.69
#3001	Checkwriter (Delta)	$33.99
#9640	Colorblind (Energy)	$30.69
#9014	Computer Almanac (Huntington Computing)	$24.99
#6380	Crossword Magic	$39.99
#4401	Cyborg (Sentient)	$28.99
#9880	Deadline (Infocom)	$42.39
#9742	Electric Semicolon	$97.69
#9900	Financial Facts (Hanson)	$22.99
#9600	The Game Show	$34.29
#6870	Handwriting Analysis (Micro Lipp)	$16.89
#9840	Jabbertalky (Mind Toys)	$26.29
#9580	The Menu II (C & H Video)	$33.89
#9380	The Menu Generator	$35.09
#4400	OO-Topos (Sentient)	$27.99
#6240	Paddle-Adapple (Southern California)	$26.89
#7650	Pornopoly (CCI)	$25.39
#7920	Property Management	$292.49
#3400	Raster Blaster (Budgeco)	$25.39
#9500	Recipe Handler (Soft Touch)	$33.89
#9841	Ricochet (Mind Toys)	$16.89
#6600	Rubik's Cube (Software Alternatives)	$16.89
#9620	Shadow Hawk One (Horizon)	$43.89
#3380	Space War I (Galaxy)	$33.89
#4252	Star Blaster (Piccadilly)	$25.39
#9180	Starship Commander (Voyager)	$35.89
#4251	Suicide (Piccadilly)	$26.29
#9680	Taxman (Hal Labs)	$19.49
#4850	Time Lord (Ramware)	$25.39
#9012	Understand Yourself (Huntington)	$24.99
#9740	Volcanoes (Earthware)	$43.49
#9741	Volcanoes - Educational Version (Earthware)	$46.69
#9880	Zork I (Infocom)	$33.89
#9881	Zork II (Infocom)	$33.89

ON-LINE

#1120	Cranston Manor	$29.69
#1104	Crossfire	$26.29
#1126	The Dictionary	$87.89
#1125	M M S II	$43.89
#1115	Marauder	$29.69
#1101	Mission Asteroid	$17.99
#1113	Missile Defense	$21.19
#1102	Mystery House	$21.19
#1149	Pegasus II	$25.39
#1117	Sabotage	$21.19
#1105	Screenwriter II	$99.99
#1121	Soft Porn Adventure	$25.39
#1122	Threshold	$33.89
#1114	Ultima II	$46.69
#1112	Ulysses	$29.69
#1100	The Wizard and the Princess	$28.89

SIRIUS

#1058	Autobahn	$25.39
#1077	Bandits	$29.69
#1073	Borg	$26.89
#1065	Copts & Robbers	$29.69
#1053	Cyclod	$25.39
#1062	Epoch	$25.39
#1070	Joyport	$67.39
#1052	Minotaur	$29.69
#1066	Outpost	$26.29
#1057	Pulsar II	$25.39
#1071	Snake Byte	$25.39
#1064	Sneakers	$25.39
#1056	Space Eggs	$19.99
#1072	Twerps	$26.89

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.