Advice To PET Owners: How To Be A VIC Expert

The 6502 Resource Magazine
PET • Apple • Atari • OSI • KIM • SYM • AIM

Unleash The Power of Your Atari CPU

COMPUTE!

$2.50
April, 1981
Issue 11
Vol. 3, No. 4
63379

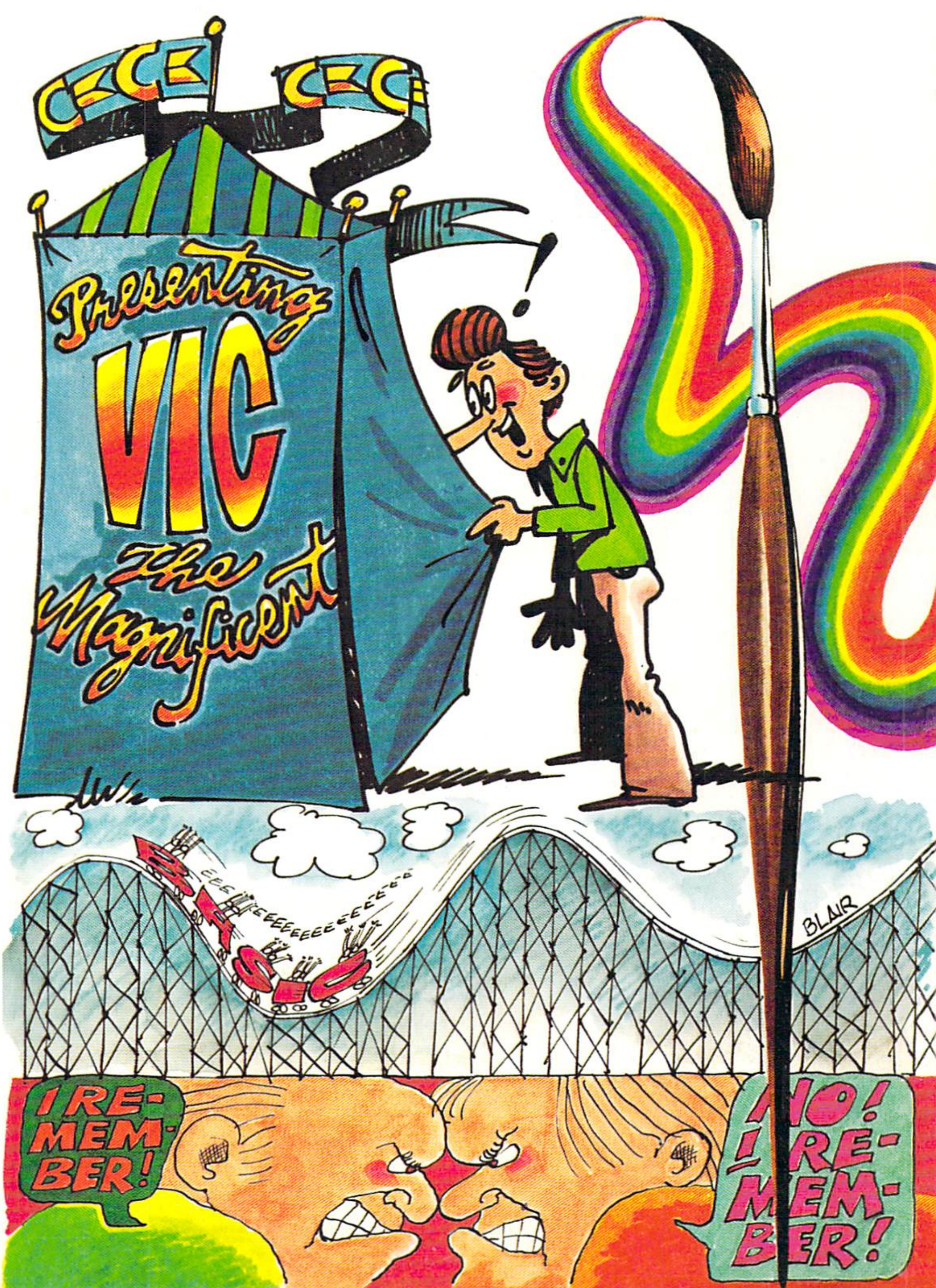
The Journal For Progressive Computing™

Product Preview: Commodore's $299 VIC-20 Computer

Super Cube: Artistry On Your Atari

Basically Useful BASIC: An Ascending-Descending Sort

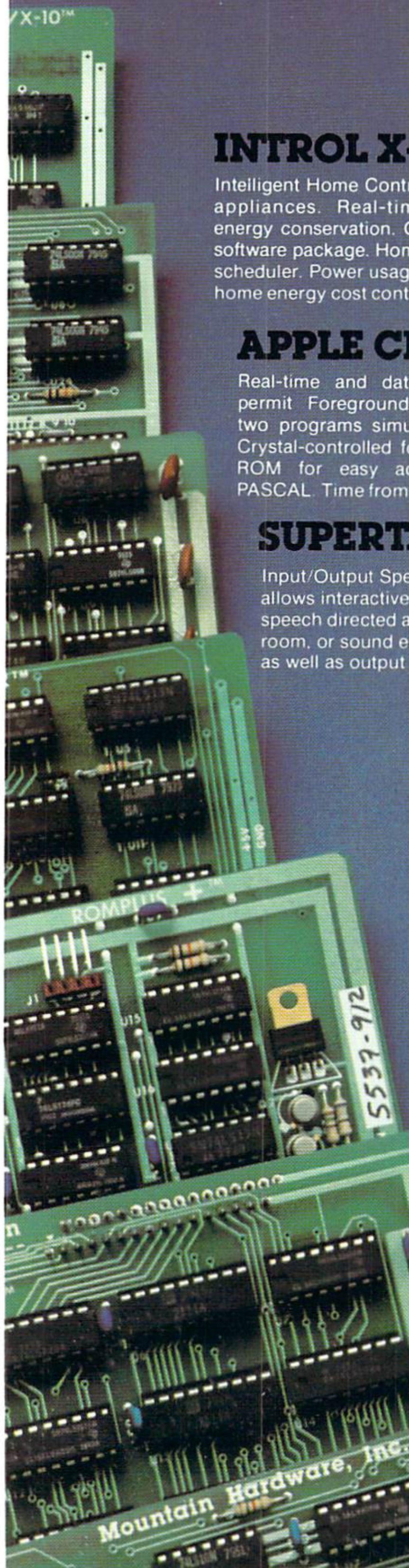
Resolving Applesoft And Hires Graphics Memory Conflicts



04
74470 63379

Mountain Computer makes more peripherals for the Apple Computer than Anybody.

and . . . a place to put them



INTROL X-10

Intelligent Home Controller for lights and appliances. Real-time schedules and energy conservation. Complete applications software package. Home security with random scheduler. Power usage accounting package for home energy cost control. No wiring required.

APPLE CLOCK

Real-time and date information. Interrupts permit Foreground/Background operation of two programs simultaneously. Battery back-up. Crystal-controlled for ± .001% accuracy. Onboard ROM for easy access from BASICs. Supports PASCAL. Time from one millisecond to one year.

SUPERTALKER SD200

Input/Output Speech Digitizer. Permits talking programs. I/O capability allows interactive programs with speech-prompted inputs. Use output for speech directed activities in business systems, announcements in a control-room, or sound effects in entertainment programs. Easy to use because input as well as output is under user control with special software operating system.

ROMWRITER

Program your own EPROMs. Create your own firmware. Programs 2K, 2716 5V EPROMs. Disk software package provides easy EPROM programming. EPROMs are verified after BURN. RUN your programs from on-board socket or install them on ROMPLUS+.

ROMPLUS+

More power for your system through firmware. Six sockets accept 2716 EPROMs or ROM equivalents. Six or any combination can be used at once. Scratch-pad RAM and two TTL connectors. Special 2K ROMs available for powerful system enhancement: Keyboard Filter ROM—COPYROM—Others coming soon.

MusicSystem

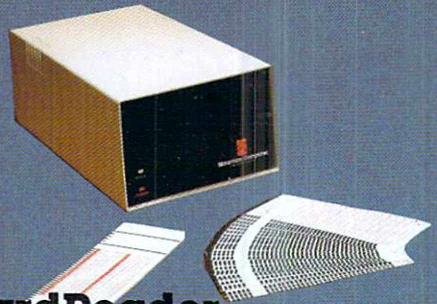
Sophistication previously available only on experimental mini and mainframe computer synthesizers. Digital instrumental music synthesizer system. 16 voices in stereo. Instrument definitions simulate the sound of real instruments—and more. Fully programmable waveforms. Envelope Control. Composition system—sheet music input using standard music notation. Chords and multi-part scoring up to 16 voices. A true instrument that anyone with an Apple can play.

A/D+D/A

16 channels analog to digital input. 16 channels digital to analog output. Eight bit resolution. Super-fast 9μ sec. conversion time. Monitor and output to the real world. All on one card.

EXPANSION CHASSIS

By popular demand! Eight more slots for your Apple. Attractive sturdy enclosure. Its own heavy duty power supply. Easy to use. Address cards in Expansion Chassis the same way as in your Apple. Only one additional command to specify in Apple or in Expansion Chassis. Compatible with all Apple peripherals.



CardReader

At last! An intelligent, high-quality device for data entry from user-marked cards. Implement BASIC programming, examination scoring, inventory maintenance and other applications requiring off-line data preparation for batch entry later. Connects to any computer having RS-232 interface. Software and cards are available for jobs in business, science and education.

MOUNTAIN COMPUTER has the most comprehensive line of Apple peripherals available. Anywere. From anybody. We know the Apple inside and out and are committed to providing the most innovative and unique products to expand and enhance its capabilities and use. After all, we were the first company to make an Apple peripheral—except Apple Computer.

Available at Apple Dealers worldwide.

Mountain Computer
INCORPORATED
300 Harvey West Blvd., Santa Cruz, CA 95060
(408) 429-8600 TWX 910 598-4504

*Based on Manufacturer's published catalogs—Apr. 1980

Apple is a trademark of Apple Computer Inc.

IF YOU'RE WAITING FOR THE PRICE OF WORD PROCESSORS TO FALL WITHIN REASON,

Everyone expected it would happen sooner or later...with WordPro PLUS™ it already has! Now all the marvelous benefits of expensive and advanced word processing systems are available on Commodore computers, America's largest selling computer line. WordPro PLUS, when combined with the new 80 column CBM 8032, creates a word processing system comparable to virtually any other top quality word processor available—but at savings of thousands of dollars!

New, low cost computer technology is now available at a fraction of what you would expect to pay. This technology allowed Commodore to introduce the new and revolutionary CBM 8032 Computer.

WordPro PLUS turns this new CBM 8032 Computer into a sophisticated, time saving word processing tool. With WordPro PLUS, documents are displayed on the computer's screen. Editing and last minute revisions are simple and easy. No more lengthy re-typing sessions. Letters and documents are easily re-called from memory storage for editing or printing with final drafts printed perfectly at over five hundred words per minute!

Our nationwide team of professional dealers will show you how your office will benefit by using WordPro PLUS. At a price far less than you realize.

Invest in your office's future...
Invest in WordPro PLUS...
Call us today for the name of the WordPro PLUS dealer nearest you.

Professional Software Inc.
166 Crescent Road
Needham, MA 02194
(617) 444-5224
TELEX: 95 1579

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

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

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

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

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

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

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

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

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

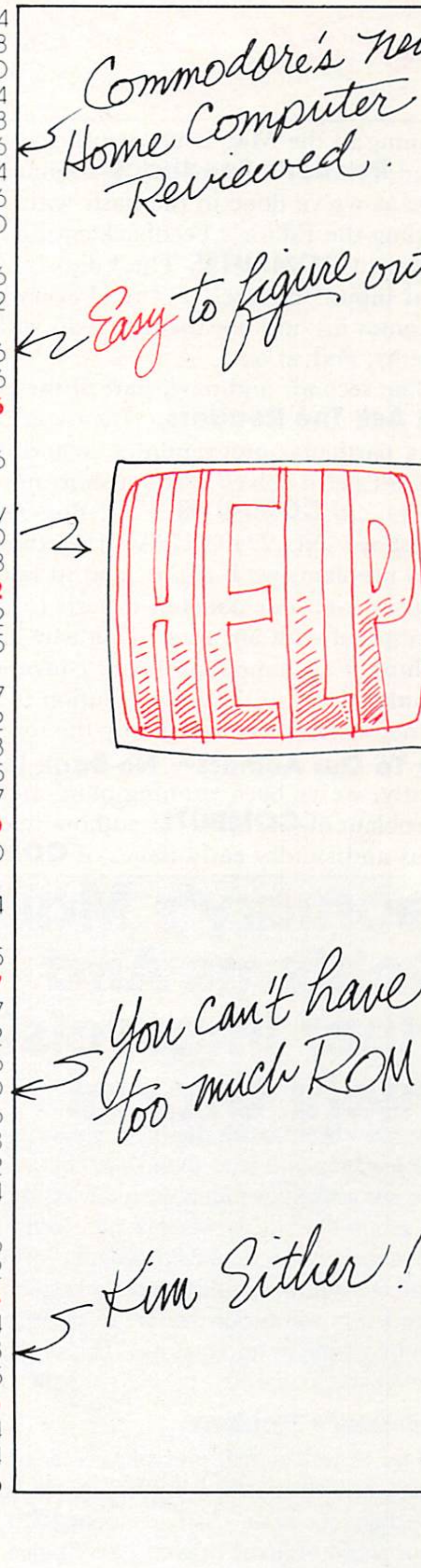
MICROSOFT
CONSUMER PRODUCTS

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

CIRCLE READER SERVICE NUMBER

Table of Contents

April, 1981, Vol. 3, No. 4



COMPUTE! The Journal for Progressive Computing (USPS: 537250) is published 12 times each year by Small System Services, Inc., P.O. Box 5406, Greensboro, NC 27403 USA. Phone: (919) 275-9809. Editorial Offices are located at 200 East Bessemer Ave., Greensboro, NC 27401.

Domestic Subscriptions: 12 issues, $20.00. Send subscription orders or change of address (P.O. Form 3579) to Circulation Dept., **COMPUTE!** Magazine, P.O. Box 5406, Greensboro, NC 27403. Controlled circulation postage paid at Greensboro, NC 27403. Application to mail at controlled circulation rates pending at Hickory, NC 28601.

 ISSN 0194-357X.

A Small System Services, Inc. Publication

COMPUTE! The Journal for Progressive Computing

The Editor's notes...

Robert Lock, Editor/Publisher

Beginning in the May issue, you'll see a new and expanded **Reader's Feedback** column. One part will be as we've done in the past, with reader input regarding the Editor's Feedback card. (For those of you new to **COMPUTE!,** The Editor's Feedback card is your input "hotline"...I read every single one that comes in, and use them to help with planning, problems, and so on.)

The second, and new, part of the column is called **Ask The Readers.** When you're trying to solve a particular programming or technical problem, and can't get it solved, drop a short note to Ask The Readers, c/o **COMPUTE!**, P.O. Box 5406, Greensboro, NC 27403 USA. If we think it's a shared problem, we'll run it, and in later issues run responses from our panel of experts (other readers who respond with answers). Don't be intimidated if you think it's a simple problem; conversely, don't be untimidated if you think the solution is simple. For beginner's, those are frequently the toughest kind.

Note To Our Authors— No Back Issues!

Recently, we've been running more and more into the problem of **COMPUTE!** authors referring back to various and sundry early issues of **COMPUTE!** One problem this causes is that many of our readers don't have access to back issues. Please remember this in your articles. It's fine to refer back to an earlier issue, but please take the additional time to incorporate the information from that article that's necessary to your own point. Here's our checklist of available back issues:

Issue 1, 2, 3, 4, 5, 6, 7 SOLD OUT
Issue 8, 9 and 10 Still available

If you're interested in ordering back issues, we've an ad in here someplace called **COMPUTE's Book Corner.** You'll also be pleased to know that we're putting together our first three books:

COMPUTE!'s First Book of Atari
COMPUTE!'s First Book of PET/CBM
COMPUTE!'s First Book of Educational Computing

We'll tell you more about these next time, but we expect to be releasing them in the June/July time frame. We'll send your local dealer information as soon as it's available. ©

February's Mailing Problems And Other Imprecisions. Ouch! Groan!

We have been aware that your subscription copies are reaching you later than store copies reach dealers. We are actively working on resolving the disparity, so no group of readers, whether newsstand or subscriber, is discriminated against. In this effort to promote timeliness and maintain quality, you subscribers will notice that we've returned to mailing the magazine in an envelope. This is done entirely to protect the magazine.

February's Problem

As far as we can tell, everyone received their magazine, albeit late. Murphey struck hard, but hopefully not again. As part of our effort to improve your speed of home delivery, we changed our mailing services to be geographically closer to our printer. Unfortunately, the local post office had not dealt with a magazine with a volume such as ours before and told our mailing personnel that the magazines needed to be bundled (e.g. by zone) only, and did not have to be bagged. Also unfortunately, the mailing service personnel believed them. The result was that the magazines went out promptly. They did not reach you promptly because, as far as we can establish, they sat in a regional bulk mail center for a week or two, waiting to be bagged. In effect, although they were in the hands of the Post Office, they were trickling out to their destinations. We apologize for the delay and concern it caused many of you. We are gradually speeding up the subscription delivery, and expect to reach par with newsstand/dealer delivery over the next few issues.

Subscription Price Increases

This probably isn't the optimal place to mention it, but I thought I'd take a moment to explain the new prices, and in particular the disparity between US and Canadian subscriptions. As of last issue, a twelve-issue subscription to **COMPUTE!** is $20.00 in the US, $25.00 in US funds in Canada, and for surface delivery elsewhere in the world. You're all aware of the rising costs of production, postage, etc., and the price increase, in part due to the tremendous growth in physical size of **COMPUTE!** is quite necessary.

As of January 1, our postage cost for sending the magazine to Canada increased by 93%. We found out about this increase when we went to the post office to mail some individual magazines on January 2. We are actively looking for alternative methods of reducing these costs. When we find them, we'll pass the savings along. ©

Subscription Information

(12 Issue Year):
COMPUTE! Circulation Dept.
P.O. Box 5406
Greensboro, NC 27403 USA
U.S. $20.00
Canada $25.00 (U.S. funds)
Europe: Surface Subscription, $25.00 (U.S. funds) if ordered direct, or available in local currency from the following distributors:

United Kingdom
Contact L. P. Enterprises,
8-11 Cambridge House
Cambridge Road
Barking, Essex
England IG1 18NT

Germany,
Switzerland,
Austria,
Contact Ing. W. Hofacker GMBH
8 Munchen 75
Postfach 437
West Germany

Canadian Retail Dealers should contact:
Micron Distributing
409 Queen Street West
Toronto, Ontario M5V 2A5
(416) 361-0609

Advertising Sales

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

Southern California,
Arizona, New Mexico
Jules E. Thompson, Inc.
2560 Via Tejon
Palos Verdes Estates, CA 90274
213 378-8361
Jo Ann Sullivan

Northern California,
Pacific Northwest, Rocky Mountain States,
Texas, Oklahoma
Jules E. Thompson, Inc.
1290 Howard Avenue, #303
Burlingame, CA 94010
415 348-8222
Phoebe Thompson

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

New England,
New York State
The Gittelman Company
Statler Office Building
Suite 582
20 Providence Street
Boston, MA 02110
617 451-0822
Joan Donahue

New York City Metro Area,
Mid-Atlantic and Southeastern States:
Local numbers:
New York 212 567-6717
Atlanta 404 523-1252
The Gittleman Company
Summit Office Centre
7266 Summit Avenue
Fort Washington, PA 19034
215 646-5700
Doug Johnson

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

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

Address all advertising materials to:
COMPUTE
200 East Bessemer Avenue
Greensboro, NC 27401 USA

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

Telephone: (919) 275-9809

An Interview With Dr. Chip

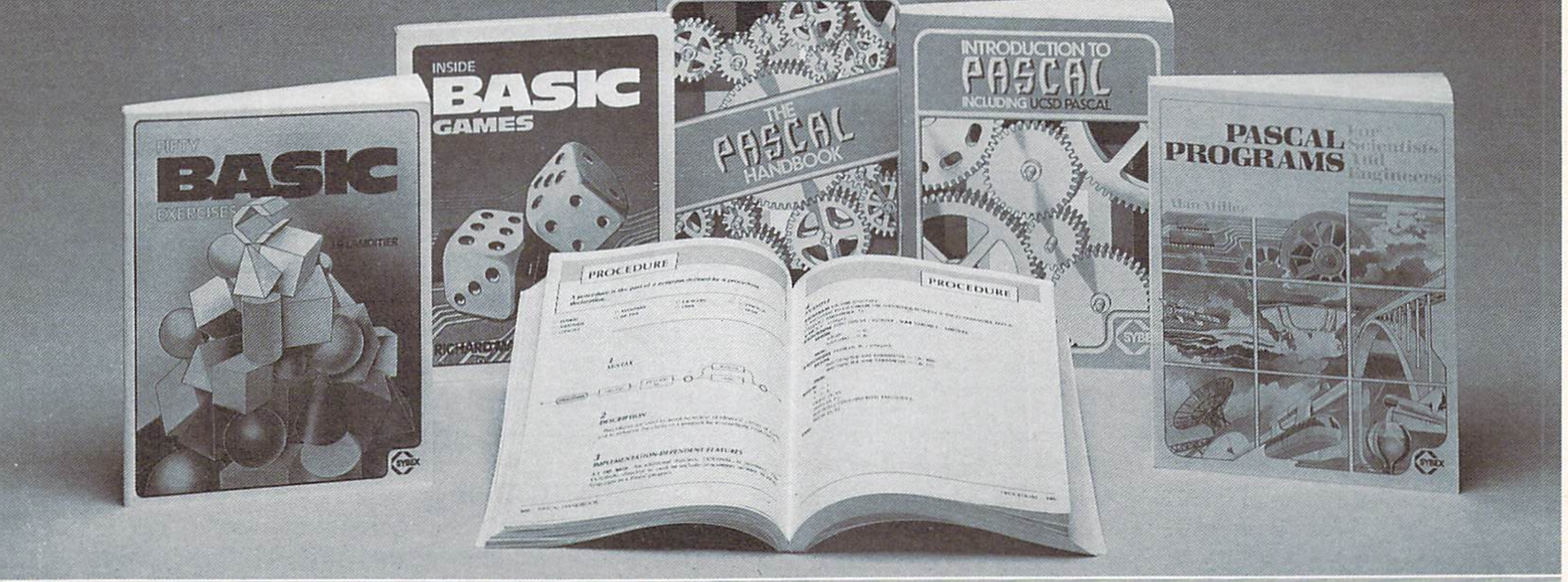
Robert Lock, Editor/Publisher

For those readers new to **COMPUTE!**, *Dr. Chip is Professor of 6502 Science at Figment U. He's also head of the Figment U. 6502 User's Group, a collection of 6502 users located at various Figment U. branches around the world. From time to time he consents to these interviews.*

RCL: Dr. Chip, it's good to see you again. I understand you've been quite busy.
Chip: I've been trying to piece together some late happenings in this industry, that is, when I can get any work done. That character who's trying to adopt me, the Silver Streak, has taken to calling me up on the phone in the middle of the night trying to sell me stock in his new T-shirt factory.
RCL: I'll have to admit, Chip, that The Silver Streak got some appreciative mail the first time he wrote you.
Chip: Harumph! Can you imagine me on a T-shirt...? Never mind. Back to business. First things first. Commodore has taken their dealer relations problem squarely on the chin. A recent business/financial article raked them over the coals. I'll tell you this; if they don't resolve some of their communications and customer relations problems, they'll be in a 6502 pickle. Finke (the new President of Commodore) has apparently taken direct responsibility for getting the Northeast distribution region ship-shape. That's one of Commodore's seven US regions, and I think he'll set up a model for the rest of the country.
RCL: I've received a good bit of mail lately from readers complaining about never receiving their Commodore US PET Users Group Newsletter. Commodore US told me last fall it was all being taken care of.
Chip: I suspect they'll get it together. Just as an example of how other Commodore operations treat their customers, I'll point out that Commodore Canada has been making refunds direct to their Canadian customers who sent orders to the US and never received anything. They seem to have a good handle on customer relations up there.
RCL: I can tell. We have extensive Canadian circulation, and I never get customer relations letters from Canada like those I get from the US readers.
Chip: Well, my money's on Finke.
RCL: I feel as though I've said this before, but we'll have to wait and see. By the way, have you found out what's happening to the KIM? (The KIM, for you uninitiated readers, is the ''single-board'' 6502 computer from Commodore that started the 6502 family several years ago).
Chip: It looks as though they have stopped producing it. We can't find out what's going on.
RCL: I tried to check on it and they said there were plenty in supply. Turns out, at least from the information that I get, that the plenty in supply aren't necessarily new units. Dealers we've heard from are completely out and can't get any. One industrial client called (he has an installed base of 175 KIMs running in an industrial environment), and all of a sudden he can't get any more. No warning, no comment, no answers. I'd like to hear from anybody caught up in the midst of this.
Chip: Sometimes I can't figure those guys out.
RCL: Any other news Chip?
Chip: Bits and pieces. There's a lot going on in the language area. Atari's new PILOT, (previewed by David Thornburg in last issue's ''Computers and Society'' Column), looks quite exciting, especially for beginners at any level. There's a new language coming along from Commodore called COMAL, ''Commodore Algorithmic Language''. Reports have it that it's a combination of the structured preciseness of PASCAL and the simplicity of Basic. It was developed by a Danish educator and is being used extensively in the Danish educational system. We hear it may be given away as public domain material.

University of Waterloo also has some interesting projects underway at the moment, but I can't provide details until later.
RCL: That sounds interesting. Waterloo is where they've done extensive research and development on the advanced languages for the larger machines, isn't it?
Chip: That's the place. I'll fill you in on their latest projects next time.
RCL: What's happening with VIC, Chip?
Chip: Commodore's new color computer seems to have been pushed back in US introduction time to a May-June time frame. We do know they're in the process of final redesign to meet the new FCC regulations. There are mixed reports on the Apple II and compliance with the new regs as well, but we can't yet tell what Apple's doing about them. The Atari units appear to already meet the new specs from the FCC, a point consistent with their methodical approach to this marketplace from the beginning. ©

SYBEX BREAKS THE LANGUAGE BARRIER



Let the chips fall where they may. These two books on BASIC assure comprehension and competence.

■INSIDE BASIC GAMES
by Richard Mateosian teaches interactive game design and BASIC programming through thorough analysis of eight different kinds of computer games. Programs are presented in Microsoft BASIC with versions for PET/CBM, TRS-80 and APPLE II.
350 pp., 120 illustr., Ref. B245, $13.95

■FIFTY BASIC EXERCISES
by J.P. Lamoitier provides the surest way of learning BASIC—actual practice. Graduated exercises, each containing a detailed explanation, flowchart and sample run, develop skill and competence rapidly. Applications include mathematics, business, operations research, statistics and more.
256 pp., 194 illustr., Ref. B250, $12.95

Get in gear and accelerate your programming productivity with Pascal's power.

■INTRODUCTION TO PASCAL
by Rodnay Zaks is a simple yet comprehensive guide to standard and UCSD Pascals: step-by-step presentation with exercises for beginners, complex concepts and extensive appendices for experienced programmers. An indispensable book for everyone who wants to learn Pascal programming.
320 pp., 100 illustr., Ref. P310, $14.95

■THE PASCAL HANDBOOK
by Jacques Tiberghien is an easy-to-read, easy-to-use dictionary containing all the features for most existing versions of Pascal (Standard, Jensen-Wirth, OMSI, UCSD, HP1000, Pascal/Z). Over 180 entries, arranged alphabetically; each includes definition, description, syntax diagram, details of implementation, variations and examples. The perfect reference tool for any Pascal user.
500 pp., 150 illustr., Ref. P320, $14.95

Scientists and engineers involved in significant work have been delayed by having to reinvent algorithms for a new computer language. No more.

AVAILABLE MAY 1981

■PASCAL PROGRAMS FOR SCIENTISTS AND ENGINEERS
by Alan Miller is a comprehensive collection of frequently used algorithms for scientific and technical applications programmed in PASCAL. This time saving book includes programs for curve fitting, fast Fourier transform, approximations, random number generation, integrals, statistical techniques and more.
250 pp., 80 illustr., Ref. P340 ☐$16.95 paper, ☐$25.00 cloth

MORE SYBEX BOOKS

■YOUR FIRST COMPUTER *by Rodnay Zaks*
280 pp., 150 illustr., 2nd Edition, Ref. C200A, $7.95

■MICROPROCESSORS: FROM CHIPS TO SYSTEMS *by Rodnay Zaks*
420 pp., 250 illustr., 3rd Edition, Ref. C201, $12.95

■MICROPROCESSOR INTERFACING TECHNIQUES *by Rodnay Zaks & Austin Lesea*
464 pp., 400 illustr., 3rd Edition, Ref. C207, $15.95

■PROGRAMMING THE 6502 *by Rodnay Zaks*
392 pp., 160 illustr., 3rd Edition, Ref. C202, $12.95

■6502 APPLICATIONS *by Rodnay Zaks*
288 pp., 207 illustr., Ref. D302, $12.95

■6502 GAMES *by Rodnay Zaks*
304 pp., 140 illustr., Ref. G402, $12.95

■PROGRAMMING THE Z80 *by Rodnay Zaks*
620 pp., 200 illustr., 2nd Edition, Ref. C280, $14.95

■PROGRAMMING THE Z8000
by Richard Mateosian
312 pp., 124 illlustr., Ref. C281, $15.95

■THE CP/M HANDBOOK (With MP/M)
by Rodnay Zaks
336 pp., 100 illustr., Ref. C300, $14.95

SYBEX

MAIL TO:
SYBEX DEPT. C 41
2344 SIXTH STREET
BERKELEY, CA 94710
PHONE ORDERS:
INSIDE CA 415/848-8233
TOLL FREE OUTSIDE CA 800-227-2346

or at book and computer stores everywhere

NAME ______ ☐ SEND ME YOUR FREE CATALOG
ADDRESS ______
CITY ______ STATE ______ ZIP ______
ADD ☐$1.50/book UPS or ☐75¢/book 4th class mail or ☐$8/book overseas airmail
(CA add tax) Total Amt. Enclosed ______ OR CHARGE MY ☐VISA ☐MC ☐AM EX.
CARD NO. ______ EXP. DATE ______
SIGNATURE ______

Computers and Society

David D. Thornburg
Innovision
P.O. Box 1317
Los Altos, CA 94022

My last column concentrated on the software interface between people and computers. As I said at that time, the mechanical devices through which we communicate with our computers are no less deserving of attention. I find it both sad and amusing to see that the principal method by which humans interact with computers is through a keyboard whose 100 year old design is based on the limitations of antiquated mechanical devices.

I know that I touched on this topic in a previous column, but a phone call I received a few weeks ago has rekindled this issue for me, and perhaps it is time to talk about keyboards again. It was my pleasure to receive a phone call from Dr. Mary Humphrey in Canada. She told me of her experiences in working with children who use computers extensively in an educational environment. As I recall, these children range from the educationally handicapped to the gifted, and cover a wide range of ages. After working with the computers for a while, the children are asked to describe what things they like and what things they dislike. She told me that the major problem that children encounter with the computer is the arrangement of keys on the keyboard. "The keys are all mixed up." Some children even propose alternative keyboard layouts — all of which are alphabetical.

I know that this is a tender topic, and one which has been with us for a long time, but I persist in thinking that there is a golden opportunity, right now, for us to improve this interface between people and computers.

This month we will explore the development of the commonly used Sholes keyboard, the evolution of alternatives, and the promise of a new keyboard environment for the many millions of new computer users who have no training in typing.

Did you know that the 100 year old layout of the keyboard used on most personal computers was intentionally designed to be hard to use? This arrangement was developed in 1872 by typewriter inventor C. Latham Sholes (1) and his attorney, James Densmore, to overcome a major problem in the design of Sholes' original typewriter. Originally the keys were arranged in alphabetical order. Unfortunately, this arrangement made it very easy for certain commonly used keys to be typed in such quick succession that adjacent type elements would jam together before hitting the ribbon. To overcome this problem, Sholes and Densmore placed the most commonly typed letters as far apart in the type basket as possible, and the result was the QWERTY keyboard we have today (see Figure 1). The name QWERTY is derived from the first five keys in the top alphabet row of this keyboard, and is a folksy name for the Sholes arrangement.

Most of the effort applied to improvements on the Sholes keyboard has been geared towards improving typing speed and reducing operator fatigue.

...the 100 year old layout of the keyboard used on most personal computers was intentionally designed to be hard to use...

In 1932, after many years of work, August Dvorak (2) suggested a keyboard arrangement similar to that shown in Figure 2. As with the Sholes keyboard, the DSK (for Dvorak Simplified Keyboard) requires a lot of training to use effectively. Its principal advantage for touch typists is that skilled DSK users can type at up to twice their previous typing speed with less fatigue.

If DSK is so much better than Sholes, it is logical to ask why the improved keyboard has not displaced the older inefficient model, especially since the original mechanical limitations leading to QWERTY have been overcome for many years. There seem to be two causes for this failure. The first is the inertia associated with displacing the many millions of Sholes keyboards in use today. Second, there is the understandable resistance on the part of typists trained on the Sholes arrangement, each of whom would have to spend about a month making the transition to a new and (presently) hard to obtain keyboard.

Neither of these problems has deterred other researchers however, and many other alternatives to QWERTY have been proposed in recent years. Among the more interesting concepts that have been studied is the "chord" keyboard. This idea, pioneered by E. T. Klemmer at IBM (3) in 1958, entails the use of a keyboard with only ten keys, one for each finger. Letters are typed by pressing the correct sequence of keys at the same time, much as one would play a chord on a keyboard instrument. With ten keys, 1023 different patterns can be generated. To make it easier on the user, Klemmer didn't ask the user to press more than two keys at once. Users of this system were able to type at more than 40 words per minute after intensive training. Klemmer felt that the real power of his keyboard would come from using additional finger chords for the entire words.

While Klemmer's keyboard was designed from the human's point of view (commonly used letters used single keystrokes and favored the stronger fingers), other keyboard designers have decided that people should change their behavior to match that of the machine. Several designers have introduced five-key keyboards on which the user is expected to type the alphabet in raw ASCII-like code. While having certain appeal to some high-tech acquaintances of mine, most of these efforts have, thankfully, died a quiet death.

My concern is that neither the Sholes, the DSK arrangement, nor any of the chorded keysets makes any sense to the novice user. As the personal computer market continues to expand, an increasing number of people are being asked to type on a keyboard which makes no sense whatsoever. Anyone who has watched a child use a computer has seen the intense concentration with which he or she scans the keyboard looking for the right key. These novice "hunt and peck" typists typically use the index finger of one hand to do their typing, with "advanced" novices using the index fingers of both hands. When one considers the myraid applications for the computers used by novice typists, it seems almost criminal that a powerful modern tool like the personal computer should be constrained to use a keyboard designed as an apology to the limitations of nineteenth century mechanical skill.

Since you, most likely, have had some exposure to the Sholes keyboard, you might think I am overstating my case. You should perform the following experiment: Look at the DSK keyboard shown in Figure 2 and type: The quick brown fox jumped over the lazy dog.

Now imagine how a child feels when presented with QWERTY for the first time!

I feel that a solution to this problem exists. It is logical to ask why I think the time is ripe for change when Dvorak had so much trouble forty years ago, and since none of the other systems has moved far from the research laboratory.

The answer to this question is that, for the first time since 1873, a major keyboard market has opened for which the purchasers and users of these keyboards are not already skilled typists. The personal computer market in the United States jumped from almost nothing in 1977 to 150,000 machines in 1979. The annual sales figure appears to be doubling every year, and sales this year might reach a rate of over one million computers per year. While there is no reliable figure on the saturation level of this market, conservative estimates of 50 million computers represent a probable lower bound for this marketplace. This massive market, coupled with the fact that the overwhelming majority of new personal computer users are not already "touch typists" is what gives encouragement to the concept of a new keyboard arrangement.

In thinking about new keyboard arrangements useful to novices, it is fairly obvious that the keys should be arranged in alphabetical order. Consider the environmental forces which lead to this conclusion. Children are taught their ABC's from the moment they can talk. One often hears children singing the alphabet song:

"Ay bee cee dee ee eff gee,
aitch eye jay kay,
ell em en oh pee..."

On the other hand, I have never heard a child sing:

"Kew doubleyou ee are tee wy,

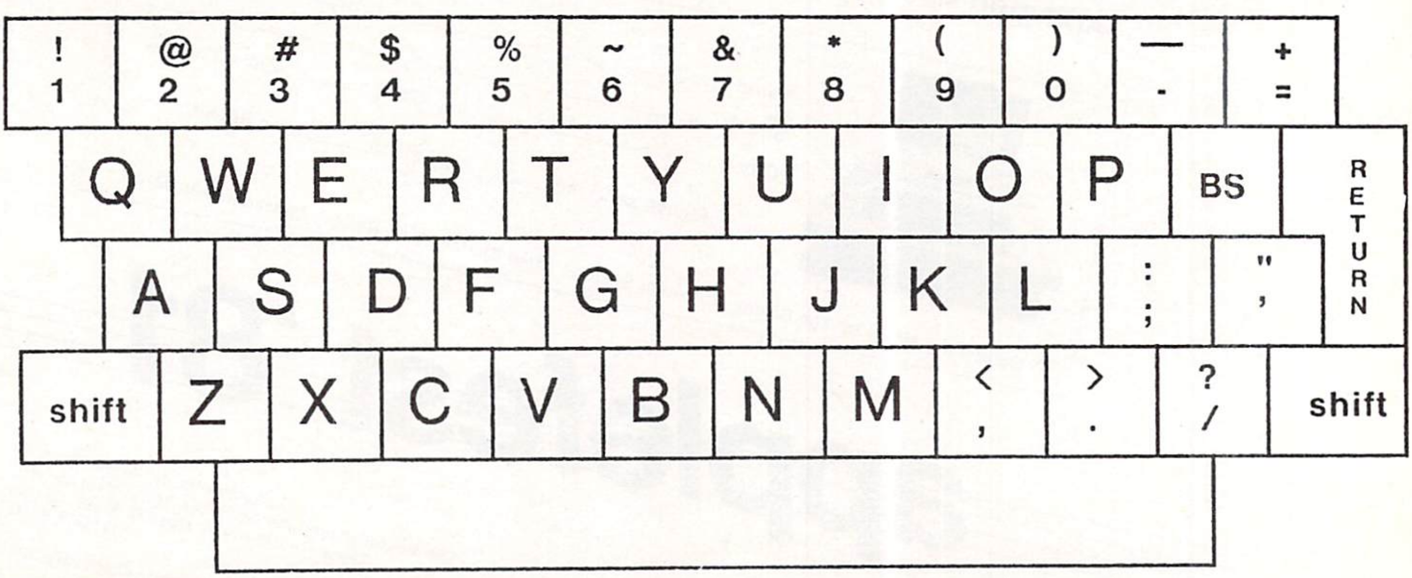


Figure 1:
Modern keyboard layout based on the Sholes arrangement

you eye oh pee,
ay ess dee.''

Later on, as reading skills are better developed, we teach our children how to use the dictionary. Once again, they encounter the letters in alphabetical order. The reason that words in dictionaries are arranged in alphabetical order is very simple. The user's pre-existent knowledge of the letter sequence allows rapid location of a chosen word among thousands of other words. Those of you who have learned a language which uses a different alphabet (such as Russian) probably experienced some frustration in learning to use a dictionary in that language until the ''new'' alphabetical order became entrenched in your mind.

If the QWERTY arrangement is so good, then one must ask why office workers who use it for typing don't also use it for filing documents.

As it turns out, some enlightened vendors of consumer products have realized the value of our early childhood education, and offer alphabetic keyboards on their products. The Texas Instruments' Speak & Spell, Mattel's Brain Baffler and the Craig ''pocket translator'' immediately come to mind. In one of the more ambitious projects of considerable relevance to computer using educators, Children's Television Workshop used alphabetical keyboard arrangements in the seventy-odd Apple computers located at Sesame Place in Bucks County Pennsylvania.

If these domestic projects aren't enough, consider the fact that the government of France will be performing a test of their electronic phone directory system by installing 250,000 computer terminals in people's homes this year. Within a few years three million of these terminals will be installed. A few years after that, perhaps 37 million such terminals will be in daily use - each of them with an alphabetic keyboard layout. The point is not just that alphabetical keyboard arrangements are possible, but that some companies are actually finding that these arrangements are commercially successful.

There are many merits to using an alphabetical keyboard arrangement. Unlike either of the keyboards shown in Figures 1 and 2, the user does not have to scan the whole keyboard to find a given key. This is important since the foveal regions of a user's eyes can only be focused on one or two keys at a time. Once a key is perceived (assuming it is the wrong key) the logic behind the alphabetic layout helps to reduce the time required to find the desired letter or symbol. The use of color coded keytops (with vowels having a different color than consonants, for example) may also be of benefit to some users.

The development costs associated with alphabetic keyboards are no different from those associated with the Sholes arrangement, so there is no particular reason for this new keyboard to be more expensive than the more traditional model.

Can a logical keyboard designed for novices find its way to the marketplace? The answer is yes if those millions of us who are buying computers make it happen. The tyranny of QWERTY can be stopped at last!

References:

1. C. L. Sholes, U. S. Patent 207,559 (1878).
2. A. Dvorak, et al., **''Typewriting Behavior: Psychology Applied to Teaching and Learning Typewriting''**, American Book Co., New York (1936).
3. E. T. Klemmer, **''A Ten-Key Typewriter''**, IBM Research Memo #RC-65 (1958).

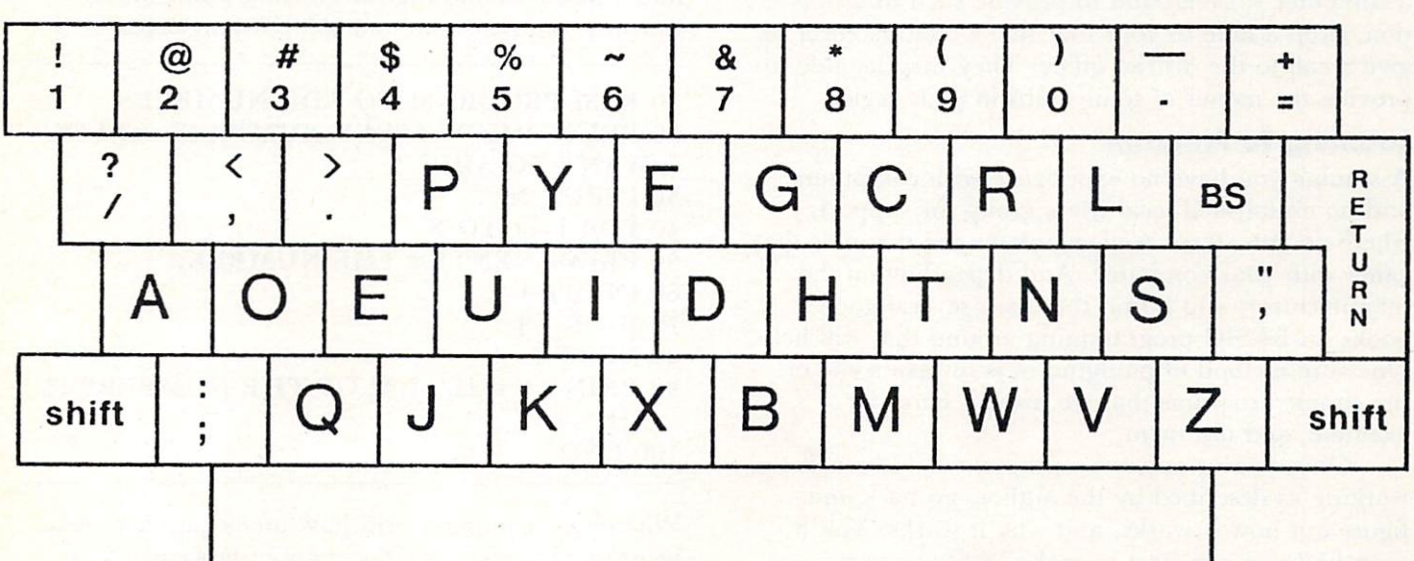


Figure 2:
Modern keyboard layout based on Dvorak Simplified arrangement (except for numerals)

The Beginner's Page

Robert Lock
Editor/Publisher

This page is a continuing, "re-cycling" feature in **COMPUTE!**. *It consists of a set of articles that repeat, in sequence, across issues. Thus, if you're a beginner to computing, you can pick up the series whenever you start with* **COMPUTE!**, *and within four or five issues, have the set. By then, you'll find you've advanced far beyond where you are now, especially if you have your hands on a computer.*

Part Two

Access to Resources

If you're just getting started, you'll find several important sources of information are available to you. Beyond the obvious channels, such as magazines and books, you'll quickly discover a community of users. Your local computer store can help there. They can frequently specialize in, or at least cover, your particular computer. Depending on the size of your user community, you may even find seminars for beginners, a lending library of back issues of magazines, and so on.

If you're in an area where activity hasn't yet grown to the point of established clubs, or there's not a computer store around to provide such information, drop a note to your machine's manufacturer or give a call to the district office. They may be able to provide the names of some clubs in your region.

Learning To Program

Assuming you have no experience with computers, and no established local users group for support, where do you start? Well, you have the manuals that came with your computer. And depending on the manufacturer, you'll find there are several good books on BASIC programming around that will help. One sure method of plunging in is to take some of the simple programs that we present here, for example, and use them.

Once you've entered a program, and have it working as described by the author, go back and figure out how it works, and why it works. You'll soon find you can start to make additions to programs from books or magazines that help "customize" them for your own use. This is an ideal way to learn.

My advice is to start at the very beginning, and use some feature of BASIC until you understand its usefulness and purpose. Continue to add on features as you need them or want to understand them. Above all, don't get frustrated. The best way to learn to program is to program.

Here's a sample of what I mean. Type this program into your computer (press return after each line):

```
10 REM PROGRAM #1
20 PRINT "HELLO"
30 END
```

NOW TYPE RUN, AND PRESS RETURN.
Your computer should print HELLO on the screen, followed by READY. Ta Da! A working program. Surely, you say, I bought this machine to do more than this. Of course you did. Let's turn our sample into a more useful program, adding a few more features common to all our BASIC languages.

```
10 REM PROGRAM TO ADD NUMBERS
20 PRINT "HOW MANY NUMBERS DO YOU
   WANT TO ADD?"
30 INPUT N
40 FOR I = 1 TO N
50 PRINT "ENTER THE NUMBER."
60 INPUT J
70 K = K + J
80 NEXT I
90 PRINT "THE SUM OF THE NUMBERS IS
   ";K'
100 END
```

When your computer asks how many numbers you want to add, type in some small number like **5.** It will then ask you, 5 times, to **"Enter the number."** Each time, type in one number that you want to add to the sum.

COLLEGE BOARDS

for TRS-80 PET, APPLE

The best way to sharpen your skills for the College Boards is to work on actual examinations. Each of these program sets confronts the user with a virtually limitless series of questions and answers Each program is based on past exams and presents material of the same level of difficulty and in the same form as used in the College Board examination. Scoring is provided in accordance with the formula used by College Boards.

SAT, PSAT, N.M.S.Q.T., set includes 7 programs covering Vocabulary, Word Relationships, and Mathematics. Price $79.95

EDUCATOR EDITION - SAT, PSAT Includes all of the above programs plus detailed solutions and explainations for each problem plus drill exercises. SAT set includes 14 programs. $149.95

GRADUATE RECORD EXAMINATION set includes 10 programs covering Vocabulary, Word Relationships, Mathematics, Logical Diagrams, Analytical Reasoning. $139.95

EDUCATOR EDITION - Graduate Record Exam Set includes 20 programs. $199.95

SWORD OF ZEDEK

Fight to overthrow Ra, The Master of Evil. In this incredible adventure game, you must confront a host of creatures, natural and supernatural. To liberate the Kingdom, alliances must be forged and treasures sought. Treachery, deceit and witchcraft must be faced in your struggles as you encounter wolves, dwarves, elves, dragons, bears, owls, orcs, giant bats, trolls, etc. Each of the 12 treasures will enhance your power, by making invisible, invulnable, more eloquent, more skillful in combat etc., etc., as you explore the realms of geography both on the surface and underground. Dungeons, temples, castles, mountains etc., are all a part of the fantastic world of Ra. Each game is unique in this spectacular and complex world of fantasy. $24.95

KRELL GAME PAK

Incredible bargain, 15 unique and challenging games!!!

Hostage	**Primary Fight**
Prime Time	**Black Gold**
The Black Death	**Wordsworth**
Star Clipper	**Hard Scrabble**
Bulls & Bears	**Bible Quode**
Banzai	**Shakespeare Quode**
Banzai II	**Bill of Rights Quode**
Super Banzai	*TRS-80 only, $39.95*

SUPER STAR BASEBALL

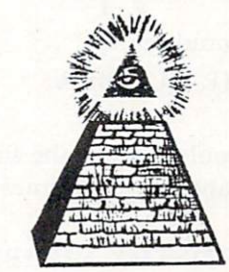
ALL TIME SUPER STAR BASEBALL Sample Lineup		SUPER STAR BASEBALL Sample Lineup	
B. Ruth	T. Williams	D. Parker	J. Rice
L. Gehrig	J. Foxx	W. Stargell	H. Aaron
J. DiMaggio	H. Greenberg	W. Mays	L. Brock
J. Jackson	R. Hornsby	P. Rose	R. Carew
G. Sisler	H. Wilson	O. Cepeda	H. Killebrew
S. Musial	B. Terry	C. Yazstremski	R. Allen
T. Cobb	M. Mantle	W. McCovey	R. Leflore
W. Mays	H. Aaron	R. Jackson	R. Zisk
C. Young-P	W. Johnson-p	G. Brett	B. Madlock
		R. Guidry-P	T. Seaver-p

Performance is based on the interaction of actual batting and pitching data. Game can be played by one or two players with the computer acting as a second player when desired. Players select rosters and lineups and exercise strategic choices including hit and run, base stealing, pinch hitting, intentional walk, etc. Highly realistic, there are two versions, ALL TIME SUPER STAR BASEBALL, and SUPER STAR BASE—BALL featuring players of the current decade. Each includes about 50 players allowing nearly an infinite number of roster and lineup possibilities.

$14.95

TIME TRAVELER

The best of the adventure games. Confronts player with complex decision situations and the demand for real time action. Using the Time Machine, players must face a challenging series of enviornments that include; the Athens of Pericles, Imperial Rome, Nebuchadnezzar's Babylon, Ikhnaton's Egypt, Jerusalem at the time of the crucifixion, The Crusades, Machiavelli's Italy, the French Revolution, the American Revolution, and the English Civil War. Deal with Hitler's Third Reich, Vikings, etc. At the start of each game players may choose a level of difficulty... the more difficult, the greater the time pressure. To succeed you must build alliances and struggle with the ruling powers. Each game is unique. $24.95



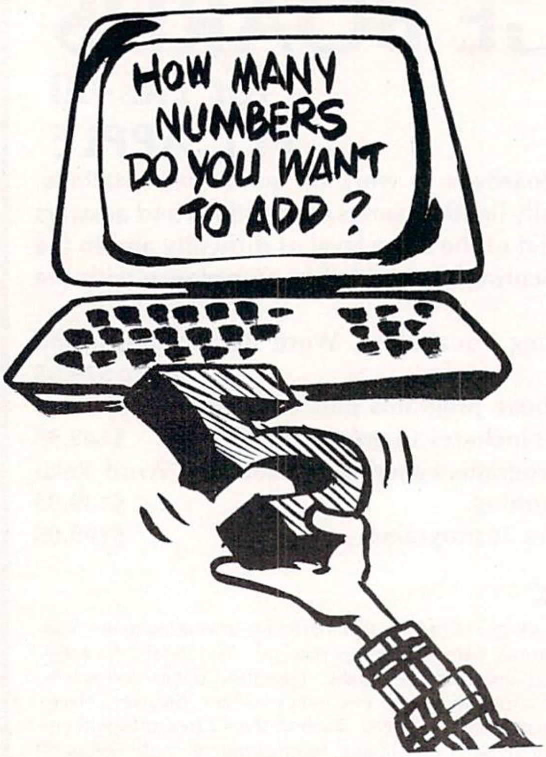
*ALL PROGRAMS AVAILABLE FOR TRS-80, APPLE II & PET

*Programs for APPLE or TRS-80 are on disk or cassett, please specify.

All programs require 16K ● TRS-80 programs require LEVEL II BASIC ● APPLE programs require Applesoft BASIC

KRELL SOFTWARE

Send check or money order to
21 Milbrook Drive, Stony Brook, NY 11790
(516) 751-5139

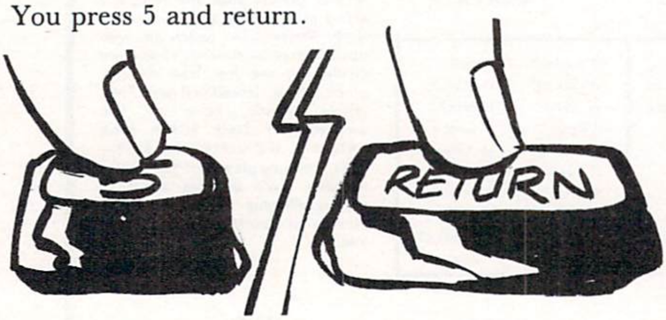


Here's what you should see on your screen after typing run.

HOW MANY NUMBERS DO YOU WANT TO ADD?

?■

You press 5 and return.



Now you should see:

ENTER THE NUMBER.

?■

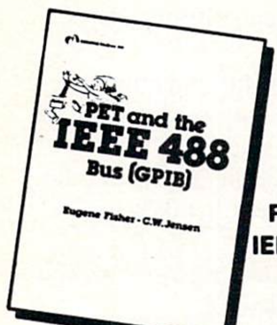
Here you should type in the first number of your group of 5, and so on (5 times) until the computer says:

THE SUM OF THE NUMBERS IS:_________

This will be the sum.

Try this one out, and next time we'll expand it further, explaining how it works, and some nice ways to make it work more usefully.

©



PET and the IEEE 488 Bus (GPIB)
by E. Fisher and C. W. Jensen

This is the only complete guide available on interfacing PET to GPIB. Learn how to program the PET interface to control power supplies, signal sources, signal analyzers and other instruments. It's full of practical information, as one of its authors assisted in the original design of the PET GPIB interface.

#31-4 $15.99

NEW PET/CBM edition

Some Common BASIC Programs
by L. Poole, M. Borchers, C. Donahue

76 Programs you can use even if you don't know BASIC. This book gives you a variety of math power including personal finance, taxes and statistics as well as other programs you'll want like Recipe Cost and Check Writer. All programs can be run on a PET or CBM with 8K or more.

#40-3 $14.99

PET owners can purchase the programs ready-to-run on cassette or disk. Use the book as a manual for operating instructions and programming options.

Disk #33-0 $22.50
Cassette #25-X $15.00

Practical BASIC Programs
ed. Lon Poole

These are 40 easy to use programs that each do something useful.

Income averaging, checkbook reconciliation, statistics, factorials, temperature conversion and musical transposition are just a few. It offers a wealth of practical computing power. Includes write-ups, program notes and instructional examples to help you realize the potential uses of each program.

#38-1 $15.99

6502 Assembly Language Programming
by L. Leventhal

Increase the capabilities and performance of PET (and other 6502-based computers) by learning to program in assembly language.

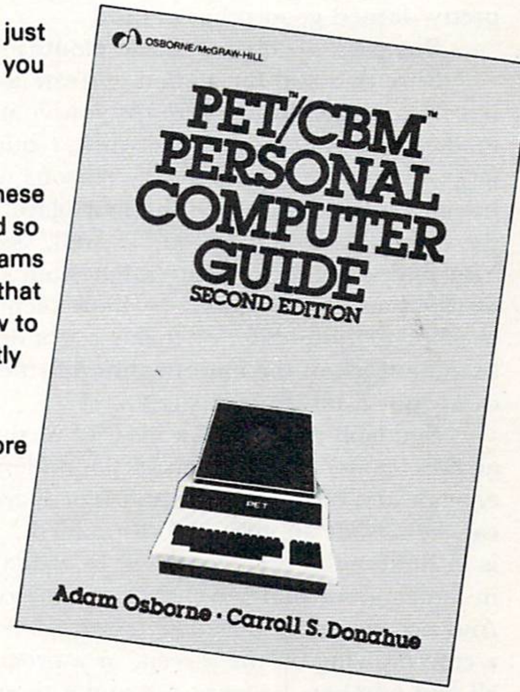
#27-6 $16.99

New for your PET
from OSBORNE/McGraw-Hill

PET™/CBM™ Personal Computer Guide Second Edition
by Adam Osborne and Carroll S. Donahue

The PET/CBM Personal Computer Guide is a step-by-step guide that assumes no prior knowledge of computers. If you can read English, you can use this book.

This book provides the important information and documentation that PET/CBM users have sought for so long. After reading this book you will have a good understanding of what a computer — especially the PET/CBM computer — can do for you. If you've just bought a PET or CBM this is the book you must have to really understand your computer. By using the examples found in this book, you will quickly get your PET/CBM up and running. These examples are thoroughly documented so you can learn how and why the programs work. It is the "how" and the "why" that are important if you want to learn how to make your PET or CBM work efficiently for you.



This second edition contains even more useful information than the first edition of this book. The guide contains a wealth of information on everything from keyboard variations to a detailed description of PET and CBM memory.

Included are:

Complete operating instructions for
- keyboard
- tape cassette
- disk

Description of all CBM BASIC statements
Optimal programming techniques including
- input/output programming
- file handling
- screen editing

Solutions to programming problems
CBM capabilities and limitations

#55-1 $15.00

Book/Cassette/Disk	Price	Quantity	Amount
27-6 6502 Assembly Language Programming	$16.99		
30-6 PET Personal Computer Guide	$15.00		
31-4 PET and the IEEE 488 Bus (GPIB)	$15.99		
40-3 Some Common BASIC Programs PET/CBM ed. (book)	$14.99		
25-X Some Common BASIC Programs PET Cassette	$15.00		
33-0 Some Common BASIC Programs PET Disk	$22.50		
38-1 Practical BASIC Programs	$15.99		
California resident tax			
Shipping			
Total Amount Enclosed			

California residents add 6% sales tax.
S.F. BART residents add 6½% sales tax.

Shipping: (Shipping for large orders to be arranged)
- All foreign orders $4.00 per book for airmail
- $0.45 per book 4th class in the U.S. (allow 3-4 weeks)
- $0.75 per book UPS in the U.S. (allow 10 days)
- $1.50 per book special rush shipment by air in the U.S.

Cassettes and Disk:
- No additional charge in the U.S.
- $1.50 each foreign airmail

OSBORNE/McGraw-Hill
630 Bancroft Way, Dept.G5
Berkeley, California 94710
(415) 548-2805 · TWX 910-366-7277

Name: ______________________

Address: ______________________

City: ______________________

State: ____________ ZIP: ____________

Phone: ______________________

☐ VISA ☐ Master Card

Credit Card No.: ______________________

McGraw-Hill

An Applications Commentary
Stimulating Simulations

Gregory R. Glau
Prescott, AZ

Well, there it sits: *your computer.* You've spent the past few months learning BASIC, writing all kinds of programs and learning how to use and interact with the computer and its tape or disk storage system. And, if you do say so yourself, you've turned into a pretty darned good programmer!

But gosh — after all those months — *there it sits.*

Sure, it's still fun to demonstrate a game or two if a friend stops by, or perhaps you've invested in a modem and can access a Network... but by and large, if you made a list of the reasons why you bought your computer in the first place, a list of 'all the things I'll be able to do,'... well, *you've done them.* You find yourself spending a half hour or so a day working on the computer, perhaps keeping your checkbook up-to-date, or making a budget listing, or keeping track of the amortization for the new car... but it mostly just sits there!

Suddenly it dawns on most of us that finding the answers is not the problem — we understand BASIC enough and disk data files and tape loading that we can figure out an answer to a problem. The difficulty is in finding the problem itself, in *asking the questions,* in figuring out things we can have the computer do. And not just ideas that take ten minutes and display a cute drawing on the screen, or a program to print all odd numbers between ten and a thousand. The whole purpose of any computer is to save time and make us more efficient in our work and/or home affairs.

But where do you get ideas to — as the ad says — simplify your life?

The first place to look is to examine any and all paperwork you handle, whether you use your computer in your home or business.

The businessman has some obvious needs — invoicing, monthly statements, payroll. We've found that our APPLE II saves an hour or so every week by figuring and printing payroll checks. So-so. But it also automatically balances all the figures and keeps them on a disk for all employee's year-to-date totals. The old way, balancing those figures by hand every quarter, literally took hours and hours. Here's a case where the initial time-saving didn't seem too terrific, but since everything is always 'in-balance' and up-to-date, over a period of a year it'll save hundreds of dollars in labor costs.

Accounts receivable and accounts payable are obvious savings, compared to the way we used to do things (and many small businesses still do) — by hand posting. Right out of the Middle Ages! Sending statements used to take a day... now it takes two hours.

But the businessman has to take a closer look at the other paperwork he's involved with.

How about keeping mileage and cost-per-mile records for any vehicles you own (the homeowner can do the same)?

While the businessman is making sales projections on his computer, the homeowner can project a budget/expense program on his.

The difficulty is in finding the problem itself, in asking the questions, in figuring out things we can have the computer do.

The businessman can keep a running record of each employee's job efficiency (is he making or losing money for you?)... and the homeowner can keep a record of what his wife (or husband) spends!

The businessman is able to *project* what a major investment will do to his cash flow and net profit, and the homeowner, with the right program, can readily tell if he can afford that Coleman tent-trailer.

The businessman can see exactly what will happen to his profits *if* sales drop ten percent. *If* the housing industry stays in its slump. *If* the Summer gets hot and his air conditioning units sell like crazy. And the homeowner can predict his cash situation nine months from now when the new arrival is due (they don't let you take babies home unless they're paid for!).

Could you forecast the weather based on past trends and current data?

Would you like to know who's got the best chance of winning the second race tomorrow?

Are you interested in your youngster's projected SAT test score?

Could you plan what you'd do if your health insurance costs increased 15% next month?

Would it help to know that if you spent X dollars for insulation that you'd save Y dollars on air conditioning costs? Or if you install that solar water heater what effect it'll have on your tax return?

Is it useful to understand that if you could somehow *save* so many dollars per month over the next so many months that you could save X dollars in interest for that awning/cooler/television/exercycle you wanted, by paying cash instead of financing it?

Wouldn't it be interesting to program your computer to analyze the miles you and your next-door neighbor and Joe (he lives down the block) and your brother-in-Law around the corner each drive to work every day? To perhaps route a carpool for the four of you? To get an exact projection of the gas you'd save? To perhaps do the same thing with you wife and her friends...who all drive their kids to school? To program your computer with the basic items your family uses (Cheerios, ketchup, kleenex, soup,

Anything that you now do with data can be adapted into a simulation, a projection of the future and its results on you.

whatever), along with the average rate of usage... and then printout a grocery list *before* you ran out of anything?

Anything that you now do with *data* can be adapted into a simulation, a projection of the future and its results on you.

This — the area of simulation — is perhaps the most powerful thing a computer can do for any of us. Yes, it's wonderful to have the computer keep track of our monthly bills, to have it do the mundane record-keeping chores, but how much more invaluable it becomes when we project the future with our ideas!

Let's see...if we're making X number of dollars per year now, and inflation will average Y percent over the next so many years and my raises will give me a total income of X dollars at that time...will the kids be able to go to College? Will the wife and I be able to buy that motorhome?

The whole idea is to project — and thus *predict* — the future.

Now what if I get laid off work for ten days over the next year? What if the wife could find a part-time job (perhaps running ammortization schedules for banks in your area on your Computer?) — what would her income do to our net total? How about that duplex we've been wanting to buy and fix up? If we pay so much for it and it costs us this much to fix it up...we should be able to rent it for... And what will it do to our income tax situation to get all that depreciation from it as a rental? How much would I have to save every month to have enough to pay cash for a new car, say, in twenty months?

Well, you get the idea — *simulation* on your computer can help you find out what the future will bring, and perhaps in time to allow you to change things, if you don't like what it displays for you on your CRT!

The whole idea is to project — and thus predict — the future.

Most simulations, by the way, can be generated from past data. The businessman can project labor costs based on the jobs he sold last year. The homeowner can predict what his salary needs are by basing his estimates on last year's budget printout (My God, Helen, did we spend *that much* on shoes?)

Once you've exhausted all the record-keeping and paper-work handling and forms-filling-out things that you once did by hand (but now your APPLE II or TRS-80 or OHIO SCIENTIFIC or PET does better and faster), the logical place to turn to is this area of *simulation*. And after all, there are only so many record-keeping chores we have to take care of, and once they're accomplished — and the computer is being used only an hour or so a day — the ideal place for one's creativity is in simulation.

And the best part is that you'll quickly discover that one idea leads to another — you might start projecting your net income and end up looking at life insurance values in relation to education costs ten years down the road. This in turn might lead to new record-keeping ideas, which will give you more simulation directions...

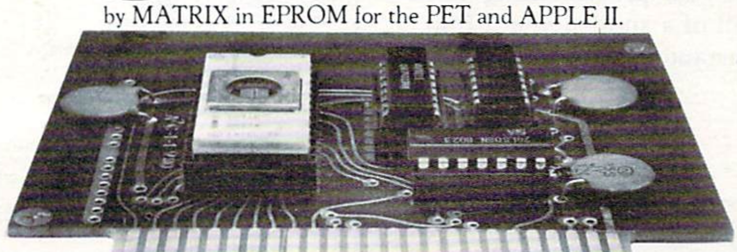
So, here are a baker's dozen ideas for **simulation** in areas the average programmer should have an interest in (after all, it's your money) in addition to the ideas already mentioned, so that perhaps a few will sound good to you and be of some help in your own financial planning:

1. Design a program to show you how much money you'd have to save weekly/monthly/yearly to end up with X dollars Y number of years from now. By being able to change every combination, you'll soon find a plan you can afford that'll give you the cash you need...when you need it.

2. Project the cash savings by replacing your present air conditioning system/furnace/water heater/cookstove with a new energy-savings one. How long will it take to pay for itself at the present gas or electric rates? What if the rates increase X percent?

4. Printout the values of your stock portfolio if inflation goes up so much percent while the market goes down X percent.

3. Compare the overall costs of remodeling the basement vs. the cost of a new home.



5. Figure out what salary you'll need with an annual raise of X dollars, an inflation rate of Y percent... when you son starts college in 19—.

6. Estimate the cost savings of adding an evaporative cooler to your air conditioning system.

7. Find out what's best for your own situation by comparing the net cost of life insurance: whole life gets cash value plus dividends while term insurance doesn't. Which costs less?

8. Find out how much of a raise you'll have to ask for based on X percent inflation this year.

9. Get a budget projection comparing the number of movies you go see (the cost of tickets & popcorn) to what that new cable TV deal costs.

10. Discover what your annual car expenses are over the next five years if you either (A) keep your present car or (B) buy that new gas-saving model. Factor-in different per-gallon costs, and don't forget the new one costs more for insurance and license plates.

11. Figure out exactly how much life insurance you need right now — determine how much income your wife will need for how many years.

12. If you can save so many dollars per month, display or printout the various options you might have (money market funds, regular savings, certificates of deposit, mutual fund programs); which is best for you?

13. Find out exactly what happens if you buy a rental unit, by using forced inflation. The duplex might cost you X dollars, but if you spend Y dollars to fix it up, how much will it increase in value? Compare this duplex with that triplex. Which is best? If you fix a place up and then keep it, how much can you raise the rents? What does it do to your tax return?

Somewhere in all these ideas is a problem that *you need a solution to.* So... get your programming pad out and get to work... and all of a sudden you'll find your computer in almost constant use... and not just sitting there any more!

Editor's Note: Once you have one of these programs up and running, write a tutorially descriptive article to go with it, and send it in to me at **COMPUTE!** *We'll look forward to it, and so will all our readers who aren't quite programmers yet. RCL.* ©

NEC spinwriter™

quality, reliability, and affordability is NOW available backed by service and support you can depend on.

Dealers, O.E.M.'s, and end-users all realize that service is the key ingredient of a wise purchase—SO DOES T-COM!

T-COM, a fully authorized NEC distributor, believes a successful sale doesn't end with the delivery of the product. We offer the service and support to keep your Spinwriter operating with the least amount of down time for you and your customers. With our fully factory trained service staff and unique service policies, we make every effort to keep your purchase from T-COM an investment you can count on to deliver satisfaction everyday.

T-COM represents a select group of dependable state-of-the-art word and data processing products and supplies. From Data-MATE and Smith Systems terminal furniture to NEC printers and supplies, you can count on the finest products available all backed by T-COM's firm commitment to service and support.

Dealer and end user inquiries are welcome for these fine products:

■ Data-MATE Terminal Furniture ■ Printer Interface Cables ■ Gates Sound Covers
■ NEC Spinwriter ■ Spinwriter supplies and accessories ■ Smith System Furniture

FOR INFORMATION ON DEALERSHIPS, PRODUCTS, AND SERVICE

Contact T-COM, INC.
Two North Plaza P.O. Box 3005
Jackson, Michigan 49204
Phone (517) 787-9280

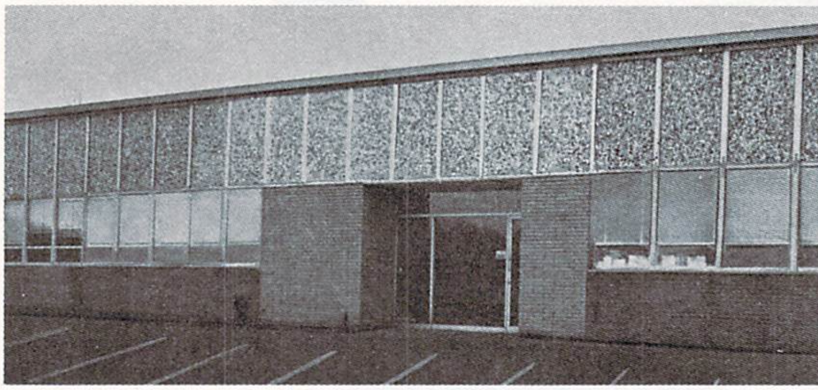
T-COM, INC.

a C, C&S Systems Company

"The ideal software source...

"We couldn't find it.

says Richard Taylor, noted software author and President, Programs Unlimited



Programs Unlimited national headquarters, Jericho, N.Y.

We know what it's like out there. Because that's where we came from.

Before we decided to become software entrepreneurs, we were just like you—micro-computer enthusiasts searching and sifting through myriad magazines for the ideal mail order software source.

What we hoped to find was a single entity that offered an ultra-wide selection of software (and peripherals) for our particular micro-computers. That pre-selected only the best of the many similar-sounding programs that reached the market each month. That could give us personal assistance with the purchase-decision process. And that stood behind its products with a no-holds-barred guarantee.

Isn't that what you've been looking for too?

When we couldn't find it, we decided to become it. And PROGRAMS UNLIMITED was born.

An Information Services Department: To Solve, Not Sell

We're professionally staffed and solidly financed. We set high standards for the programs we carry, whether developed by ourselves or others; if they don't meet our standards, we know they won't meet yours.

We have a computer Bulletin Board to keep you apprised of our latest offerings, and a toll-free number for telephone orders. In addition, we've established an Information Services Department; its primary function is not to sell merchandise, but to provide expert answers—from Richard Taylor and others—to questions you may have about your system.

Our Guarantee is *Really* a Guarantee

We know who we are and what we're up to, but **you** may not—yet. That's why we give you the best guarantee we can think of: if for any reason you're not satisfied with any PROGRAMS UNLIMITED purchase, return it within 14 days for a full, no-hassle refund.

Think we're going overboard? You're right.

We know that the future of our business is **repeat** business. We want you to be satisfied first time, second time, **every** time you buy.

We want to earn the right to be **your** *software source.*

the software source™

PROGRAMS

The Commodore VIC-20: A First Look

David D. Thornburg
Innovision
P.O. Box 1317, Los Altos, CA 94022

It seems surprising to realize that it was only three years ago when the Commodore PET started to show up in dealer's showrooms. At a price of $799, this astounding machine (along with its temporal counterparts from Apple, Radio Shack, and others) broke down many price/performance barriers in the computer field.

It wasn't much later that people started wondering if the small computer industry was going to copy the calculator industry, with ever more sophisticated machines being made available at the same, or lower, cost than that of earlier models. Mythical price barriers were erected, only to be smashed by new product announcements.

It was the breaking of the $400 barrier last fall by Radio Shack's TRS-80 Color Computer that caused some industry observers to predict the final arrival of the true "consumer" computer (see the review on this machine which appeared in the November-December 1980 issue of **COMPUTE!**). But almost before the TRS-80 Color Computer (which I will refer to as the TRS-80 from now on) was in full production, Commodore announced the collapse of yet another barrier with the introduction of the VIC-20, priced at only $299.

For those of you who have yet to see a photograph of the VIC, it is about the smallest size a computer could be and still have a full-sized keyboard. This compact size makes the VIC fit easily into almost any imaginable home location — an important feature which other manufacturers have yet to understand.

Before going into details, I want to mention a little about the "ambience" of the VIC. Those of you who are familiar with the PET will find many of the good PET features on the VIC. Running a program from tape, for example, requires merely pressing SHIFT RUN and the cassette PLAY button. The full screen editor (using cursor control keys) is supported by the VIC, as is the PET graphics character set.

In a move which is certain to guarantee much support from the indigenous PET software community, Commodore even kept the tape formats identical so that PET programs could be loaded directly into the VIC. Most of the PET programs I have run on the VIC required only a few lines of revision to work perfectly. This suggests that outside software support for the VIC will appear instantaneously upon its arrival in the marketplace.

As a sign of Commodore's attention to detail in this area, both the user port and the cassette connector are identical to their PET counterparts, so many plug-in peripherals for the PET will plug into the VIC as well.

An IEEE-488 interface (standard with the PET) is available as an add-on for the VIC. Except for this interface, the PET and VIC interface environments are quite similar.

That the VIC-20 is an astounding machine for

...it is about the smallest size a computer could be and still have a full-sized keyboard.

the price is unquestioned. What we will try to do in this review is describe the VIC's capabilities and features in comparison with the machines with which it is likely to share the limelight — the TRS-80 (Color Computer) and the Atari 400.

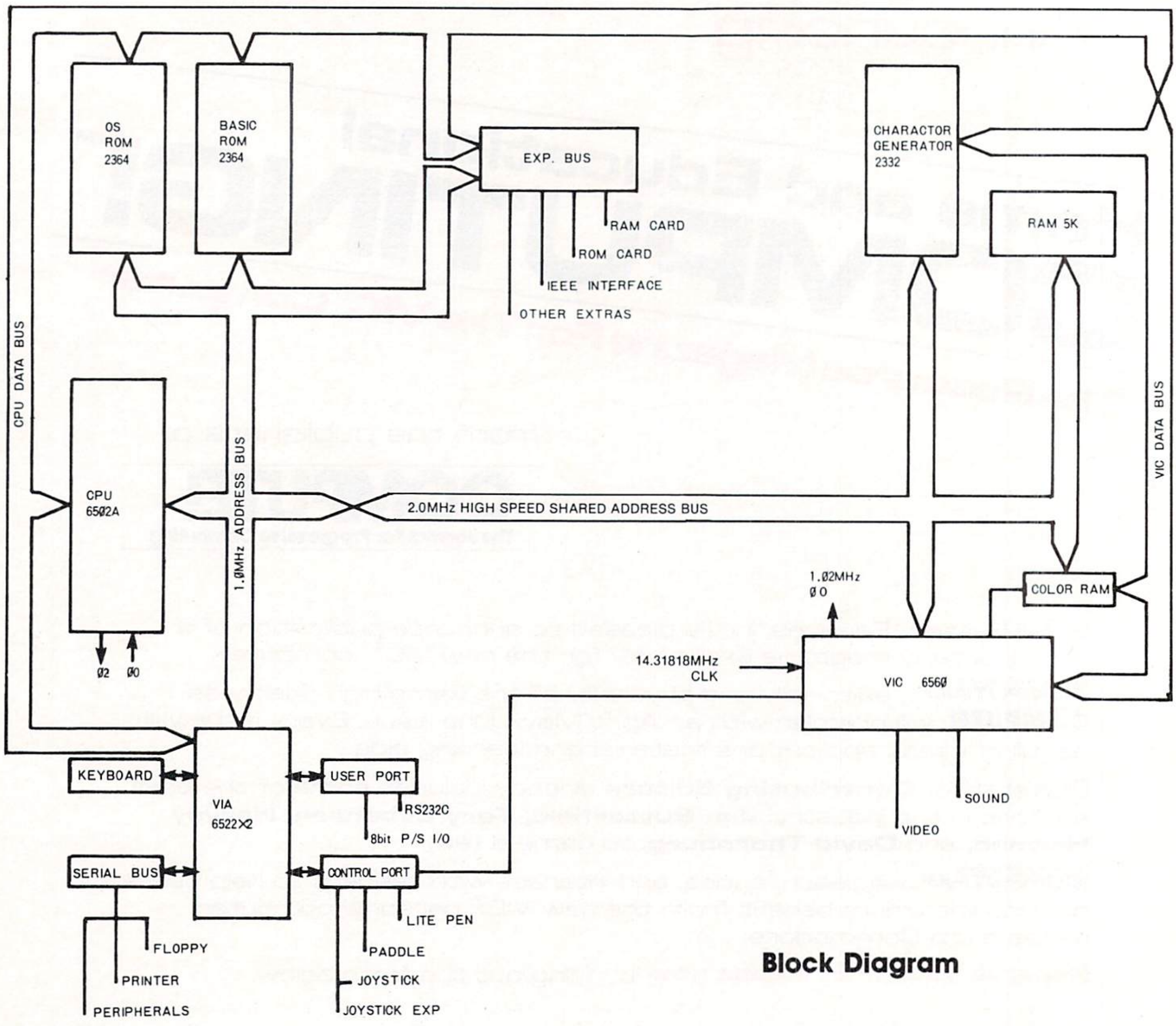
In order to provide some structure to this review, I have prepared a table which shows the salient features of each machine. This table is divided into four categories: OUTPUT, INPUT, EXTERNALS, and INTERNALS. We will discuss each of these in turn.

OUTPUT...

Communication from the computer to the user requires (for all three machines) a television set. All three computers support color and generate user programmable sounds which are heard through the TV loudspeaker. All three computers come with RF modulators, thus making the connection to the home television a moderately trouble-free task.

These similarities between machines should in no way be taken to imply that there are no substantiative differences between these computers, however; the differences are *most* important.

In the area of alphanumeric display, for example, the VIC displays a maximum of 23 lines of 22 characters, compared to 16 lines of 32 characters for the TRS-80, and 24 lines of 40 characters for the Atari 400. In terms of character display quality, I rank the machines in the order: Atari, VIC, TRS-80; with the VIC and Atari both having a very high quality display. It should be noted that the



Block Diagram

Atari has more display modes than either of the other computers, and has the ability to display 24 lines of 20 characters, and 12 lines of 20 characters as well as its default (24 x 40) arrangement.

Both the VIC and Atari 400 display upper and lower case characters, and have an alphamosaic (graphics character) display feature also. The TRS-80 does not.

True "bit map" graphics modes are available on all three machines, with each machine's format being unique. The *maximum* display resolution for the VIC or 176 x 176 pixels (picture elements) compares favorably with the TRS-80 limit of 256 x 192 and the Atari maximum resolution of 320 x 192 pixels.

In terms of color control, the Atari is way out in front, since it has independent control over hue and luminance to achieve 128 colors. For most normal programs, however, the Atari user can only work with any four of these 128 colors on the screen at a time, compared with eight fixed colors for the TRS-80 and 16 fixed colors for the VIC.

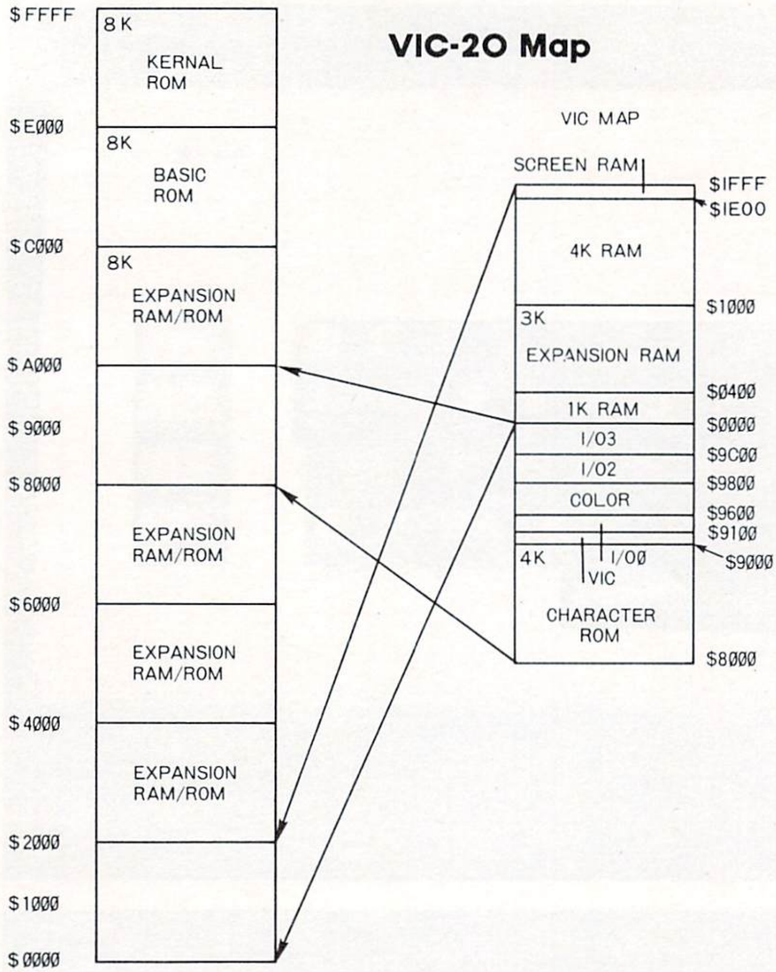
Just as color is important for many applications, the creative use of sound can do much to enhance one's programs. The VIC supports three musical tone generators (3 octaves each) and one "sound effects" generator. The TRS-80, by comparison, only supports one musical sound channel. The Atari 400 has four sound generators, each of which is capable of musical sounds (4 octaves), or a wide variety of user programable sound effects.

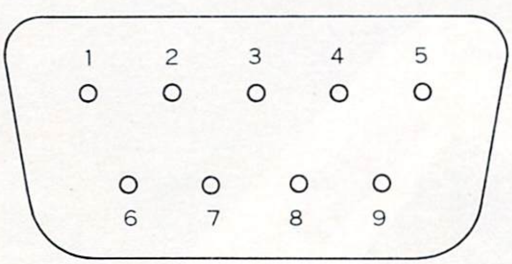
Input...

Those of you who appreciate nice keyboards will love the VIC-20. It is supplied with a 66 key arrangement with full typewriter-like key travel. Physically, the VIC keyboard resembles that on the Atari 800. The

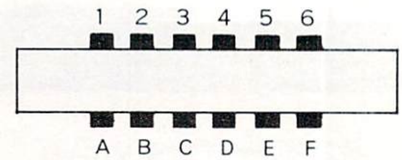

commodore
VIC20
COMPUTER
POWER


commodore
CONTROL PORT
OFF
ON
9V
AC

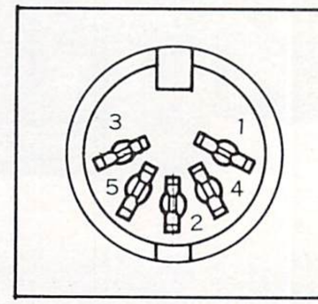




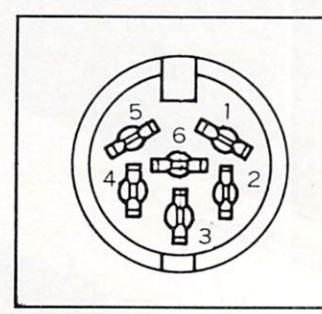
1	JOYØ	
2	JOY1	
3	JOY2	
4	JOY3	
5	POT Y	
6	LIGHT PEN	
7	+5V	MAX. 1ØØmA
8	GND	
9	POT X	



A-1	GND
B-2	+5V
C-3	CASSETTE MOTOR
D-4	CASSETTE READ
E-5	CASSETTE WRITE
F-6	CASSETTE SWITCH



1	+6V	1ØmA MAX
2	GND	
3	AUDIO	
4	VIDEO LOW	
5	VIDEO HIGH	



1	SERIAL SRQ IN
2	GND
3	SERIAL ATN IN/OUT
4	SERIAL CLK IN/OUT
5	SERIAL DATA IN/OUT
6	NC

TRS-80, on the other hand, has a medium travel "clicky" keyboard, and the Atari 400 (which places third in this comparison) has a membrane "micromotion" keyboard.

In terms of overall keyboard layout, I find that the keys on the Atari keyboard are slightly easier to find than those on the VIC, but presumably this is a result of my much longer experience with the Atari key arrangement.

Externals...

All three computers come with connections to support external cassette tape units for program and data storage. Unlike the Atari and Radio Shack pro-

ducts, the VIC provides its own power to the recorder, thus reducing the number of power connections needed to get everything running. This is an important consideration when one realizes that most power outlets in the house have two sockets on them. With the VIC connected to a television set, both power outlets are used. With the Atari and Radio Shack computers, a third outlet is needed to connect the tape recorder.

Floppy disk drives are plannned for the VIC and TRS-80, and are available for the Atari 400. All three machines support printers, and can be connected (through an external modem and coupler) to the telephone line. This latter feature will be a crucial test for *all* modern computers as the growth of information utilities continues.

Additional input devices are also supported by the three computers. The VIC supports one (Atari-like) joystick, while the TRS-80 accepts two joysticks, and the Atari 400 accepts up to four. Rotary paddles are also supported on all three machines. The VIC can handle one, the TRS-80 can accept four, and the Atari 400 can accept up to eight game paddles. Both the VIC and the Atari 400 work with a light pen, but the TRS-80 does not.

As you can see, the VIC has many features which are not available on the TRS-80.

Internals...

The VIC uses the 6502 microprocessor, but has one important architectural difference from the PET. In keeping with the designs of the TRS-80 and Atari 400, the VIC uses a special display controller chip (after which the VIC was named). The use of dedicated graphics chips is most beneficial to the end user, since it takes some of the load off the processor and makes the computer capable of feats which would otherwise be quite hard to perform in an eight-bit machine.

The entry level RAM of 5 Kbytes (of which 3583 bytes are available for program space) places VIC above the TRS-80 (4 Kbytes of RAM) and under the Atari 400 (8 Kbytes of RAM). In what appears to be a unique packaging idea, the VIC has a single external cartridge slot which accepts combinations of RAM and ROM. A 3 Kbyte RAM expander will be available from Commodore soon. This plug-in cartridge also has sockets for up to 24 Kbytes of ROM.

The VIC BASIC follows in the Microsoft tradition, and has the same "feel" as the PET BASIC, while adding capabilities associated with the color and sound features of the VIC.

To get some idea for the execution speed of VIC BASIC language, I ran the following program on the VIC, the TRS-80 Color Computer and on the Atari 400.

```
5 FOR J = 1 TO 100
10 FOR I = 1 TO 100
20 A = (I*I)/I
40 NEXT I
50 NEXT J
```

The execution times (in seconds) are shown below for each computer.

VIC-20	77 seconds
TRS-80 Color Computer	103 seconds
Atari 400	159 seconds

As can be seen from these figures, the VIC has the clear lead. Next I added one line to this program:

```
30 PRINT A
```

and ran the experiment again with the following results:

VIC-20	347 seconds
TRS-80 Color Computer	280 seconds
Atari 400	540 seconds

This time the Radio Schack entry is the leader of the BASIC race.

Additional Comments...

From the comparisons shown above, it is pretty clear that the VIC will provide very stiff competition to the TRS-80 Color Computer. To help cement the VIC in the marketplace, Commodore has announced a plan by which they will help cottage industry software developers in their development of cartridge-based firmware. It is as though Commodore genuinely realizes the positive impact of outside software

Table I

	VIC	TRS-80	Atari 400
Price:	$299	$399	$499
OUTPUT			
Display Incl.:	NO	NO	NO
Color Display:	YES	YES	YES
Characters:	22x23	32x16	40x24
Upper/Lower:	YES	NO	YES
Alphamosaic:	YES	NO	YES
Bit Map:	176x176	256x192	320x192
Sound Gen.:	4	1	4
INPUT			
# Keys:	66	53	57
Kbd. Type:	Full Stroke	Med. Stroke	Flat Panel
EXTERNALS			
Tape:	YES	YES	YES
Disk:	SOON	SOON	YES
Printer:	YES	YES	YES
Joysticks:	1	2	4
Light Pen:	YES	NO	YES
Paddles:	1	4	8
Modem:	YES	YES	YES
INTERNALS			
Processor:	6502	6809	6502
Display Chip:	YES	YES	YES
RAM:	5K	4K	8K
ROM:	?	8K	18K
MAX RAM:	see text	16K	16K
MAX ROM:	see text	16K	26K
Plug-in ROM:	YES	YES	YES

on hardware sales — a fact which Texas Instruments, for example, has only begun to appreciate.

I expect that the VIC will be quite popular with children, and that it will thus find its way into primary grade classrooms as well as into homes. Personally, I find the 22 character display to be too small to support any but the most rudimentary business applications — perhaps including home finance applications. But at a price of $299, that is hardly the point. The VIC is a much more valuable computer literacy tool than either the hand held Sharp computer being sold by Radio Shack, or any of the similarly priced single board computers which have been on the market for several years. It comes with a very fast, high quality BASIC, and with some excellent graphic and sound capabilities.

VIC will create its own market, and it will be a big one.

©

Editor's Note: Two vendors have recently announced memory expansion modules for the Atari 400. These modules bring the max RAM capacity of the 400 to 32K. Both the VIC-20 and the TRS-80 Color Computer have built-in RS-232C serial ports; the Atari 400 does not.

Advice to PET Owners:

How To Be A VIC Expert

Jim Butterfield, Toronto

There are going to be a lot of VIC computers arriving very soon. All those new VIC owners are going to look to you for advice and counsel. After all, you've owned a PET for several months; and a VIC is just a junior version of a PET, right?

You don't want to blow your chance to become the block's VIC guru. It really is very much the same as the PET, but there are new things you will need to get used to.

I'll give a few hints here: Basic is the same as on the PET, right down to the LOG, SQR, and ATN functions. Because the VIC has fewer columns, you can link up to four rows together to generate a line of Basic. Many of the advanced VIC features, like color, are done the same way as programmed cursor on the PET. You clear the PET screen by printing a special reverse-character; you'll set color on the VIC exactly the same way.

The old familiar PEEK the POKE locations have moved around; when you find them, they will work the same as in the PET...but that's a whole other story.

A good way to start is with a sample program. Here's a VIC program which you can type in on your PET, and save on tape. It won't work properly on the PET, but your tape will load in nicely on the VIC, and you'll get an insight into how some of the VIC things work. By the way, your cassette unit itself will plug into the VIC, so carry both tape and cassette unit over to the new machine.

```
100 remark: big letter display
110 rem    by jim butterfield
120 rem   peeks the VIC character generator
130 rem   in hex 8000 to 8FFF
140 input "graphic/text";g$
150 g=asc(g$)
160 if g=71 then b=0:goto 190
170 if g=84 then b=2048:goto 190
180 stop
190 input"character #";n
200 if n<0 or n>255 or n<>int(n) goto 140
210 m=32768+b+8*n
220 print chr$(176);:for j=1 to 8:print chr$(192);:
next j:print chr$(174)
230 for j=m to m+7
240 print chr$(221);
250 x=peek(j)
260 for k=1 to 8
270 y=146:x=x*2:if x>255then x=x-256:y=18
280 print chr$(y);chr$(32);
290 next k
300 print chr$(146);chr$(221)
310 next j
320 print chr$(173);:for j=1 to 8:print chr$(192);:
next j:print chr$(189)
330 goto 140
```

What does the above program do? It prints out the 256 characters used by VIC in large size. The user picks a character (from 0 to 255) and it is displayed on VIC's screen. The characters are screen format, not ASCII, so that a value of 1 gives an A character. You'll find that the characters are similar to those used by the PET.

How does this program behave in the VIC? You'll find a few differences that will cause you to change your programming style when you shift to the new machine. The first thing you'll notice is that you'll have trouble stopping the program. The INPUT statement on line 140 does not stop the machine if you press RETURN with no input. It continues running, and leaves variable G$ at its previous value. That's different: it means that you can set up a default for G$ and the user can invoke it by just pressing RETURN. It also means that you have to find another way of stopping the program. The trick here is to input a character such as X when asked, "GRAPHIC/TEXT". The program will continue only on a response of G or T - or no response, as noted before.

You'll have noticed that the program is very PET-compatible. In fact, it will run on the PET with two small differences. First, the PET can't read its own character generator, so you'll get nonsense displayed. Secondly, the PET behaves a little differently on the INPUT statement as we have noted.

Here's a puzzler: when you punched up your program on the PET, it occupied memory space starting at decimal 1025. On the VIC, the program will want to take up residence starting at decimal 4096. How can your PET program load properly to the VIC? Easy: VIC has a relocating loader; it just moves the program to the new place. Transferring programs the other way — from the VIC to the PET — isn't as easy, since the PET does not relocate programs.

A final note on the coding. There's a lot of use of CHR$ characters: why didn't I use the more familiar characters in quotes, which would certainly work? Answer: it would drive the staff of **COMPUTE!** wild, since they wouldn't be able to typeset all those fancy characters. Then they would substitute their own symbols, with a translation legend somewhere near, and you'd be driven wild in turn trying to type it in. Trust me: it's better this way. As an exercise, you can work out how to recode most of the CHR$ expressions into screen characters.

Thought for the day: if the character generator is accessible in memory, do you think that you might be allowed to code your own set of characters in RAM memory? The answer, of course, is yes; but you'll have to encode the whole character set you need since all characters must be grouped together. But that's another story...

You've generated your first VIC program. Hopefully, you've discovered a few things about how the VIC works. Much of it will be the same as with the PET, but a few features are different.

Now, when all of the new VIC owners on the block beat a path to your door, practice saying wise things like, "Of course, on the big machines, we do it this way..." ©

Basically Useful BASIC

Ascending/ Descending Sort

Rick Keck
Overland Park, KS

At some point in time, every computer system user needs the services of a sort program. There has been much work done in the field of "sorting theory" and from this there has come a variety of different sorting methods. Some of these include the bubble sort, shell sort, binary sort, and tag sort. The benefit of this exists in the ability to select a method which is best for sorting data based upon the data's unique characteristics (if any). The factors which influence the decision of which sorting method to use include the following data characteristics: volume, relative order, and storage form (random access or sequential files). In a majority of cases, a simple sorting method will work fine. The standard order of sorting data is to have it sorted from smallest to biggest (ascending order). On occasion, sorting of data may need to be done from biggest to smallest (descending order). The following modified bubble sort routine allows the data to be sorted in either ascending or descending order. Note that the data is handled by character string variables so as to allow alphanumeric data to be sorted.

```
100 REM*****************************
110 REM*                           *
120 REM* ASCENDING / DESCENDING    *
130 REM*                           *
140 REM*      SORT  ROUTINE        *
150 REM*                           *
160 REM*  BY  RICK KECK  01/81     *
170 REM*****************************
180 DIM C$(100)
190 DATA "JOHN","BILL","MARY"
200 DATA "CAROLINE","FRED","SUE"
210 DATA "JOE"
220 REM* N HOLDS # OF DATA
230 N=7
240 REM* READ DATA INTO C$
250 FOR J=1 TO N
260 READ C$(J)
270 NEXT J
280 REM* ASCENDING OR DESCENDING
290 PRINT:PRINT:PRINT
300 PRINT:PRINT"WHAT ORDER DO YOU WISH TO SORT:"
310 PRINT:PRINT"      A - ASCENDING (SMALL TO BIG)"
320 PRINT:PRINT"      D - DESCENDING (BIG TO SMALL)"
330 PRINT:PRINT
340 INPUT A$
350 IF A$<>"A" AND A$<>"D" THEN 340
360 PRINT:PRINT
370 REM*****************************
380 REM*      SORT  BEGINS         *
390 REM*****************************
400 FOR K=1 TO (N-1)
410 IF A$="A" THEN 440
420 IF C$(K)>=C$(K+1) THEN 540
430 IF A$="D" THEN 450
440 IF C$(K)<=C$(K+1) THEN 540
450 FOR J=K TO 1 STEP -1
460 IF A$="A" THEN 490
470 IF C$(J)>=C$(J+1) THEN 540
480 IF A$="D" THEN 500
490 IF C$(J)<=C$(J+1) THEN 540
500 T$=C$(J)
510 C$(J)=C$(J+1)
520 C$(J+1)=T$
530 NEXT J
540 NEXT K
550 REM*****************************
560 REM*       SORT  ENDS          *
570 REM*****************************
580 FOR L=1 TO N
590 PRINT C$(L)
600 NEXT L
610 PRINT:PRINT"NORMAL TERMINATION"
620 END
```

```
BILL
CAROLINE
FRED
JOE
JOHN
MARY
SUE
```

```
SUE
MARY
JOHN
JOE
FRED
CAROLINE
BILL
```

©

Program Compactor

Edward H. Carlson
Okemos MI

There are two evils that sneak up on you as your programs attain moderate length. The programs begin to take up too much space in memory, and they become increasingly obtuse. These evils combine in positive feedback. Increased internal documentation by **REM**ark statements is a partial antidote to program complexity but this, of course, compounds the problem of fitting the whole glob into memory.

The answer is to have two copies of each program: a "working copy" occupying minimum space, and a fully documented "archives copy" that may,

The first 2 bytes of a BASIC line are a pointer to the start of the next BASIC line.

in fact, be too long to be run in your machine. (It must be short enough to fit in your memory as source code, sans variable tables, of course.)

But consider, you say, how much finger tapping, eyeball twitching, and obsessive concentration it takes to go through a program and remove all the REMarks, especially those buried inside lines of active code, and most especially those "invisible" REM statements like this one:

```
120 GOTO 232:NO "REM" NEEDED HERE
```

(In such a statement, the BASIC interpreter jumps to another line before passing the colon and so does not detect the syntax error caused by the omitted REM.)

Looky here, I say, repetitive decisions are just what the logic machine was invented to perform! One needs to write a "program compacting" program, and that is just what I have done, showing my results in Listing 1. The program was written for use with my Ohio Scientific C2-4P, but should work with little change in other Microsoft BASIC machines, such as the PET. All that needs changing is the starting address of the source code, $0300 for OSI BASIC, and the numerical values of the tokens, which differ in the OSI and PET versions of Microsoft BASIC.

This compactor is a moderately complex program in itself. It is put at high line numbers so as to be out of the way of any program you are writing. When you have a version of your own program that needs compacting, **first save it to tape,** then read in the Compactor (from a tape that does not have the Test Program in front). Do a "RUN 62000". The compacted program will then be POKEd into memory, ready to SAVE to tape or to RUN. The Compactor will still be in memory, but now invisible to you and inaccessible to BASIC.

Listing 1 starts off with a very short Test Program that has most of the features that would give trouble in a poorly contrived Compactor. Then follows the compactor itself. After initializing addresses, etc., a loop over I is started. Each time through, one line is compacted. Line 62036 contains the exit from the compacting process. This occurs when the line number to be processed is above 9999. You may wish to change this, but all my programs use only line numbers below 9000. Next, leading colons and spaces are removed. I haven't used such things in my own code, but it is legal and so I include that case in the Compactor Program.

Following these preliminaries, the program enters a loop over K, at line 62050, which walks through a single line. Spaces are removed, and the line is terminated if a REM, STOP, RETURN, or GOTO is encountered. The compacted line is stored in an array called L(I). This is an artifact left over from the program construction period. Before allowing my infant program to actually POKE into tender source code memory, I had it make a string and print it. Upon reaching voting age, the string became the L array.

During all this, it is necessary to keep a sharp eye out for quotation marks, as you do not want to alter any of the text inside them. Line 62080 detects opening quotes and jumps to a routine to march along looking for the closing quotes so that control can be returned to the main loop. If a colon or a null is found before the closing quotation marks, the statement or line has terminated and analysis of the next is begun.

Every Microsoft BASIC line ends in a null. Detection of a null character sends control to the top of the "I" loop. The next command sends control to the subroutine at 62600 where the compacted line is POKEd into memory. Some tricky address changing is needed here. The first 2 bytes of a BASIC line are a pointer to the start of the next BASIC line. This chain of pointers must remain intact during interpretation of any part of the Compactor that would do a line number search. Such a search would start at the first line of the program to be compacted, even though the code being interpreted is all above line number 62000. So lines 62601 and 2 pick up the starting address of the code that is next to be compacted in POKEs it into the first two bytes of the newly compacted line.

DYNACOMP

Quality software for⁺:

ATARI	TRS-80 (Level II)*
PET	NORTH STAR
APPLE II Plus	CP/M 8" Disk

GAMES, SIMULATIONS, EDUCATION and MISCELLANEOUS

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

HEARTS 1.5 (Available for all computers) — **Price: $14.95 Cassette/$18.95 Diskette**
An exciting and entertaining computer version of this popular card game. Hearts is a trick-oriented game in which the purpose is not to take any hearts or the queen of spades. Play against two computer opponents who are armed with hard-to-beat playing strategies.

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

POKER PARTY (Available for all computers) — **Price: $17.95 Cassette/$21.95 Diskette**
POKER PARTY is a draw poker simulation based on the book, POKER, by Oswald Jacoby. This is the most comprehensive version available for microcomputers. The party consists of yourself and six other (computer) players. Each of these players (you will get to know them) has a different personality in the form of a varying propensity to bluff or fold under pressure. Practice with POKER PARTY before going to that expensive game tonight! Apple Cassette and diskette versions require a 32 K (or larger) Apple II.

VALDEZ (Available for all computers) — **Price: $14.95 Cassette/$18.95 Diskette**
A simulation of supertanker navigation in the Prince William Sound and Valdez Narrows. The program uses an extensive 256X256 element radar map and employs physical models of ship response and tidal patterns. Chart your own course through ship and iceberg traffic. Any standard terminal may be used for display.

FLIGHT SIMULATOR (Available for all computers) — **Price: $17.95 Cassette/$21.95 Diskette**
A realistic and extensive mathematical simulation of take-off, flight and landing. The program utilizes aerodynamic equations and the characteristics of a real airfoil. You can practice instrument approaches and navigation using radials and compass headings. The more advanced flyer can also perform loops, half-rolls and similar aerobatic maneuvers.

CRIBBAGE 2.0 (TRS-80 only) — **Price: $14.95 Cassette/$18.95 Diskette**
This is a well-designed and nicely executed two-handed version of the classic card game, cribbage. It is an excellent program for the cribbage player in search of a worthy opponent as well as the beginner wishing to learn the game, in particular the scoring and jargon. The standard cribbage score board is continually shown at the top of the display (utilizing the TRS-80's graphics capabilities), with the cards shown underneath. The computer automatically scores and also announces the points using the traditional phrases.

CHESS MASTER (North Star and TRS-80 only) — **Price: $19.95 Cassette/$23.95 Diskette**
This complete and very powerful program provides five levels of play. It includes castling, en passant captures and the promotion of pawns. Additionally, the board may be preset before the start of play, permitting the examination of "book" plays. To maximize execution speed, the program is written in assembly language (by SOFTWARE SPECIALISTS of California). Full graphics are employed in the TRS-80 version, and two widths of alphanumeric display are provided to accommodate North Star users.

STARTREK 3.2 (Available for all computers) — **Price: $ 9.95 Cassette/$13.95 Diskette**
This is the classic Startrek simulation, but with several new features. For example, the Klingons now shoot at the Enterprise without warning while also attacking starbases in other quadrants. The Klingons also attack with both light and heavy cruisers and move when shot at! The situation is hectic when the Enterprise is besieged by three heavy cruisers and a starbase S.O.S. is received! The Klingons get even!

SPACE TILT (Apple only) — **Price: $10.95 Cassette/$14.95 Diskette**
Use the game paddles to tilt the plane of the TV screen to "roll" a ball into a hole in the screen. Sound simple? Not when the hole gets smaller and smaller! A built-in timer allows you to measure your skill against others in this habit-forming action game.

GAMES PACK I (Available for all computers) — **Price: $9.95 Cassette/$13.95 Diskette**
GAMES PACK I contains the classic computer games of BLACKJACK, LUNAR LANDER, CRAPS, HORSERACE, SWITCH and more. These games have been combined into one large program for ease in loading. They are individually accessed by a convenient menu.

GAMES PACK II (Available for all computers) — **Price: $9.95 Cassette/$13.95 Diskette**
GAMES PACK II includes the games CRAZY EIGHTS, JOTTO, ACEY-DUCEY, LIFE, WUMPUS and others. As with GAMES PACK I, all the games are loaded as one program and are called from a menu.

Why pay $7.95 or more per program when you can buy a DYNACOMP collection for just $9.95?

NOMINOES JIGSAW (ATARI and TRS-80 only) — **Price: $16.95 Cassette/$20.95 Diskette**
NOMINOES JIGSAW is an intriguing and sophisticated graphical puzzle. The jigsaw consists of a 9 by 9 board partially filled with randomly chosen shapes (nominoes), of which there are 60 types. By knowing that the shapes must be legally connected, and by guessing the shape at each location, all the nominoes may be eventually deduced. Scoring is based on the number of guesses required and the difficulty of the board set-up.

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

BLACK HOLE (Apple only) — **Price: $14.95 Cassette/$18.95 Diskette**
This is an exciting graphical simulation of the problems involved in closely observing a black hole with a space probe. The object is to enter and maintain, for a prescribed time, an orbit close to a small black hole. This is to be achieved without coming so near the anomaly that the tidal stress destroys the probe. Control of the craft is realistically simulated using side jets for rotation and main thrusters for acceleration. This program employs Hi-Res graphics and is educational as well as challenging.

TEACHER'S PET I (Available for all computers) — **Price: $ 9.95 Cassette/$13.95 Diskette**
This is the first of DYNACOMP's educational packages. Primarily intended for pre-school to grade 3, TEACHER'S PET provides the young student with counting practice, letter-word recognition and three levels of math skill exercises.

CRYSTALS (ATARI only) — **Price: $ 9.95 Cassette/$13.95 Diskette**
A unique algorithm randomly produces fascinating graphics displays accompanied with tones which vary as the patterns are built. No two patterns are the same, and the combined effect of the sound and graphics are mesmerizing. CRYSTALS has been used in local stores to demonstrate the sound and color features of the Atari.

CRANSTON MANOR ADVENTURE (North Star only) — **Price: $19.95**
At last! A comprehensive Adventure game for the North Star. CRANSTON MANOR ADVENTURE takes you into mysterious CRANSTON MANOR where you attempt to gather fabulous treasures. Lurking in the manor are wild animals and robots who will not give up the treasures without a fight. The number of rooms is greater and the associated descriptions are much more elaborate than the current popular series of Adventure programs, making this game the top in its class. Play can be stopped at any time and the status stored on diskette. Requires 32K.

NORTH STAR SOFTWARE EXCHANGE (NSSE) LIBRARY — **Price $9.95 Diskette**
DYNACOMP now distributes the 20 + volume NSSE library. Most of these diskettes offer an outstanding value for the purchase price. Write for details regarding the contents of this library and quantity (four or more) purchases.

Availability

DYNACOMP software is supplied with complete documentation containing clear explanations and examples. Unless otherwise specified, all programs will run within 16K program memory space (ATARI requires 24K). Except where noted, programs are available on ATARI, PET, TRS-80 (Level II) and Apple (Applesoft) cassette and diskette as well as North Star single density (double density compatible) diskette. Additionally, most programs can be obtained on standard (IBM format) 8" CP/M floppy disks for systems running under MBASIC.

⁺ *ATARI, PET, APPLE II, TRS-80, NORTH STAR, CP/M and IBM are registered trade names and/or trademarks.*

BUSINESS and UTILITIES

MAIL LIST II (Apple and North Star diskettes only) — **Price: $24.95**
This many-featured program now includes full alphabetic and zip code sorting as well as file merging. Entries can be retrieved by user-defined code, client name or Zip Code. The printout format allows the use of standard size address labels. Each diskette can store more than 1100 entries (single density North Star or Apple DOS 3.2; over 2200 with double density North Star or Apple DOS 3.3)!

FORM LETTER SYSTEM (FLS) (Apple and North Star diskette only) — **Price: $17.95**
FLS may be employed to generate individually addressed form letters. The user creates the address file and separately composes the letter. FLS will then print form letters using each address. FLS is completely compatible with MAIL LIST II, which may be used to manage your address files.

FLS and MAIL LIST II are available as a combined package for $37.95.

TEXT EDITOR I (Letter Writer) — **Price: $14.95 Cassette/$18.95 Diskette**
An easy to use, line-oriented text editor which provides variable line widths and simple paragraph indexing. This text editor is ideally suited for composing letters and is quite capable of handling much larger jobs. Available for all computers.

PERSONAL FINANCE SYSTEM (ATARI only) — **Price: $34.95 Diskette**
PFS is a single disk menu oriented system composed of 10 programs designed to organize and simplify your personal finances. Features include a 300 transaction capacity; fast access; 26 optional user codes; data retrieval by month, code or payee; optional printing of reports; checkbook balancing; bar graph plotting and more. Also provides on the diskette is ATARI DOS 2.

FINDIT (North Star only) — **Price: $19.95**
This is a three-in-one program which maintains information accessible by keywords of three types: Personal (eg: last name), Commercial (eg: plumbers) and Reference (eg: magazine articles, record albums, etc). In addition to keyword searches, there are birthday, anniversary and appointment searches for the personal records and appointment searches for the commercial records. Reference records are accessed by a single keyword or by cross-referencing two or three keywords.

DFILE (North Star only) — **Price: $19.95**
This handy program allows North Star users to maintain a specialized data base of all files and programs in the stack of disks which invariably accumulates. DFILE is easy to set up and use. It will organize your disks to provide efficient locating of the desired file or program.

COMPARE (North Star only) — **Price: $12.95**
COMPARE is a single disk utility software package which compares two BASIC programs and displays the file sizes of the programs in bytes, the lengths in terms of the number of statement lines, and the line numbers at which various listed differences occur. COMPARE permits the user to examine versions of his software to verify which are the more current, and to clearly identify the changes made during development.

COMPRESS (North Star only) — **Price: $12.95**
COMPRESS is a single-disk utility program which removes all unnecessary spaces and (optionally) REMark statements from North Star BASIC programs. The source file is processed one line at a time, thus permitting very large programs to be compressed using only a small amount of computer memory. File compressions of 20-50% are commonly achieved.

GRAFIX (TRS-80 only) — **Price: $12.95 Cassette/$16.95 Diskette**
This unique program allows you to easily create graphics directly from the keyboard. You "draw" your figure using the program's extensive cursor controls. Once the figure is made, it is automatically appended to your BASIC program as a string variable. Draw a "happy face", call it H$ and then print it from your program using PRINT H$! This is a very easy way to create and save graphics.

TIDY (TRS-80 only) — **Price: $10.95 Cassette/$14.95 Diskette**
TIDY is an assembly language program which allows you to renumber the lines in your BASIC programs. TIDY also removes unnecessary spaces and REMark statements. The result is a compacted BASIC program which uses much less memory space and executes significantly faster. Once loaded, TIDY remains in memory; you may load any number of BASIC programs without having to reload TIDY!

STATISTICS and ENGINEERING

DATA SMOOTHER (Not available for ATARI) — **Price: $14.95 Cassette/$18.95 Diskette**
This special data smoothing program may be used to rapidly derive useful information from noisy business and engineering data which are equally spaced. The software features choice in degree and range of fit, as well as smoothed first and second derivative calculation. Also included is automatic plotting of the input data and smoothed results.

FOURIER ANALYZER (Available for all computers) — **Price: $14.95 Cassette/$18.95 Diskette**
Use this program to examine the frequency spectra of limited duration signals. The program features automatic scaling and plotting of the input data and results. Practical applications include the analysis of complicated patterns in such fields as electronics, communications and business.

TFA (Transfer Function Analyzer) — **Price: $19.95 Cassette/$23.95 Diskette**
This is a special software package which may be used to evaluate the transfer functions of systems such as hi-fi amplifiers and filters by examining their response to pulsed inputs. TFA is a major modification of FOURIER ANALYZER and contains an engineering-oriented decibel versus log-frequency plot as well as data editing features. Whereas FOURIER ANALYZER is designed for educational and scientific use, TFA is an engineering tool. Available for all computers.

HARMONIC ANALYZER (Available for all computers) — **Price: $24.95 Cassette/$28.95 Diskette**
HARMONIC ANALYZER was designed for the spectrum analysis of repetitive waveforms. Features include data file generation, editing and storage/retrieval as well as data and spectrum plotting. One particularly unique facility is that the input data need not be equally spaced or in order. The original data is sorted and a cubic spline interpolation is used to create the data file required by the FFT algorithm.

FOURIER ANALYZER, TFA and HARMONIC ANALYZER may be purchased together for a combined price of $44.95 (three cassettes) and $56.95 (three diskettes).

REGRESSION I (Available for all computers) — **Price: $19.95 Cassette/$23.95 Diskette**
REGRESSION I is a unique and exceptionally versatile one-dimensional least squares "polynomial" curve fitting program. Features include very high accuracy; an automatic degree determination option; an extensive internal library of fitting functions; data editing; automatic data and curve plotting; a statistical analysis (eg: standard deviation, correlation coefficient, etc.) and much more. In addition, new fits may be tried without reentering the data. REGRESSION I is certainly the cornerstone program in any data analysis software library.

REGRESSION II (PARAFIT) (Available for all computers) — **Price: $19.95 Cassette/$23.95 Diskette**
PARAFIT is designed to handle those cases in which the parameters are imbedded (possibly nonlinearly) in the fitting function. The user simply inserts the functional form, including the parameters (A(1), A(2), etc.) as one or more BASIC statement lines. Data and results may be manipulated and plotted as with REGRESSION I. Use REGRESSION I for polynomial fitting, and PARAFIT for those complicated functions.

MULTILINEAR REGRESSION (MLR) (Available for all computers) — **Price: $19.95 Cassette / $23.95 Diskette**
MLR is a professional software package for analyzing data sets containing two or more linearly independent variables. Besides performing the basic regression calculation, this program also provides easy to use data entry, storage, retrieval and editing functions. In addition, the user may interrogate the solution by supplying values for the independent variables. The number of variables and data size is limited only by the available memory.

REGRESSION I, II and MULTILINEAR REGRESSION may be purchased together for $49.95 (three cassettes) or $61.95 (three diskettes).

BASIC SCIENTIFIC SUBROUTINES, Volume I (Not available for ATARI)
DYNACOMP is the exclusive distributor for the software keyed to the text *BASIC Scientific Subroutines, Volume I* by F. Ruckdeschel (see the BYTE/McGraw-Hill advertisement in BYTE magazine, January 1981). These subroutines have been assembled according to chapter. Included with each collection is a menu program which selects and demonstrates each subroutine.

Collection #1: Chapters 2 and 3: Data and function plotting, complex variables
Collection #2: Chapter 4: Matrix and vector operations
Collection #3: Chapters 5 and 6: Random number generators, series approximations
Price per collection: $14.95 Cassette/$18.95 Diskette

All three collections are available for $39.95 (three cassettes) and $49.95 (three diskettes).

Because the text is a vital part of the documentation, *BASIC Scientific Subroutines, Volume I* is available from DYNACOMP for $19.95 plus 75¢ postage and handling.

ROOTS (Available for all computers) — **Price $9.95 Cassette/$13.95 Diskette**
In a nutshell, ROOTS simultaneously determines all the zeroes of a polynomial having real coefficients. There is no limit on the degree of the polynomial, and because the procedure is iterative, the accuracy is generally very good. No initial guesses are required as input, and the calculated roots are substituted back into the polynomial and the residuals displayed.

Ordering Information

All orders are processed and shipped postpaid within 48 hours. Please enclose payment with order along with computer information. If paying by VISA or Master Card, include all numbers on card. For orders outside North America add 10% for shipping and handling.

Add $2.50 to diskette price for 8" floppy disk (IBM format soft sectored, CP/M, Microsoft BASIC)
*TRS-80 diskettes are not supplied with DOS or BASIC.

Deduct 10% when ordering 3 or more programs.

Ask for **DYNACOMP** programs at your local software dealer. Write for detailed descriptions of these and other programs from **DYNACOMP**.

VISA

DYNACOMP, Inc. Dept. E
1427 Monroe Avenue
Rochester, New York 14618
24 hour mail order phone: (716)586-7579
Office phone (9AM-5PM EST): (716)442-8960

MasterCard

New York State residents please add 7% NYS sales tax.

Program Compactor

```
1 A=1:REM *** TEST PROGRAM ***
2 REM "RUN 62000" TO COMPACT THE TEST
PROGRAM
3 :  ::C=3:D=4:REM AAAAA
4 END:DON'T SEE THIS AFTER COMPACTION
5 RETURN:NOR THIS
6 GOTO 11111:NOR THIS
7 A$="SEE THIS":REM NOT THIS
999 STOP
62000 REM
62001 REM    *** COMPACTOR ***
62002 REM
62003 REM         Edward H. Carlson
62004 REM         3872 Raleigh Drive
62005 REM         Okemos MI 48864
62006 REM         (517) 349-1219
62007 REM
62010 PRINT:PRINT:PRINT "COMPACTING":
PRINT:PRINT
62015 DIM L(80):A=3*256:AP=A+1:AD=A-3
62020 FOR I=1 TO 9999:A=A+4:REM NEW
LINE
62025 IF L<>0 THEN GOSUB 62600
62035 L=PEEK(A-1)+PEEK(A)*256:AN=0
62036 IF L>9999 THEN POKE AP,0:POKE
AP+1,0:END
62039 REM REMOVE LEADING COLONS AND SPACES
62040 A=A+1:B=PEEK(A):IF (B=32)OR(B=58) THEN 62040
62050 A=A-1:FOR K=1 TO 255:A=A+1:B=PEEK(A)
62060 IF B=0 THEN NEXT I
62065 IF B=142 THEN GOTO 62100
62068 IF (B=128)OR(B=143)OR(B=141) THEN L(AN)=B:AN=AN+1:GOTO 62100
62070 IF B=58 THEN GOTO 62400
62072 REM STORE CHAR. FOR COMPACT LINE
62073 IF B<>32 THEN L(AN)=B:AN=AN+1
62075 IF B=136 THEN GOTO 62200
62080 IF B=34 THEN GOTO 62300
62090 NEXT K:STOP
62100 FOR K=1 TO 255:A=A+1:B=PEEK(A):REM LOOKING FOR LINE END
62110 IF B=0 THEN NEXT I
62120 NEXT K
62200 FOR K=1 TO 255:A=A+1:B=PEEK(A):REM FOUND "GOTO"
62210 IF B=0 THEN NEXT I
62215 IF B=32 THEN A=A+1:B=PEEK(A):GOTO62210
62220 IF B=58 THEN GOTO 62100
62225 L(AN)=B:AN=AN+1:NEXT K
62300 FOR K=1 TO 255:A=A+1:B=PEEK(A):REM FOUND " CHAR.
62320 IF B=34 THEN L(AN)=B:AN=AN+1:GOTO 62090
62325 IF B=0 THEN NEXT I
62327 IF B=58 THEN 62400
62330 L(AN)=B:AN=AN+1:NEXT K
62400 A=A+1:B=PEEK(A):IF (B=32)OR(B=58) THEN 62400:REM FOUND :
62410 IF B=0 THEN NEXT I
62420 IF B=142 THEN GOTO 62100
```

```
62430 L(AN)=58:L(AN+1)=B:AN=AN+2:
GOTO62120
62600 PRINT L;:REM POKE MEMORY WITH
COMPACTED LINE
62601 AH=INT((A-3)/256):AL=(A-3)
-256*AH
62602 POKE AP,AL:POKE AP+1,AH:PRINT
TAB(8) AL;AH;
62603 REM AN IS LENGTH OF COMPACTED
LINE
62604 IF AN=0 THEN PRINT:RETURN
62605 AH=INT(AP/256):AL=AP-256*AH
62607 PRINT TAB(16) AL;AH;
62608 POKE AD,AL:POKE AD+1,AH:AD=AP:
AP=AP+2
62610 AH=INT(L/256):AL=L-256*AH
62611 POKE AP,AL:AP=AP+1:POKE AP,AH:
AP=AP+1
62616 FOR I=0 TO AN-1:POKE AP,L(I):
PRINT CHR$(L(I));:AP=AP+1:NEXT I
62620 POKE AP,0:AP=AP+1:PRINT:RETURN
62700 REM
62705 REM    VARIABLES
62710 REM
62715 REM  L    LINE NUMBER
62720 REM  A    ADDRESS BEING SCANNED
62725 REM  AP   ADDRESS BEING POKED
62730 REM  AD   ADDRESS OF PREVIOUS LINE
62732 REM  AN   INDEX IN L(I) OF CHAR.
TO BE POKED
62735 REM  B    CHARACTER BEING SCANNED
62739 REM
62740 REM    TOKENS AND CHARACTERS
62741 REM
62745 REM  32      SPACE
62750 REM  34      " CHARACTER
62752 REM  58      : CHARACTER
62755 REM 128      END TOKEN
62760 REM 136      GOTO
62765 REM 141      RETURN
62770 REM 142      REM
62775 REM 143      STOP
OK
```


Now if the compacted line is zero length, one returns to the "I" loop and starts compacting the next line. If not, one has to update the pointer in the previous compacted line and this is done in lines 62605 and 7. This restores continuity to the chain of pointers. Then the line number is POKEd in, and then the line of compacted text and finally the null at the end.

While developing this program, I used a somewhat longer "Test Program" than I am giving here. The final test was made on two of my old war horses. The first is a Knight's Tour and the other I informally call "Godzilla Eats Tanks". Both programs are rather long, involved, and use PEEKed and POKEd graphics extensively. "Godzilla" uses PEEKed keyboard input with ANDed and ORed data. ("Godzilla" is invisible and lays down an invisible trail that is sniffed out by a "stench seeking" guided missile.) Both programs ran successfully after compacting to about 75% of their original length. However, I did need to repair the Knight's Tour because I had some GOTO's that went to free standing REM's. These REM's were removed by the Compactor and had to be put back in by hand. I have since learned my lesson. Never GOTO or GOSUB to a REM.

This program is very useful but not yet the ultimate in compaction technique. At least one of the large software houses sells a compactor that is combined with a "branch locator" so that statements which are not the target lines of a GOSUB or GOTO are candidates to be put on the same line, separated by colons. It would certainly be faster and easier to buy such an efficient compactor in preference to tapping this one in through the keyboard, but if your interest is in playing around with logical tasks, you may prefer to modify this program to have multistatement-per-line capability. ©

INTRODUCING COGNIVOX Series VIO-1000 A Revolutionary New Voice Input and Output Peripheral

High Fidelity Voice Response
Industrial Quality Recognition

PET — AIM-65 — APPLE II

COGNIVOX series VIO-1000 is a top-of-the-line voice I/O peripheral for business and educational applications and the demanding hobbyist.

It can be trained to recognize words or short phrases drawn from a vocabulary of 32 entries chosen by the user. It will talk back with up to 32 words or short phrases. In disk based systems, response vocabularies can be stored on the disk and brought to memory as needed, giving an effectively unlimited number of vocabular y entries. The quality of voice response is excellent, and it is far superior to that of speech synthesizers.

COGNIVOX series 1000 comes complete and ready to plug into your computer (the computer must have at least 16K of RAM). It connects to the parallel I/O port of the PET, to the game paddle connector on the Apple and to the J1 port on the AIM-65. Connectors are included as required. Also included are a microphone, cassette with software and extensive user manual. A built-in speaker/amplifier is provided as well as a jack for connecting an external speaker or amplifier.

Software supplied with COGNIVOX includes two voice operated, talking video games, VOTH and VOICETRAP. These games are absolutely captivating to play, and the only voice operated talking games that are commercially available.

Adding voice I/O to your own programs is very simple. A single statement in BASIC is all that is required to say or to recognize a word. Complete instructions on how to do it are provided in the manual.

In keeping with the VOICETEK tradition of high performance at affordable price, we have priced COGNIVOX series 1000 at the unbelievably low, introductory price of **$249** (plus $5 shipping in the US, CA add 6% tax. Foreign orders welcome, add 10% for handling and shipping via AIR MAIL). When ordering, please give the make and model of your computer, the amount of RAM and whether you have disks or not.

In addition to COGNIVOX series VIO-1000, VOICETEK manufactures a complete line of voice I/O peripherals for most of the popular personal computers. Speech recognition-only peripherals are available for the 8K PET and the 4K AIM.

For more information call us at 805-685-1854 or write at the address below.

Dealer Inquiries invited.

VOICETEK Dept A, P.O. Box 388
Goleta, CA 93116

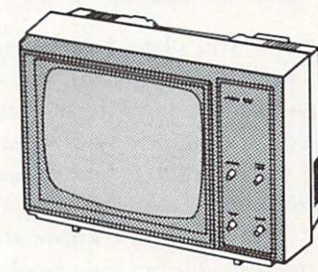
HOBBYWORLD ELECTRONICS, INC.

19511 Business Center Dr. • Dept. G4
Northridge, CA 91324

April Specials

NEW Leedex Monitors

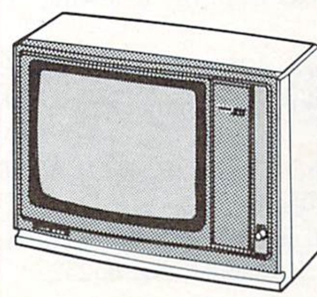
Leedex 100G



- *Green phosphor for easy viewing*
- *Hi-resolution*

Cat No. 2999

$169⁰⁰



13" Color Monitor

- *Hi-Resolution*
- *Full color*
- *Low cost*

Cat No. 3000

$399⁰⁰

Don't Miss Out . . .

. . . on the other terrific buys in the Hobbyworld catalogue. Page after page of computer products for the Atari, Apple, TRS-80 and others. Our catalogue is FREE, so write or phone for your copy NOW!

Toll Free:
(800) 423-5387

Local & Outside USA:
(213) 886-9200

Open Monday thru Friday: 10am-6pm
Saturday: 9am-5pm

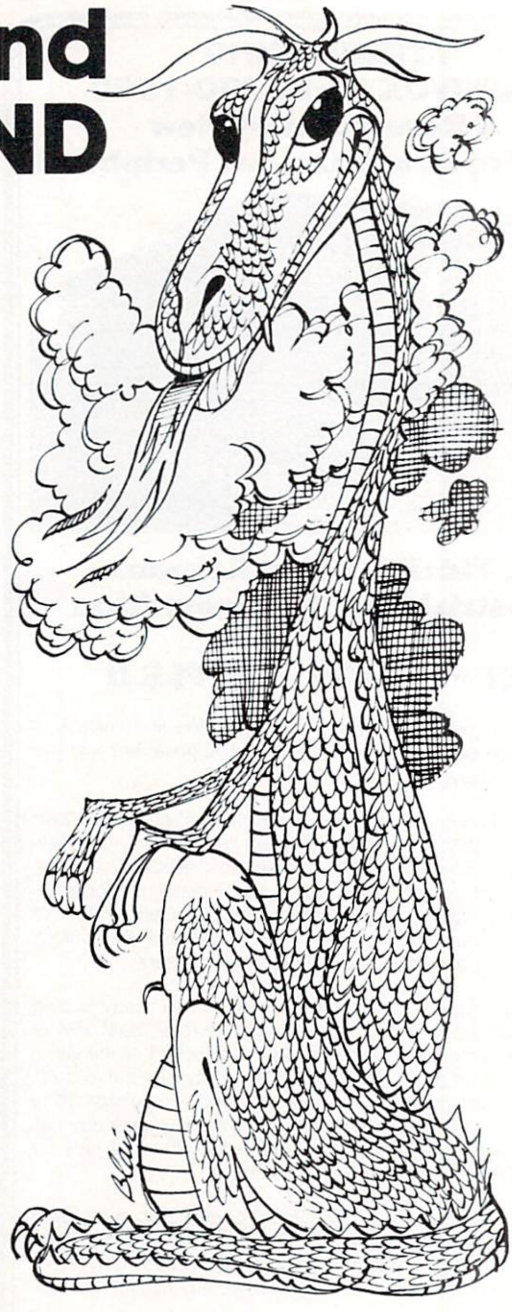
We're changing our name to keep pace with the times! Look for our new name, HW Electronics, in our ads beginning in June, 1981.

The Mysterious And Unpredictable RND

Bob Albrecht and
George Firedrake

From a book of the same name by Dymax Publishing Company; copyright ©1980 Dymax. Permission to reprint by teachers for classroom use is granted.

Editor's Note: You may reach Bob & George by mail at: P.O. Box 310 Menlo Park, CA 94025



Create An Adventure

Have you played *Dungeons and Dragons* or *Runequest* or *Tunnels and Trolls*? These are fantasy adventure games. To play these games, you create a character. You then guide your character through adventures in a world created by a Gamemaster and... sigh... completely controlled by the Gamemaster.

If you don't know about these games, we suggest you get rule books from the following companies.

- *Dungeons and Dragons* (D & D) from TRS Hobbies, P.O. Box 756, Lake Geneva, WI 53147.
- *Runequest* from the Chaosium, P.O. Box 6302, Albany, CA 94706.
- *Tunnels and Trolls* from Flying Buffalo, Inc., P.O. Box 1467, Scottsdale, AZ 85252.

To create a character to play any of these games, you will need to roll three six-sided dice several times. We will demonstrate by rolling a character to

play *Tunnels and Trolls.* Why? Because T & T is the simplest game for beginners to begin with.

An adventurer has six *Prime Attributes:* STRENGTH, INTELLIGENCE, LUCK, CONSTITUTION, DEXTERITY and CHARISMA. The six Prime Attributes determine the character's ability to speak languages, use weapons, overcome monsters, sense things, use magic, and... go successfully adventuring in the Gamemaster's world.

The Prime Attributes are usually abbreviated, as follows.

STRENGTH	STR
INTELLIGENCE	INT or IQ
LUCK	LK
CONSTITUTION	CON
DEXTERITY	DEX
CHARISMA	CHR

In T & T (**Tunnels and Trolls**), these are the Prime Attributes:

STRENGTH is primarily the ability to exert force—lifting, shoving, pushing down, etc. It shows how much junk (measured in weight units, see below) the character can move around. If his strength is ever depleted until it goes to 0, the character is dead.

INTELLIGENCE is the measure of a character's ability to reason clearly, solve problems, remember well, etc. It is also a factor in language ability.

LUCK is the ability to be in the right place at the right time, or to put something else in the right place at the right time. It is useful in avoiding traps, striking luck blows with weapons, and gambling of all sorts.

CONSTITUTION is the general measure of a character's health. It is also the measure of endurance and how much punishment the body can absorb before it dies. Hits taken in combat are subtracted from Constitution. If CON ever goes to 0, the character dies.

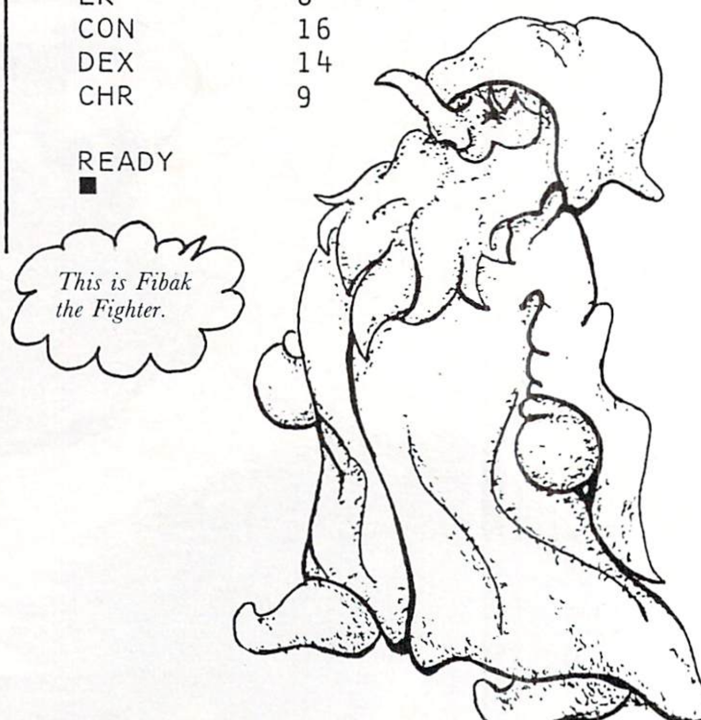
DEXTERITY refers to manual dexterity and general agility.

CHARISMA is the measure of one's personal attractiveness and leadership ability. It is not necessarily synonymous with personal beauty, although there is often a high correlation. Charisma is the only attribute which can fall to 0, or even go negative, without resulting in death. Generally speaking, characters with charismas less than 7 are unappreciated in human society, and anything less than 3 is positively unwelcome. Monstrous characters, when rated with attributes, have negative charismas.

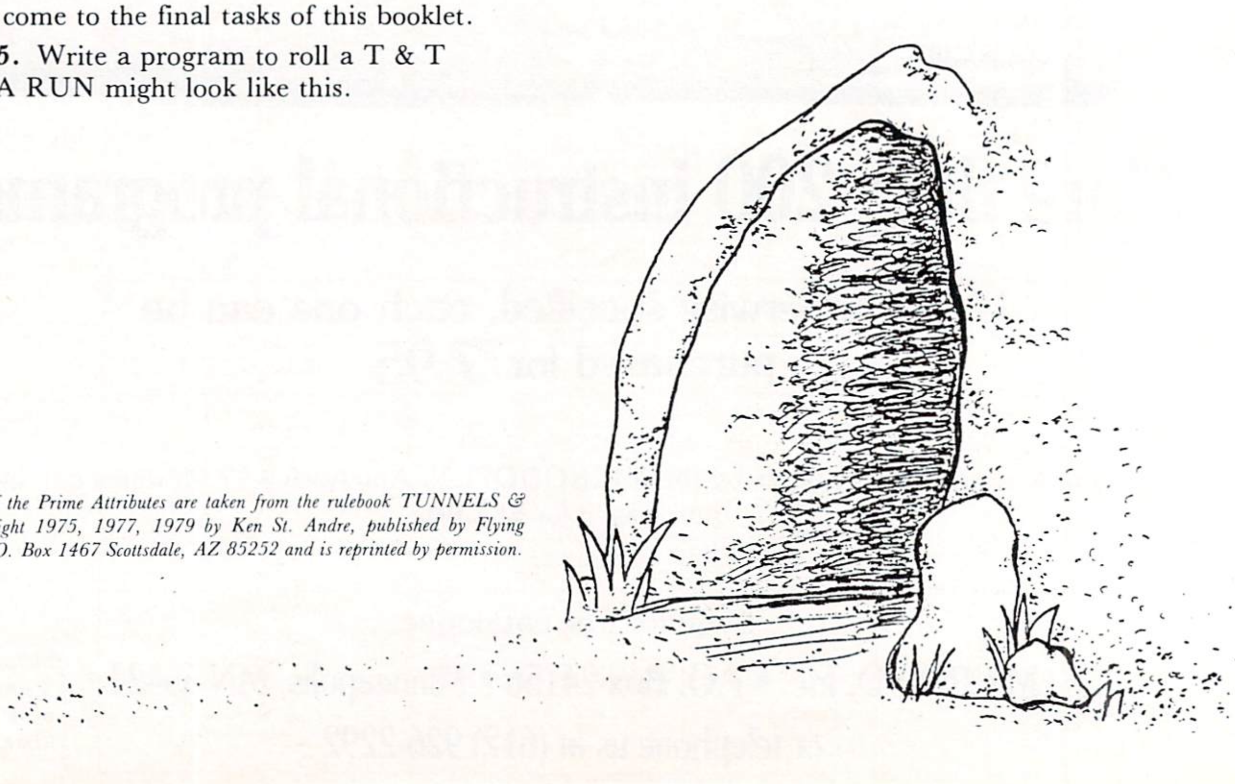
And so we come to the final tasks of this booklet.

Exercise 15. Write a program to roll a T & T character. A RUN might look like this.

```
STR        15
IQ         11
LK         8
CON        16
DEX        14
CHR        9

READY
■
```



Hmmm...obviously a fighter. Remember, when we roll three dice, the possible outcomes are 3 to 18. So, our character is strong (STR = 15), agile (DEX = 14), with a very high ability to sustain damage (CON = 16). He or she is about average in intelligence (IQ = 11), obviously *not* a leader (CHR = 9) and must depend on skill, not luck (LK = 8).

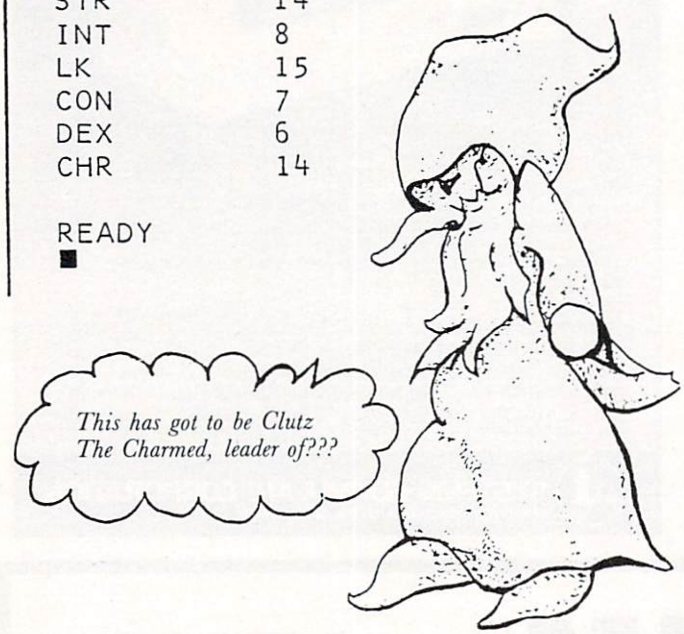


The definitions of the Prime Attributes are taken from the rulebook TUNNELS & TROLLS, copyright 1975, 1977, 1979 by Ken St. Andre, published by Flying Buffalo, Inc., P.O. Box 1467 Scottsdale, AZ 85252 and is reprinted by permission.

In fantasy games, usually a bunch of characters get together to explore the Gamemaster's universe. So, let's roll some more characters to accompany Fibak the Fighter, who awaits in the frontier town of Ziredrac, hoping to collect a company of adventurers to explore the caves under Mt. Skybison, a place of jagged (but sometimes fuzzy) peaks and mysterious dark valleys that sometimes light up with brilliant flashes of wisdom.

We RUN the program again.

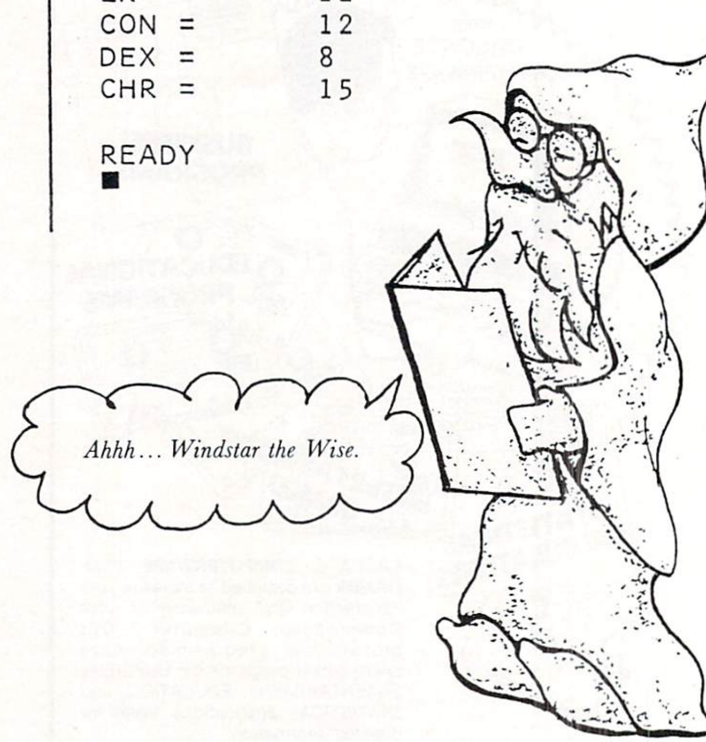
```
STR          14
INT          8
LK           15
CON          7
DEX          6
CHR          14

READY
■
```



This is, indeed, a strange character! Strong (STR = 14), but easily damaged (CON = 7). Clumsy (DEX = 6). But look at luck (LK = 15) and charisma (CHR = 14). This character will convince others to follow her or him into...what? (Again, look at INT = 8.) But, there is that luck.

Help! Let's roll another adventurer.


```
STR =        11
INT =        17
LK  =        12
CON =        12
DEX =        8
CHR =        15

READY
■
```



Saved! Our group is saved! Windstar the Wise wandered by, saw our forlorn little group of adventurers and decided to take charge.

Exercise 16. Describe Windstar as we described Fibak and Clutz. Also describe the way in which Windstar, Fibak and Clutz might work together to explore the Gamemaster's universe, overcome monsters, acquire treasure and...survive!

Exercise 17. (and last of this booklet...) The group of adventurers now numbers three: FIBAK, CLUTZ, WINDSTAR.

Too small a group! They could never survive in the Gamemaster's world. They need at least four more adventurers. More are OK. So, you roll up four more adventurers, then tell who they are and how they relate to and work with our three adventurers.

The Dragons of Pern wish you well in your adventuring.

Do it wisely with luck.

Editor's Note: Next time we come to the end of a series, we'll be printing "Solutions & Stuff"; the author's solutions to the problems raised in this series. Hope you've enjoyed it. ©

Micros With The Handicapped

Susan Semancik and
The Delmarva Computer Club

One of our major objectives at this time is to devise an inexpensive means for providing a listing for a blind programmer. The possibilities we've considered to date are as follows:

a) Have someone continuously available to read the program to the programmer.
b) Buy a speech synthesizer that will vocalize BASIC words, letters, punctuation marks, numbers, graphics characters, and control characters.
c) Build a tactile device that will pulse Braille equivalents of each character encountered in a listing.
d) Sound out the Braille equivalents of each character encountered in a listing by using a different note or sound for each dot in the Braille cell using the CB2 line.

We know that possibility a) works. We have been forced through lack of money and/or equipment to use this means to solve the blind programmer's needs. But, it is certainly not a desirable solution, especially to the programmer. One thing that is soon apparent to anyone working with the handicapped is that the majority have an overwhelming desire for independence and self-reliance. To support this desire, we must look beyond possibility a).

Possibility b) is possible since Commodore has announced the development of a synthesizer for the PET, but we do not yet have access to one. It's price of almost $400 may or may not be a deterrent to its wide-spread use. We'll have to wait and see. Possibility c) is being considered by the Delmarva Computer Club at this time, and hopefully we can give you details on its use and cost in the very near future. The other devices that we know about that are currently on the market are very expensive and/or computer dependent. For example, Maryland Computer Services, Inc., Bel Air, Maryland, has advertised a Hewlett-Packard desktop computer and a talking interface that provides spoken output of 64 ASCII characters for $10,500. Triformation Systems, Inc., Stuart, Florida, has advertised a high-speed braille output on paper tape from computer via built-in acoustic coupler for $2,950. And this past March, ELINFA, Inc., Washington, D.C., announced a portable braille recorder with braille display and computer interface for $4,600.

That brings us to possibility d), which is immediately available to anyone with an amplifier and speaker connected to the PET's CB2 line. Since this is commonly used already to produce sound and music, this would mean no additional expense for most people, and so seems to be a good place to start exploring. It is certainly inexpensive, and an ear plug can be used with the speaker so that only the user can hear the sounds. This would require the blind programmer to learn something new; but, essentially the code is the same; only the method of perceiving the code is new. Rather than feeling the dot's position within the braille cell, the user will listen to determine a note's position within a range of notes.

The program entitled "A Sound Idea for the Blind" is an assembly language program that will run on either an OLD or NEW ROM 8K PET computer. It is designed to teach anyone the Braille equivalents of letters in both a visual and auditory form. After typing RUN, the user can type any letter on the keyboard and will automatically see that letter's braille dot configuration, and hear the notes that correspond to those dots. The user can also hit any number from 1 through 6 to hear singly the notes assigned to each dot position. This program could be used by the visual user to learn Braille or by the non-visual to learn the sounds associated with each position of the Braille cell.

The speed at which the notes are sounded is controlled by the data number in line 240. To speed it up, put a smaller number like 75 in place of the 200. If a change in either the quality of the sound or the pitch is desired, the data numbers in line 310 can be changed. These are listed pairwise for the six dot positions of the cell.

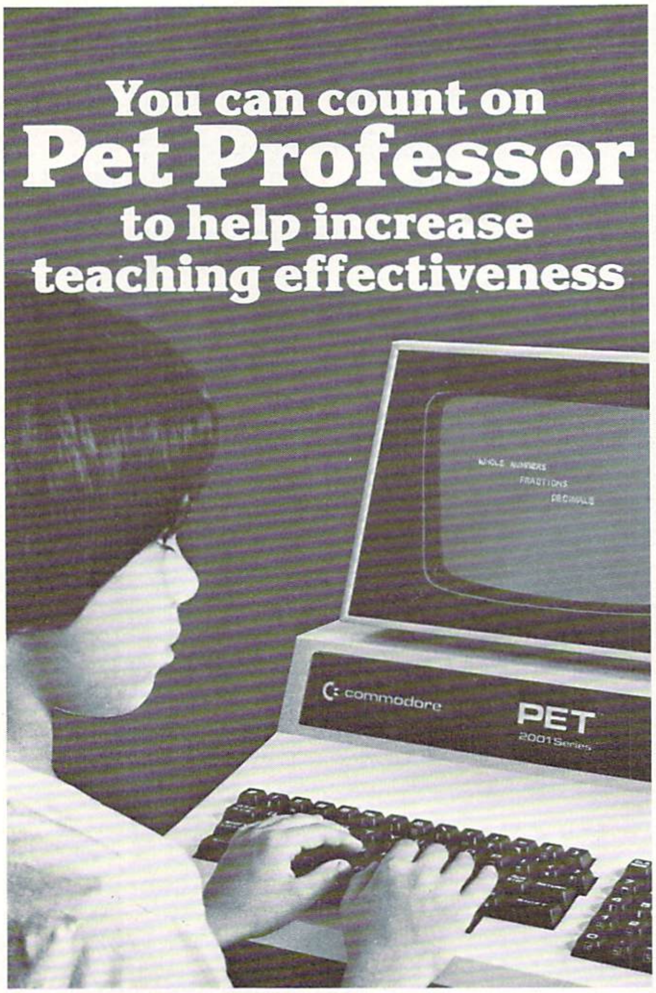
To test how well the sounds have been learned, this program can be modified as in the program entitled "Braille Letter Tester". The Braille codes of the 26 letters of the alphabet are randomly sounded and printed on the screen for the user to identify. The sound can be repeated by pressing the equal sign key instead of a letter. The line

```
455 IF A$<"A" OR A$>"Z" THEN 430
```

must be added so that the program is not terminated with an illegal value when a non-alphabetic character is entered. If the visual user really wants to "see" what the code would sound like to someone blind, the printing of the Braille code on the screen can be eliminated by changing six operation codes to NOP with the addition of the following line:

```
295 POKE 838, 234: POKE 839, 234: POKE 840, 234:
    POKE 883, 234: POKE 884, 234: POKE 885, 234
```

We hope to be able to expand this "sound" concept to realize our full objective of a program's listing for a blind programmer. Let us know if this is a realistic alternative, or if there are other possibilities that we haven't considered.



It reteaches the 4 basic arithmetic operations step-by-step.

PET PROFESSOR includes 77 programs that reteach the four fundamental arithmetic operations for whole numbers, fractions and decimals by providing more than just practice drills. Each program includes a complete tutorial sequence that takes a problem apart, then leads the student step-by-step through solving it.

Each program covers a single objective to allow the teacher to select the skill to be reinforced, It states the objective, supplies vocabulary, displays an example, then leads the student step-by-step through other examples. When the student is ready, a quiz requiring writing is given, then scored by the computer. Every program has been carefully selected. All problems were written by teachers and tested in classrooms..

Use PET PROFESSOR to supplement classroom instruction in individualized or traditional programs. Available for PET 2001 Series. Will run within 8K on any PET.

- Whole numbers—32 programs @ $235.00
- Fractions—24 programs @ $175.00
- Decimals—21 programs @ $175.00

Buy all 3 for $499.00. Save $86.00!

Send $5.00 for more info., sample tape & shipping (deducted from total purchase).

List of other educational programs available on request.

Melad Associates, Inc.
66 Main Street, South River, N.J. 08882 • (201) 238-7330

By Netronics

ASCII/BAUDOT, STAND ALONE
Computer Terminal

COMPLETE FOR ONLY . . . $149.95

ASCII/BAUDOT
Computer Terminal Kit

The Netronics ASCII/BAUDOT Computer Terminal Kit is a microprocessor-controlled, stand alone keyboard/terminal requring no computer memory or software. It allows the use of either a 64 or 32 character by 16 line professional display format with selectable baud rate, RS232-C or 20 ma. output, full cursor control and 75 ohm composite video output.

The keyboard follows the standard typewriter configuration and generates the entire 128 character ASCII upper/lower case set with 96 printable characters. Features include onboard regulators, selectable parity, shift lock key, alpha lock jumper, a drive capability of one TTY load, and the ability to mate directly with almost any computer, including the new Explorer/85 and ELF products by Netronics.

The Computer Terminal requires no I/O mapping and includes 1k of memory, character generator, 2 key rollover, processor controlled cursor control, parallel ASCII/BAUDOT to serial conversion and serial to video processing—fully crystal controlled for superb accuracy. PC boards are the highest quality glass epoxy for the ultimate in reliability and long life.

VIDEO DISPLAY SPECIFICATIONS

The heart of the Netronics Computer Terminal is the microprocessor-controlled Netronics Video Display Board (VID) which allows the terminal to utilize either a parallel ASCII or BAUDOT signal source. The VID converts the parallel data to serial data which is then formatted to either RS232-C or 20 ma. current loop output, which can be connected to the serial I/O on your computer or other interface, i.e., Modem.

When connected to a computer, the computer must echo the character received. This data is received by the VID which processes the information, converting to data to video suitable to be displayed on a TV set (using an RF modulator) or on a video monitor. The VID generates the cursor, horizontal and vertical sync pulses and performs the housekeeping relative to which character and where it is to be displayed on the screen.

Video Output: *1.5 P/P into 75 ohm (EIA RS-170)* • **Baud Rate:** *110 and 300 ASCII* • **Outputs:** *RS232-C or 20 ma. current loop* • **ASCII Character Set:** *128 printable characters—*

!"#$%&'()*+,-./0123456789:;<=>?
@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_
`abcdefghijklmnopqrstuvwxyz{|}~

BAUDOT Character Set: *A B C D E F G H I J K L M N O P Q R S T U V W X Y Z - ? : • 3 $ # ( ) . , 9 0 1 4 ! 5 7 ; 2 / 6 8* • **Cursor Modes:** *Home, Backspace, Horizontal Tab, Line Feed, Vertical Tab, Carriage Return. Two special cursor sequences are provided for absolute and relative X-Y cursor addressing* • **Cursor Control:** *Erase, End of Line, Erase of Screen, Form Feed, Delete* • **Monitor Operation:** *50 or 60Hz (jumper selectable.*

Continental U.S.A. Credit Card Buyers Outside Connecticut
CALL TOLL FREE 800-243-7428
To Order From Connecticut Or For Technical Assistance, Etc. Call (203) 354-9375

Netronics R&D Ltd., Dept.
333 Litchfield Road, New Milford, CT 06776

Please send the items checked below—

- ☐ **Netronics Stand Alone ASCII Keyboard/Computer Terminal Kit, $149.95** plus $3.00 postage & handling.
- ☐ **Deluxe Steel Cabinet** for Netronics Keyboard/Terminal In Blue/Black Finish, **$19.95** plus $2.50 postage and handling.
- ☐ **Video Display Board Kit** alone (less keyboard), **$89.95** plus $3 postage & handling.
- ☐ **12" Video Monitor** (10 MHz bandwidth) fully assembled and tested, **$139.95** plus $5 postage and handling.
- ☐ **RF Modulator Kit** (to use your TV set for a monitor), **$8.95** postpaid.
- ☐ **5 amp Power Supply Kit** In Deluxe Steel Cabinet (±8VDC @ 5 amps, plus 6-8 VAC), **$39.95** plus $2 postage & handling.

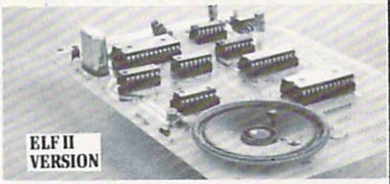
Total Enclosed (Conn. res. add sales tax) $______
By—
☐ Personal Check ☐ Cashiers Check/Money Order
☐ **Visa** ☐ **Master Charge** (Bank # ____)
Acct. # ______
Signature ______ Exp. Date ______
Print Name ______
Address ______
City ______
State ______ Zip ______
☐ **Send Me More Information**

NEW!!!-THE ELECTRIC MOUTH*

for S100, Elf II, Apple, TRS-80 Level II*

From $99.95 kit

Now — teach your computer to talk, dramatically increasing the interaction between you and your machine.



That's right: the ELECTRIC MOUTH actually lets your computer talk! Installed and on-line in just minutes, it's ready for spoken-language use in office, business, industrial and commercial applications, in games, special projects, R&D, education, security devices — there's no end to the ELECTRIC MOUTH's usefulness. Look at these features:

- Supplied with 143 words/letters/phonemes/numbers, capable of producing hundreds of words and phrases.
- Expandable on-board up to thousands of words and phrases (just add additional speech ROMs as they become available).
- Four models, which plug directly into S100, Apple, Elf II and TRS-80 Level II computers.
- Get it to talk by using either Basic or machine language (very easy to use, complete instructions with examples included).
- Uses National Semiconductor's "Digitalker" system.
- Includes on-board audio amplifier and speaker, with provisions for external speakers and amplifier.
- Adds a new dimension and excitement to programming; lets you modify existing programs and games to add spoken announcements of results, warnings, etc.
- Installs in just minutes.

"Elf II" and "The Electric Mouth" are reg. trademarks of Netronics R&D Ltd. "Apple" is a reg. trademark of Apple Computer Inc. "TRS-80 Level II" is a reg. trademark of Tandy Corp.

Principle of Operation The ELECTRIC MOUTH stores words in their digital equivalents in ROMs. When words, phrases, and phonemes are desired, they are simply called for by your program and then synthesized into speech. The ELECTRIC MOUTH system requires none of your valuable memory space except for a few addresses if used in memory mapped mode. In most cases, output ports (user selectable) are used.

Spoken Material Included

ONE	SEVENTY	CONTROL	LEFT	POUND
TWO	EIGHTY	DANGER	LESS	PULSES
THREE	NINETY	DEGREE	LESSER	RATE
FOUR	HUNDRED	DOLLAR	LIMIT	RE
FIVE	THOUSAND	DOWN	LOW	READY
SIX	MILLION	EQUAL	LOWER	RIGHT
SEVEN	ZERO	ERROR	MARK	SS
EIGHT	AGAIN	FEET	METER	SECOND
NINE	AMPERE	FLOW	MILE	SET
TEN	AND	FUEL	MILLI	SPACE
ELEVEN	AT	GALLON	MINUS	SPEED
TWELVE	CANCEL	GO	MINUTE	STAR
THIRTEEN	CASE	GRAM	NEAR	START
FOURTEEN	CENT	GREAT	NUMBER	STOP
FIFTEEN	400HERTZ TONE	GREATER	OF	THAN
SIXTEEN	80HERTZ TONE	HAVE	OFF	THE
SEVENTEEN	20MS SILENCE	HIGH	ON	TIME
EIGHTEEN	40MS SILENCE	HIGHER	OUT	TRY
NINETEEN	80MS SILENCE	HOUR	OVER	UP
TWENTY	160MS SILENCE	IN	PARENTHESIS	VOLT
THIRTY	320 MS SILENCE	INCHES	PERCENT	WEIGHT
FORTY	CENTI	IS	PLEASE	
FIFTY	CHECK	IT	PLUS	
SIXTY	COMMA	KILO	POINT	

Also, the complete alphabet, spoken letter by letter.

Continental U.S.A. Credit Card Buyers Outside Conn. **TO ORDER CALL TOLL FREE 800-243-7428**
To Order From Connecticut Or For Technical Assistance, Etc., **Call (203) 354-9375**

NETRONICS R&D LTD. Dept.
333 Litchfield Road, New Milford, CT 06776

Please send the items checked below:

- ☐ S100 "Electric Mouth" kit $99.95
- ☐ Elf II "Electric Mouth" kit $99.95
- ☐ Apple "Electric Mouth" kit $119.95
- ☐ TRS-80 Level II "Electric Mouth" kit $119.95

Add $20.00 for wired & tested units. All plus $3.00 postage & insurance. Conn. res. add sales tax.

Total Enclosed $______

☐ Personal Check ☐ Cashier's Check/Money Order
☐ Visa ☐ Master Charge (Bank No. ______)
Acct. No. ______
Signature ______ Exp. Date ______
Print Name ______
Address ______
City ______
State ______ Zip ______

```
10 REM *** BRAILLE LETTER TESTER ***
20 REM - YOU WILL HEAR & SEE THE CODE ¬
     ¬FOR A RANDOM LETTER
30 REM - TYPE THE LETTER OR TYPE = TO ¬
     ¬HEAR IT AGAIN
40 REM - ASSEMBLY PROGRAM CODE
50 I=826
60 READ OP:IF OP<>-1 THEN POKE I,OP:
     ¬I=I+1:GOTO 60
100 I=5000
110 READ OP:IF OP<>-1 THEN POKE I,OP:
     ¬I=I+1:GOTO 110
120 REM %%% OLD/NEW CONVERSION %%%
130 ON PEEK(50003) GOSUB 310
140 DATA 169,147,32,210,255
150 DATA 160,0,177,6,170,169,46,157,121,
     ¬129,200,192,6,208,243,96
160 DATA 169,168,133,0,169,19,133,1
170 DATA 169,136,133,6,169,19,133,7
180 DATA 32,58,3,160,0
190 DATA 177,6,160,0,24,106,144,15,133,
     ¬252,177,6,170,169,81
200 DATA 157,121,129,32,129,3,165,252,
     ¬200,192,6,208,232,96,169
210 REM +++ TO CHANGE SPEED OF SOUND,
     ¬ CHANGE DATA LINE 220 +++
220 DATA 200
230 DATA 133,253,152,10,168,177,0,141,
     ¬74,232,200,177,0
240 DATA 141,72,232,136,152,74,168,162,
     ¬255,202,208,253,198,253,208,247
250 DATA 169,0,141,72,232,96,-1
260 DATA 0,80,160,2,82,162,1,3,9,25,17,
     ¬11,27,19,10,26,5,7,13,29,21,15,31,
     ¬23
270 DATA 14,30,37,39,58,45,61,53
280 REM +++ TO CHANGE SOUND, CHANGE ¬
     ¬DATA LINE 290 +++
290 DATA 15,240,200,180,15,120,35,90,
     ¬100,60,15,30,-1
300 GOTO 360
310 REM %%% CONVERSION ROUTINE %%%
320 POKE 834,15:POKE 858,15:POKE 862,16:
     ¬POKE 869,15:POKE 877,171:POKE 879,
     ¬15
330 POKE 890,171:POKE 900,172:POKE 925,
     ¬172
340 RETURN
350 REM - SET SOUND PARAMETERS AND ¬
     ¬CURSOR CONTROL VALUE
360 POKE 59467,16:POKE 59466,51:
     ¬X=RND(-TI):B=224-PEEK(50003)*28
370 L=26:DIM L(26):C=0:FOR I=1 TO 26:
     ¬L(I)=I:NEXT I
380 REM - GET A RANDOM LETTER FROM ¬
     ¬THOSE REMAINING
390 N=INT(L*RND(1))+1:Y=L(N):A=Y+64
400 REM ACTIVATE THE SOUND AND PRINT ¬
     ¬ROUTINE FOR THE CELL AND PRINT A ?
410 POKE 867,Y+5:SYS (847):POKE 33448,63
420 REM - GET RESPONSE FROM THE USER
430 GET A$:IF A$="" THEN 430
440 REM - IF RESPONSE IS =, THEN SOUND ¬
     ¬IT AGAIN
450 IF A$="=" THEN 410
460 REM - INCREASE THE COUNTER FOR ¬
     ¬NUMBER OF TRIES
470 C=C+1
480 REM - PRINT THE USER'S RESPONSE
490 POKE 33450,ASC(A$)-64
500 REM -PRINT THE RIGHT ANSWER AND ¬
     ¬POSITION THE CURSOR FOR COMMENT
510 POKE 33466,Y:POKE B,168:POKE B+1,
     ¬130:POKE B+2,5
520 REM - IF ANSWER IS CORRECT, ¬
     ¬ELIMINATE THE LETTER FROM THE LIST
530 IF A$=CHR$(A) THEN PRINT "CORRECT!":
     ¬X=L(N):L(N)=L(L):L(L)=X:L=L-1:
     ¬GOTO 550
540 PRINT "WRONG."
550 PRINT TAB(4);"ɾHIT ANY KEY TO ¬
     ¬CONTINUE."
560 REM - WAIT FOR USER TO HIT A KEY ¬
     ¬BEFORE CONTINUING
570 GET A$:IF A$="" THEN 570
580 REM - CONTINUE IF THERE ARE ANY ¬
     ¬MORE LETTERS
590 IF L>0 THEN 390
600 REM - TELL THE USER THE NUMBER OF ¬
     ¬TRIES TAKEN TO COMPLETE THE TEST
610 PRINT "ĥYOU COMPLETED THE 26 ¬
     ¬LETTERS IN";C:PRINT "TRIES."
620 REM - RESET PARAMETERS BEFORE ENDING
630 POKE 59467,0:POKE 59466,0:POKE ¬
     ¬59464,12
640 END
READY.
```

```
10 REM *** A SOUND IDEA FOR THE BLIND ¬
     ¬***
20 REM ===  ASSEMBLY PROGRAM CODE ===
30 I=826
40 READ OP:IF OP<>-1 THEN POKE I,OP:
     ¬I=I+1:GOTO 40
50 REM === SCREEN LOCATIONS FOR BRAILLE ¬
     ¬DOT POSITIONS (5000-5005) ===
60 REM === AND BRAILLE CODES (5006-5031)
     ¬ ===
70 REM === AND ALTERNATING TIMBRE AND ¬
     ¬NOTE VALUES (5032-5043)
80 I=5000
90 READ OP:IF OP<>-1 THEN POKE I,OP:
     ¬I=I+1:GOTO 90
100 REM %%% OLD/NEW CONVERSION %%%
110 ON PEEK(50003) GOSUB 330
120 PRINT "ĥ": SYS 847
130 DATA 169,147,32,210,255
140 DATA 160,0,177,6,170,169,46,157,121,
     ¬129,200,192,6,208,243,96
150 DATA 169,168,133,0,169,19,133,1,169,
     ¬16,141,75,232,169,15,141,74,232
160 DATA 169,136,133,6,169,19,133,7
170 DATA 32,228,255,240,251,201,3,208,
     ¬14,169,0,141,75,232,141,74,232,
     ¬169,12
180 DATA 141,76,232,96,133,252,32,58,3,
     ¬165,252,201,65,48,7,201,91,16,3,
     ¬76,172,3
190 DATA 201,49,48,211,201,55,16,207,56,
     ¬233,49,168
200 DATA 177,6,170,169,81,157,121,129,
     ¬32,207,3,76,105,3
210 DATA 56,233,59,168,177,6,160,0,24,
     ¬106,144,15,133,252,177,6,170,169,
     ¬81
220 DATA 157,121,129,32,207,3,165,252,
     ¬200,192,6,208,232,76,105,3,169
```

```
230 REM +++ TO CHANGE SPEED OF SOUND,
     ¬ CHANGE DATA LINE 240 +++
240 DATA 200
250 DATA 133,253,152,10,168,177,0,141,
     ¬74,232,200,177,0
260 DATA 141,72,232,136,152,74,168,162,
     ¬255,202,208,253,198,253,208,247
270 DATA 169,0,141,72,232,96,-1
280 DATA 0,80,160,2,82,162,1,3,9,25,17,
     ¬11,27,19,10,26,5,7,13,29,21,15,31,
     ¬23
300 REM TO CHANGE SOUND, CHANGE DATA ¬
     ¬LINE 310
310 DATA 15,240,200,180,15,120,35,90,
     ¬100,60,15,30,-1
320 END
330 REM %%% CONVERSION ROUTINE %%%
340 POKE 834,15:POKE 868,15:POKE 872,16:
     ¬POKE 897,171:POKE 902,171:
     ¬POKE 927,15
350 POKE 945,15:POKE 953,171:POKE 955,
     ¬15:POKE 966,171:POKE 978,172:
     ¬POKE 1003,172
360 RETURN
```

©

The Delmarva Computer Club
P.O. Box 36
Wallops Island, VA 23337

Matrix Row Operations
A mathematics classroom-teaching aid

William L. Hinrichs
Rockford, Ill.

After teaching the elementary row operations on matrices and their applications on a chalkboard or overhead projector for many years, I have found a better teaching tool. Using a computer and this program, Matrix Row Operations, has eliminated a major stumbling block to learning. Detailed fraction arithmetic always obscured the big picture concepts and techniques. All of the input and output for this program is in the common fraction form used in textbooks and familiar to the students. However, all fraction arithmetic including multiplication, addition, and reducing is done by the computer. This allows us to concentrate on the big picture while avoiding the details during classtime.

Easy Conversion For Apple & TRX-80

This program was written for a Commodore PET computer, but with minor modifications it will run on an Apple, TRS-80, or other machine. No machine language is used and the BASIC used is standard. Due to a lack of standardization of screen control statements, most of the PRINT statements will have to be modified. Loop index variables have been left off some NEXT statements and may have to be added.

Classroom Equipment Set-up

All that is required to use this program with a small group is a PET computer with 8-K RAM and any ROM version. Due to the array size limitation of version 1.0, the program's use is limited but it will run and can be used effectively. For a large group, I use a Petunia video interface board from HUH Electronics to connect our PET to a TV monitor and then connect that monitor to a second monitor with very satisfactory results. This arrangement has worked well with up to 40 students in a class.

Sample Run

Before I describe the program, a sample run is in order. We will solve a system of two equations in two unknowns by matrix row reduction.

Problem:

Find all solutions, if any, of this system of linear equations:

$5x + 3y = 9$ and $-4x + 2y = -16$.

Mathematical Analysis

We will set up a matrix representation of this system of equations and manipulate it by applying the elementary row operations. At each step, we will have a matrix representation of a system of equations with the same solutions as the original one. Our goal is to end up with a matrix representation in row-reduced form, from which the solution to all the systems can be easily read. The original and row-reduced matrices are illustrated in the computer solution below. For more information on the mathematics of matrices, I would recommend College Algebra, Gustafson & Frisk, Brooks/Cole Pub. Co., 1980, pp. 116-175. Most other college algebra texts also have a chapter on matrices.

Computer Demonstration

1. LOAD the program and RUN it.
   The displays which follow are exact copies of the screen. Each display has had all inputs added to the right of the ?, and when the RETURN key is pressed we go on to the next screen. COMMAND? is always the first input requested on each screen. For some commands, additional inputs are requested above the COMMAND? line.
2. The command menu will be displayed and our response to COMMAND? is 'EM' for enter matrix.

```
**************************************
*                                    *
* COMMAND MENU                       *
*                                    *
*    ?  DISPLAY MENU                 *
*   EM  ENTER MATRIX                 *
*   *R  MULTIPLY ROW                 *
*  *R+  MULTIPLY ROW AND ADD         *
*   IR  INTERCHANGE ROWS             *
*  DOM  DISPLAY ORIGINAL MATRIX      *
*  DIM  DISPLAY CURRENT-1 MATRIX     *
*  DCM  DISPLAY CURRENT MATRIX       *
*    B  BACKUP 1 STEP                *
*    Q  QUIT                         *
*                                    *
* COMMAND? EM                        *
*                                    *
**************************************
```

80 COLUMN GRAPHICS

The Integrated Visible Memory for the PET has now been redesigned for the new 12" screen 80 column and forthcoming 40 column PET computers from Commodore. Like earlier MTU units, the new K-1008-43 package mounts inside the PET case for total protection. To make the power and flexibility of the 320 by 200 bit mapped pixel graphics display easily accessible, we have designed the Keyword Graphic Program. This adds 45 graphics commands to Commodore BASIC. If you have been waiting for easy to use, high resolution graphics for your PET, isn't it time you called MTU?

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

Mastercharge and Visa accepted

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

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

```
10 VISMEM: CLEAR
20 P=160: Q=100
30 XP=144: XR=1.5*3.1415927
40 YP=56: YR=1: ZP=64
50 XF=XR/XP: YF=YP/YR: ZF=XR/ZP
60 FOR ZI=-Q TO Q-1
70 IF ZI<-ZP OR ZI>ZP GOTO 150
80 ZT=ZI*XP/ZP: ZZ=ZI
90 XL=INT(.5+SQR(XP*XP-ZT*ZT))
100 FOR XI=-XL TO XL
110 XT=SQR(XI*XI+ZT*ZT)*XF: XX=XI
120 YY=(SIN(XT)+.4*SIN(3*XT))*YF
130 GOSUB 170
140 NEXT XI
150 NEXT ZI
160 STOP
170 X1=XX+ZZ+P
180 Y1=YY-ZZ+Q
190 GMODE 1: MOVE X1,Y1: WRPIX
200 IF Y1=0 GOTO 220
210 GMODE 2: LINE X1,Y1-1,X1,0
220 RETURN
```

MTU
Micro Technology Unlimited
2806 Hillsborough Street
P.O. Box 12106
Raleigh, NC 27605, U.S.A.
(919) 833-1458

NOW 80 COLUMN PETS CAN HAVE MTU HIGH RESOLUTION GRAPHICS

3. We can now enter our matrix row by row. In response to the prompt, we follow each entry by pressing the RETURN key. The number of rows is 2 and the number of columns is 3. We follow these responses with the matrix values by row. The COMMAND? response now is 'DOM' which causes the original matrix to be displayed in row, column form.

```
*************************************
*                                   *
* HOW MANY ROWS? 2                  *
* HOW MANY COLUMNS? 3               *
* M( 1 , 1 ) =? 5                   *
* M( 1 , 2 ) =? 3                   *
* M( 1 , 3 ) =? 9                   *
* M( 2 , 1 ) =? -4                  *
* M( 2 , 2 ) =? 2                   *
* M( 2 , 3 ) =? -16                 *
*                                   *
*                                   *
*                                   *
*                                   *
*                                   *
* COMMAND? DOM                      *
*************************************
```

4. We need a 1 in the first row, first column so our COMMAND? is *R for multiply a row. We are then prompted to enter ROW TO BE MULTIPLIED?, to which we reply 1. Next we enter 1/5 to the prompt MULTIPLIER?. Note that the entry is a common fraction. Following the last entry the screen clears and the new or current matrix is displayed with all fractions reduced.

```
*************************************
*                                   *
*                                   *
*   5     3     9                   *
*                                   *
*                                   *
*                                   *
*  -4     2   -16                   *
*                                   *
*                                   *
*                                   *
*                                   *
*                                   *
* ROW TO BE MULTIPLIED? 1           *
* MULTIPLIER? 1/5                   *
* COMMAND? *R                       *
*                                   *
*************************************
```

5. We now need a 0 in the second row, first column so our COMMAND? is *R + for multiply a row and add to another row. We answer ROW TO BE MULTIPLIED? with a 1, MULTIPLIER? with a 4, and ROW TO BE ADDED TO? with a 2. Those operations are performed and the result is displayed.

```
*************************************
*                                   *
*         3     9                   *
*   1    ---   ---                  *
*         5     5                   *
*                                   *
*                                   *
*  -4     2   -16                   *
*                                   *
*                                   *
*                                   *
* ROW TO BE MULTIPLIED? 1           *
* MULTIPLIER? 4                     *
* ROW TO BE ADDED TO? 2             *
*                                   *
* COMMAND? *R+                      *
*                                   *
*************************************
```

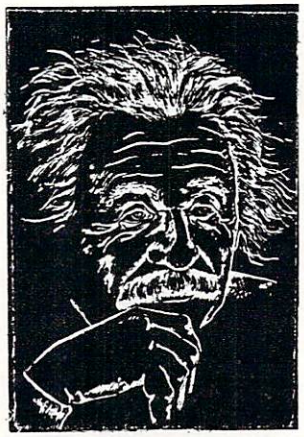
6. We now get a 1 in the second row, second column by responding with *R for COMMAND?, 2 for ROW TO BE MULTIPLIED?, and 5/22 for MULTIPLIER?.

```
*************************************
*                                   *
*         3     9                   *
*   1    ---   ---                  *
*         5     5                   *
*                                   *
*        22   -44                   *
*   0    ---   ---                  *
*         5     5                   *
*                                   *
*                                   *
*                                   *
*                                   *
* ROW TO BE MULTIPLIED? 2           *
* MULTIPLIER? 5/22                  *
* COMMAND? *R                       *
*                                   *
*************************************
```

7. Our last row operation creates a 0 in the first row, second column by responding a COMMAND? with *R +, ROW TO BE MULTIPLIED? by 2, MULTIPLIER? by −3/5, and ROW TO BE ADDED TO? by 1. The final or row-reduced matrix will then be displayed.

```
*************************************
*                                   *
*         3     9                   *
*   1    ---   ---                  *
*         5     5                   *
*                                   *
*                                   *
*   0     1    -2                   *
*                                   *
*                                   *
*                                   *
* ROW TO BE MULTIPLIED? 2           *
* MULTIPLIER? -3/5                  *
* ROW TO BE ADDED TO? 1             *
*                                   *
* COMMAND? *R+                      *
*                                   *
*************************************
```

An Intelligent Alternative



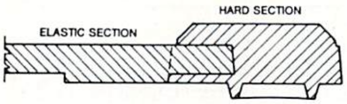
TYPRINTER 221

In the research you are doing before purchasing your computer printer, you are probably confused by the various claims, speeds, choices, shapes and prices. Well, we'd like to clear the air a bit and tell you about the most unusual computer-printer around — the TYPRINTER 221.

You see, it's unusual because it is **totally compatible** with **every** computer and word processing program . . . from the largest to the smallest. It's versatile to the point of incredibility . . . We'll discuss the broad advantages and explain the details.

THE DAISY WHEEL

The special daisy wheel supplied is of a unique design consisting of a 100 character carrying radii. Each radii is formed of two distinct types of plastic — an "elastic plastic" for the stalk of the radii, and a comparatively "hard plastic" used to form the character area. This, combined with a very narrow character profile and a special positioner on each of the 100 radii, guarantees a uniform character density. There is near perfect geometric positioning of the character with no character higher or lower than the others. And because of its unique dual material design, micro-vibrations have virtually been eliminated, leaving your final copy clean, clear and smudge free. The copy produced is comparable to that produced by metal daisy wheels and at a fraction of the cost.



THE KEYBOARD

The keyboard has been referred to as a triumph of human engineering - from the way the keys seem to have been custom designed to fit your fingers, to the way the special feature switches have been grouped. A flip of a switch (or under computer control of course) and the printer becomes a foreign language machine. Push a button, and like magic the printer automatically locates and lines up columns of figures, perfectly balanced between the margins. This incredibly fast, extraordinarily quiet electronic keyboard puts more programming power at you fingertips then printers costing five to ten times as much.

THE DISPLAY

The TYPRINTER 221 presents a new dimension in operator/machine communications. In the manual (typewriter) mode, the printer controls and verifies all entries before printing. The display exhibits the last 15 characters of the text, word-by-word, until the end of the line. The operator may control what will be printed before the actual printing takes place. This new found flexibility enables you to make modifications along the entire line and in both directions. This 20 character plasma display has the ability to scroll backwards as well as forwards; will give the operator a visual indication as to which print mode is currently being selected as well as the number of characters remaining before the right margin is reached. The display will also indicate to the operator:

The number of characters available in the memory.	What characters will be inserted into an existing text;
When the printer is in an error condition.	When the memory for the previous line has been selected.
When a pre-programmed form layout has been selected.	A warning message that the end of the page is being approached.
When the printer is operating from the internal memory.	That a hyphenation decision must be made.

PRINT MODE

The TYPRINTER 221 will allow you to automatically highlight individual characters, words or complete sentences. Whatever is entered from the keyboard or from the computer, even an existing text file, can be printed in one or more of the five different modes:

- traditional printing;
- underlined characters;
- true bold characters where the horizontal component of the character is increased without disturbing the vertical component;
- characters which are both bold and underlined, and;
- a feature unique among computer printers-printing in reverse — white on black, sort of reverse video on paper.

MULTILINGUAL CAPABILITY

A unique and useful feature of the TYPRINTER 221 is its capability of being able to print in several languages without changing the daisy wheel. In addition to English, every standard daisy wheel has the ability and the necessary characters to print in French, Spanish, Italian and German.

THE FEATURES

Automatic justification of the right margin
The electronics of the TYPRINTER 221 have made right hand justification a simple, automatic operation.

Phrase and format storage
Phrases, dates, addresses, data, etc. that may be stored in your computer's memory may be sent over to the printer and stored in one of the "memory bins" of the printer. This information may then be used by the operator in the manual mode. This can save you hours when trying to get a form "just right."

Automatic centering
The TYPRINTER 221 will not only center any title between the pre-set margins, but will also center over one or more columns, or over any specific point and will even align copy with the right margin independent of the left margin.

Automatic vertical lines
A command from the computer enables an automatic feature which prints vertical lines at any point on the paper.

Automatic tab sequence recall
With the TYPRINTER 221 you may store and recall the most frequently needed margin and tab sequences for applications such as daily correspondence, statistical reports, etc. This guarantees consistent high quality appearance of each document.

Paragraph indent
A computer command instantly sets a temporary margin in order to print one or more indented paragraphs with respect to the right margin.

Automatic decimal point location
No matter how many figures to either the left or right of the decimal point, the TYPRINTER 221 will automatically line up the figures with the decimal point in any position you choose. Statistical printing has never been easier.

Column layout
This feature allows you to obtain automatic and perfect distribution of spaces between columns in respect to the margins. A perfect page balance is assured without the need to carry out calculations or additional operations.

There is a wide variety of options that you can add to TYPRINTER 221.

By now you are probably convinced that **we are sold** on our machine, and we hope you can understand why. In fact, why don't you use these facts to measure against any and/or all the other computer printers on the market.

When you do, you will realize the TYPRINTER 221 is an intelligent electronic typewriter, a text formatter — and a brilliant computer printer — available at a suggested list price of only $2850.

TYPRINTER 221 is available at your local computer shop — or we'll tell you where you can see and try one if you call us at

HOWARD INDUSTRIES
2051 E. CERRITOS AVE., 8C
ANAHEIM, CA 92806
714/778-3443

8. We now interpret this row reduced matrix as a system of equations with the same solution as the original system
   x = 3 and y = −2.

```
**************************************
*                                    *
*                                    *
*   1     0     3                    *
*                                    *
*                                    *
*                                    *
*   0     1    -2                    *
*                                    *
*                                    *
*                                    *
*                                    *
*                                    *
*                                    *
*                                    *
* COMMAND? Q                         *
*                                    *
**************************************
```

9. Our last COMMAND? is Q for quit. This allows a clean exit from the program.

In class, we can handle systems of 5, 6, or more equations with no more difficulty than solving two on the chalkboard. The example was kept small to conserve space in the article. Please note that the input concerned strategy decisions involved in the row-reduction method and not with the details of the matrix row operations. Another nice command in the menu is B for backup one step. If we don't get the desired results at any step, B returns us to the status prior to the last command. At any step, we may display the original, current, or current-1 matrix. If an interchange of rows is desired the command IR causes it to happen.

Mathematics Topics Taught

I have found this program useful in teaching the strategies and application of row-reduction to solve systems of linear equations, finding the inverse of any square matrix that has one, and finding the value of a determinant of a square matrix by first zeroing all entries under the diagonal and then finding the product of the diagonal entries. Discussion of the algebra of the simplex method is clarified and simplified when using this program. The simplex method is a matrix technique for the solution of linear programming problems.

Non-Classroom Uses

Students can use this program on a PET in a learning center or library to practice and reinforce their understanding of matrix strategies. I have also found the program valuable for producing step by step handouts to supplement the examples contained in our textbook. I have been aided greatly in this last application by the KEYPRINT program written by Charles Brannon, **COMPUTE!** #7, pp. 84, 86. I LOAD and initialize KEYPRINT first, then whenever I want a copy of the current screen contents I proceed as usual.

Structure Of The Program

The program is a menu driven set of ten subroutines referenced from the main program by one of the menu commands. Their locations and functions are:

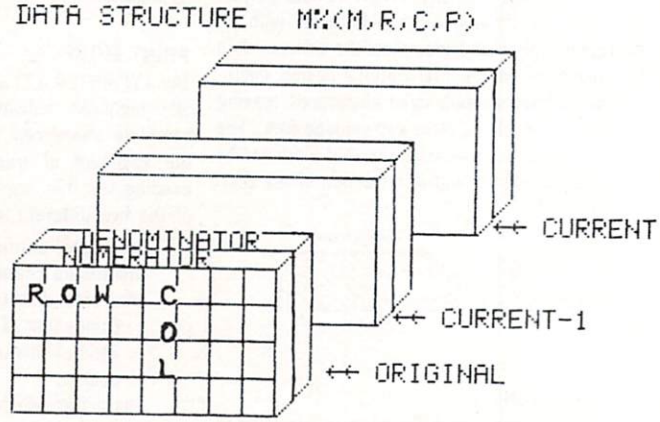
COMMAND	LOCATION	FUNCTION
?	2200-2310	Displays menu.
*R	1900-1996	Multiplies row.
*R +	1700-1880	Multiplies entries of one row and adds to another.
IR	2000-2090	Interchanges any two rows.
B	2400-2430	Backs up 1 step. Current and current-1, matrices are switches.
DOM	1205-1310	Displays original matrix.
DIM	1210-1310	Displays intermediate (current-1) matrix.
DCM	1215-1310	Displays current matrix.
EM	1000-1090	Accepts input of the original matrix.
Q	2100-2120	Clears screen and exits program.

In addition, there are four workhorse subroutines that are called by most of the main subroutines listed above. A fraction reducer routine at 1400-1470 reduces all fractions and also serves to find least common denominators for additions. A matrix switcher at 1500-1560 is the entry routine and the *R and *R + routines. The move current to current-1 routine at 1600-1650 is used by the *R, *R + , IR, and B routines.

DATA STRUCTURE

****

DATA STRUCTURE MX(M,R,C,P)



****

Three matrices are stored as one four-dimensioned array M%(M,R,C,P). M is the matrix number with 1 = original, 2 = current-1, and 3 = current. R is the

row number and C is the column number. P = 1 means numerator and P = 2 means denominator. The denominator for any integer has a value of 1. An integer array name (M%) was chosen to conserve memory so that this program can be effective in an 8K machine. This limits any numerator or denominator to the maximum integer size and occasionally causes overflow problems. This occurs in large matrix problems with many relatively prime factors. It happens seldom but a check has been patched in to detect this and give you the chance to back up one step and try something else.

Summary

This program has three notable weaknesses. It has no provisions for work on matrices with decimal entries and it's use in the classroom does require a minimal amount of time and planning for equipment set-up. It also bombs occasionally as mentioned above under data structure. The strengths of this program far outweight the weaknesses and make it an effective teaching tool. It allows concentration on strategy and concepts rather than details. It handles fractions the way the students are expected to, displaying results and accepting input in a human rather than a computer way. It has saved my students and me valuable classtime and has led them to a better understanding of the ideas being taught. In addition, it can be used outside of class for a variety of applications.

I hope other mathematics teachers will find this program as useful as I have. If you don't wish to spend your time keying the program into your machine, I will be happy to send you a copy on tape. Send a check for $10.00 and your name and complete address to William L. Hinrichs, 5056 Welsh Ct., Rockford, Ill., 61107.

```
5 REM ***MATRIX ROW OPERATIONS****
6 REM *****************************
7 REM **(C)1981 WILLIAM HINRICHS**
8 REM *****************************
9 REM *** ROCK VALLEY COLLEGE ****
10 EX=0
20 PRINT"ĥ↓↓↓↓↓↓↓→→→→→  MATRIX ROW ¬
     ¬OPERATIONS"
30 PRINT"↓→→→→→ CLASSROOM DEMONSTRATION"
40 PRINT"↓→→→→→(C)1980  WILLIAM ¬
     ¬HINRICHS"
50 T=TI
60 IF TI-T<600 THEN 60
70 PRINT"ĥ"
100 GOSUB2200
110 GOSUB900:GOTO110
900 REM*** COMMAND INPUT ***
910 DATA?,*R,*R+,IR,B,DOM,DIM,DCM,EM,Q
920 PRINT"h↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓COMMAND
     ¬";
930 INPUTA$
940 I=1
950 READB$
960 IFA$=B$THEN980
965 I=I+1:IFI>10THENRESTORE:GOTO920
970 GOTO950
980 ONIGOSUB2200,1900,1700,2000,2400,
     ¬1205,1210,1215,1000,2100
990 RESTORE:RETURN
1000 REM*** INPUT MATRIX ***
1005 PRINT"ĥ"
1010 INPUT"HOW MANY ROWS";NR
1020 INPUT"HOW MANY COLUMNS";NC
1030 DIM M%(3,NR,NC,2)
1040 FOR R=1 TO NR
1050 FOR C=1 TO NC
1060 PRINT"M("R","C") =";:INPUTA$
1070 GOSUB1100
1080 M%(1,R,C,1)=N%:M%(1,R,C,2)=D%
1085 M%(2,R,C,1)=N%:M%(2,R,C,2)=D%
1086 M%(3,R,C,1)=N%:M%(3,R,C,2)=D%
1090 NEXT:NEXT:RETURN
1100 REM*** FRACTION SPLITTER ***
1110 L=LEN(A$)
1120 FOR I=1TO L
1130 IFMID$(A$,I,1)="/"THENN%=VAL(LEFT$(
     ¬A$,I-1)):D%=VAL(RIGHT$(A$,L-I)):
     ¬I=L:K=1
1140 NEXT I
1150 IF K<>1THENN%=VAL(A$):D%=1
1160 K=0:RETURN
1200 REM*** MATRIX PRINTER ***
1205 M=1:GOTO1217
1210 M=2:GOTO1217
1215 M=3
1217 PRINT"ĥ"
1218 IFNC>7THENT=4
1219 IFNC<8THENT=5
1220 FOR R=1 TO NR
1230 FOR C=1TONC
1240 N%=M%(M,R,C,1):D%=M%(M,R,C,2)
1246 GOSUB1400
1248 IFD%=-1THENN%=-N%:D%=1
1250 N$=STR$(N%):LN=LEN(N$):D$=STR$(D%):
     ¬LD=LEN(D$)
1260 IFD%=1AND(T-LN)<>0THENPRINT"↓"SPC(T
     ¬-LN)N$"↑";:GOTO1280
1262 IFD%=1AND(T-LN)=0THENPRINT"↓"N$"↑";
     ¬:GOTO1280
1265 IF(T-LN)>0THENPRINTSPC(T-LN);
1270 PRINTN$"←←←↓CCC←←←←←↓";:IF(T-LD)>0T
     ¬HENPRINTSPC(T-LD);
1275 PRINTD$"↑↑";
1280 NEXT C
1290 PRINT"↓↓↓"
1300 NEXT R
1310 RETURN
1400 REM*** FRACTION REDUCER ***
1405 IFN%=0THEND%=1:B=1:RETURN
1410 IF N%>D% THEN A=N%:B=D%:GOTO1440
1420 IF N%<D% THEN A=D%:B=N%:GOTO1440
1430 IF N%=D% THENN%=1:D%=1:B=1:RETURN
1440 Q=INT(A/B):RZ=A-Q*B
1450 IFRZ=0THEN N%=N%/B:D%=D%/B:RETURN
1460 A=B:B=RZ
1470 GOTO 1440
1500 REM*** MATRIX SWITCHER ***
1510 REM SWITCH MATRIX M1 WITH MATRIX M2
1520 FOR R=1 TO NR
1530 FOR C=1 TO NC
1540 FOR P=1 TO 2
1550 T%=M%(M1,R,C,P):M%(M1,R,C,P)=M%(M2,
     ¬R,C,P):M%(M2,R,C,P)=T%
1560 NEXT P,C,R:RETURN
```

```
1600 REM*** MOVE CURRENT MATRIX TO PRIOR
1610 FORR=1TONR
1620 FORC=1TONC
1630 FORP=1TO2
1640 M%(2,R,C,P)=M%(3,R,C,P)
1650 NEXTP,C,R:RETURN
1700 REM*** R2=R1*CONSTANT+R2 ***
1725 PRINT"h↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓"
1730 INPUT"ROW TO BE MULTIPLIED";R1
1732 IFR1<1ORR1>NRTHEN1725
1735 GOSUB1600
1740 INPUT"MULTIPLIER";A$
1750 GOSUB1100
1760 INPUT"ROW TO BE ADDED TO";R2
1770 IFR1<1ORR1>NRORR2<1ORR2>NRORR1=R2TH
     ¬EN1725
1780 GOSUB1400:NC%=N%:DC%=D%
1790 FORC=1TONC
1791 N%=NC%:D%=M%(2,R1,C,2):GOSUB1400
1792 N1%=N%:D2%=D%
1793 N%=M%(2,R1,C,1):D%=DC%:GOSUB1400
1794 N2%=N%:D1%=D%
1795 P=N1%:Q=N2%:IFABS(P*Q)>32767THEN245
     ¬0
1796 P=D1%:Q=D2%:IFABS(P*Q)>32767THEN245
     ¬0
1800 N%=N1%*N2%
1810 D%=D1%*D2%
1815 GOSUB1400
1816 N1%=N%:D1%=D%
1818 N%=D1%:D%=M%(2,R2,C,2)
1819 GOSUB 1400
1820 LC=D1%/B*M%(2,R2,C,2)
1825 N%=LC/D1%*N1%+LC/M%(2,R2,C,2)*M%(2,
     ¬R2,C,1)
1830 D%=LC
1840 GOSUB1400
1850 M%(3,R2,C,1)=N%:M%(3,R2,C,2)=D%
1860 NEXTC
1870 GOSUB1215
1880 RETURN
1900 REM*** MULTIPLY A ROW ***
1910 PRINT"h↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓"
1930 INPUT"ROW TO BE MULTIPLIED";R1
1935 IFR1<1ORR1>NRTHEN1910
1938 GOSUB1600
1940 INPUT"MULTIPLIER";A$
1950 GOSUB1100:NC%=N%:DC%=D%
1960 FOR C=1TONC
1961 N%=NC%:D%=M%(3,R1,C,2):GOSUB1400
1962 N1%=N%:D2%=D%
1963 N%=M%(3,R1,C,1):D%=DC%:GOSUB1400
1964 N2%=N%:D1%=D%
1965 P=N1%:Q=N2%:IFABS(P*Q)>32767THEN245
     ¬0
1966 P=D1%:Q=D2%:IFABS(P*Q)>32767THEN245
     ¬0
1970 N%=N1%*N2%:D%=D1%*D2%:GOSUB1400
1990 M%(3,R1,C,1)=N%:M%(3,R1,C,2)=D%
1995 NEXTC
1996 GOSUB1215:RETURN
2000 REM *** ROW INTERCHANGE ***
2010 PRINT"h↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓"
2020 INPUT"INTERCHANGE WHICH TWO ¬
     ¬ROWS";R1,R2
2030 GOSUB1600
2040 FORC=1TONC
2050 T%=M%(3,R1,C,1):M%(3,R1,C,1)=M%(3,
     ¬R2,C,1):M%(3.R2,C,1)=T%
2060 T%=M%(3,R1,C,2):M%(3,R1,C,2)=M%(3,
     ¬R2,C,2):M%(3,R2,C,2)=T%
2070 NEXTC
2080 EX=EX+1:REM EX IS POWER OF -1 TO ¬
     ¬USE IN FINDING THE DETERMINANT
2085 GOSUB1215
2090 RETURN
2100 REM*** QUIT ***
2110 PRINT"ħ"
2120 END
2200 REM*** DISPLAY MENU ***
2210 PRINT"ħ    COMMAND MENU↓"
2220 PRINT"      ?   DISPLAY MENU
2230 PRINT"↓     EM   ENTER MATRIX
2240 PRINT"↓     *R   MULTIPLY ROW
2250 PRINT"↓    *R+   MULTIPLY ROW AND ¬
     ¬ADD
2260 PRINT"↓     IR   INTERCHANGE ROWS
2270 PRINT"↓    DOM   DISPLAY ORIGINAL ¬
     ¬MATRIX
2280 PRINT"↓    DIM   DISPLAY PRIOR ¬
     ¬MATRIX
2290 PRINT"↓    DCM   DISPLAY CURRENT ¬
     ¬MATRIX
2295 PRINT"↓      B   BACKUP 1 STEP
2300 PRINT"↓      Q   QUIT
2310 RETURN
2400 REM*** BACK UP 1 STEP ***
2410 M1=2:M2=3
2420 GOSUB1500:GOSUB1600:GOSUB1215
2430 RETURN
2449 REM PRODUCT TOO LARGE ERROR MESSAGE
2450 PRINT"ħ↓↓↓↓↓↓↓↓↓↓↓VALUE TOO ¬
     ¬LARGE!!!":PRINT"TO CONTINUE,
     ¬ PRESS RETURN."
2460 GETA$:IFA$<>CHR$(13)THEN2460
2470 RETURN
READY.
```

©

THE STAR MODEM

From Livermore Data Systems

RS232 MODEM	SALE $135
IEEE 488 MODEM	SALE $245
RS232 CCITT	$170
IEEE 488 CCITT	$280

STAR Modem is the price performance leader with a full 2 YEAR FACTORY WARRANTY.

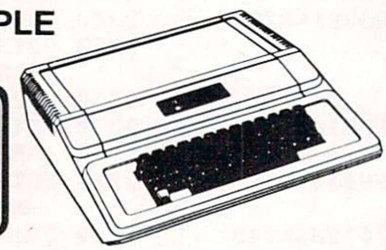
MIPLOT Intelligent Plotter
by Watanabe Instruments (Digiplot)

SPECIAL $1145

Has all intelligent functions for producing graphs and drawings including 8 vector and 4 character commands. Solid and broken lines can be specified. Character generator for alpha, numeric, and symbols. Characters can be rotated in 4 orientations, and can be 16 sizes. Coordinate axes drawn by specifying graduation interval and number of repetitions. Parallel ASCII interface, 11 x 17 paper.

WE CARRY THE BLACK APPLE

BELL & HOWELL
MADE EXCLUSIVELY FOR BELL & HOWELL BY
apple computer inc.



SPECIALS

EPSON MX-80 Printer	**Call**
EPSON MX-70 Printer	**Call**
STARWRITER Daisy Wheel Printer	$1390
Centronics 737 Printer	725
NEC Spinwriter	2500
XYMEC HI-Q 1000 Intelligent Daisy Wheel	2150
Leedex Video 100 12" Monitor	129
ZENITH DATA SYSTEMS	
Z19 Video Terminal (factory assem.)	729
Z89 with 48K (factory assem.)	2150
SYM-1	209
SYM BAS-1 BASIC or RAE-1/2 Assembler	85
KTM-2/80 Synertek Video Board	349
Seawell Motherboard - 4K RAM	195
Seawell 16K Static RAM - KIM, SYM, AIM	320

EBS Business System for PET/CBM $795
Extremely comprehensive package for small business. Fully integrated inventory and accounts receivable system including invoices, packing slips, mail labels, statements, bank deposit slips, and 17 reports.

Tiny Mail List for PET BASIC 3.0 $12
Clever disk based package by L.C.Cargile. Uses overlays to avoid garbage collection delays.

DISK MASTER by Robert Baker $12
Comprehensive diskette cataloging package to keep track of all programs and files on up to 140 diskettes. Specify 2040/4040/8050.

Other CBM Software Packages:
Wordcraft 80 -- Word Processing System
Federal Income Tax Preparation System
CBM Assembler Development System
Comprehensive Investment Analysis Package

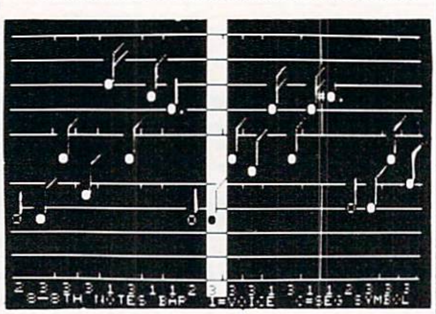
fullFORTH+ for PET/CBM $65
A full-featured FORTH with extensions conforming to Forth Interest Group standards. Includes assembler, string processing capabilities, disk virtual memory, multiple dimensioned arrays, floating point and integer processing.

6502	7.45	10/6.95	50/6.55	100/6.15
6502A	8.40	10/7.95	50/7.35	100/6.90
6520 PIA	5.15	10/4.90	50/4.45	100/4.15
6522 VIA	6.45	10/6.10	50/5.75	100/5.45
6532	7.90	10/7.40	50/7.00	100/6.60
2114-L200 ns RAM		3.75	20/3.50	100/3.25
2716 EPROM (5 volt)		9.90	5/9.45	10/8.90
TMS 2532 EPROM				23.50
4116-200 ns RAM				**8 for 29.00**
S-100 Wire Wrap				2.65

CASSETTES – AGFA PE-611 PREMIUM
High output, low noise, 5 screw housing, labels.

C-10	10/5.65	50/25.00	100/48.00
C-30	10/7.30	50/34.00	100/66.00

All other lengths available. Write for price list.



4 PART HARMONY MUSIC SYSTEM for PET
The Visible Music Monitor, by Frank Levinson, allows you to easily enter, display, edit, and play 4 part harmony music. Includes whole notes thru 64ths (with dotted and triplets), tempo change, key signature, transpose, etc. The KL-4M unit includes D to A converter and amplifier (add your own speaker).
KL-4M Music Board with VMM Program $59.90

commodore

CBM-PET SPECIALS

FREE **Up to $235 FREE merchandise with purchase of one of following CBM-PET items!**

		FREE
8032 32K - 80 column CRT	$1795	235
8050 Dual Disk Drive - 1 megabyte	1795	235
8N Full size graphics keyboard	795	75
16K Business or Graphic Keyboard	995	150
32K Business or Graphic Keyboard	1295	205
2040/4040 Dual Disk Drive - 340K	1295	205
2022 Tractor Feed Printer	795	100
C2N External Cassette Deck	95	12
CBM Voice Synthesizer	395	50
Used CBM/PET Computers		CALL

WRITE FOR SYSTEM PRICES

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

WordPro 3+ 32K, disk, printer	250
WordPro 4+ 8032, disk, printer	385
OZZ Data Base System for CBM 8032	335
VISICALC for PET, ATARI	170
BPI General Ledger, A/P, A/R for PET/CBM	270
EPROM Programmer for PET	**72.00**
Programmers Toolkit - PET ROM Utilities	34.90
2 Meter IEEE to IEEE Cable	43.00
PET Spacemaker Switch	24.90
Dust Cover for PET	6.90
IEEE-Parallel Printer Interface for PET	110.00
IEEE-RS232 Printer Interface for PET	120.00
The PET Revealed	17.00
Library of PET Subroutines	17.00
Source Hookup over 1000 programs/services	**88**

DISK SPECIALS

SCOTCH (3M) 5¼"	10/2.85	50/2.75	100/2.65
SCOTCH (3M) 8"	10/2.90	50/2.80	100/2.70
Verbatim 5¼"	10/2.45	50/2.40	100/2.35
(add 1.00 for 5¼" Verbatim plastic storage box)			
Verbatim 8" Dbl. Dens.	10/3.45	50/3.35	100/3.25
BASF 5¼"	10/2.60	20/2.50	100/2.40
BASF 8"	10/2.65	20/2.55	100/2.45

Maxell Disks in stock - write for prices

Diskette Storage Pages		10 for 3.95
Disk Library Cases	8" - 2.85	5" - 2.15

ATARI 800 $777
All Atari Modules 20% OFF

ATARI EDUCATIONAL PLAN Write for details.

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

ALL BOOK and SOFTWARE PRICES DISCOUNTED

The 8086 Book (Osborne)	14.00
Z8000 Assembly Language Programming	16.90
PET Personal Computer Guide (Osborne)	12.75
PET and the IEEE-488 Bus (Osborne)	13.60
6502 Assembly Language (Osborne)	14.45
Programming the 6502 (Zaks)	10.45
6502 Applications Book (Zaks)	10.45
6502 Software Cookbook (Scelbi)	9.45
CP/M Handbook (w/ MP/M) Zaks	11.85
Practical BASIC Programs (Osborne)	13.60
Some Common BASIC Programs (Osborne)	12.75

115 E. Stump Road
Montgomeryville, PA 18936
215-699-5826 **A B Computers**

WRITE FOR CATALOG.
Add $1.25 per order for shipping. We pay balance of UPS surface charges on all prepaid orders. Prices listed are on cash discount basis. Regular prices slightly higher.

A Floating-Point Binary To BCD Routine

Marvin L. DeJong
Department of Mathematics-Physics
The School of the Ozarks
Pt. Lookout, MO 65726

Introduction

A previous issue of **COMPUTE!** carried a BCD to Floating-Point Binary Routine that can be used to convert a series of decimal digits and a decimal exponent to a binary number in a floating-point format. The purpose of such a routine is to enable the user to perform floating-point arithmetic. The program described in this article performs the reverse operation; that is, it converts a floating-point binary number to a decimal number and a decimal exponent. With these two routines and an Am9511 Arithmetic Processing Unit one can do most of the functions found on scientific calculators. I hope to provide a few simple arithmetic routines in the near future. In the meanwhile, you can amuse yourself by converting numbers to floating-point binary numbers and then back to decimal numbers.

Hindsight

The BCD to floating-point binary routine described previously used a divide-by-ten routine that was part of the main program. With my excellent hindsight I now realize that the divide-by-ten routine should have been written as a *subroutine,* to

Listing 1. A New Divide-by-Ten Routine.

```
$0EBF 20 C5 0E   ONCMOR   JSR DIVTEN    Jump to divide-by-ten subroutine.
 0EC2 B8                  CLV           Force a jump around the routine.
 0EC3 50 51               BVC ARND      The new subroutine is inserted
 0EC5 A9 00      DIVTEN   LDA $00       here. Clear accumulator for use
 0EC7 A0 28               LDY $28       as a register. Do $28 = 40 bit
 0EC9 06 00      BRA      ASL OVFLO     divide. OVFLO will be used as
 0ECB 26 04               ROL LSB       "guard" byte.
 0ECD 26 03               ROL NLSB      Roll one bit at a time into the
 0ECF 26 02               ROL NMSB      accumulator which serves to hold
 0ED1 26 01               ROL MSB       the partial dividend.
 0ED3 2A                  ROL A         Check to see if A is larger than
 0ED4 C9 0A               CMP $0A       the divisor, $0A = 10.
 0ED6 90 05               BCC BRB       No. Decrease the bit counter.
 0ED8 38                  SEC           Yes. Subtract divisor from A.
 0ED9 E9 0A               SBC $0A
 0EDB E6 00               INC OVFLO     Set a bit in the quotient.
 0EDD 88         BRB      DEY           Decrease the bit counter.
 0EDE D0 E9               BNE BRA
 0EE0 C6 05      BRC      DEC BEXP      Division is finished, now normalize.
 0EE2 06 00               ASL OVFLO     For each shift left, decrease the
 0EE4 26 04               ROL LSB       binary exponent.
 0EE6 26 03               ROL NLSB      Rotate the mantissa left until a
 0EE8 26 02               ROL NMSB      one is in the most-significant bit.
 0EEA 26 01               ROL MSB
 0EEC 10 F2               BPL BRC
 0EEE A5 00               LDA OVFLO     If the most-significant bit in the
 0EF0 10 12               BPL BRE       guard byte is one, round up.
 0EF2 38                  SEC           Add one.
 0EF3 A2 04               LDX $04       X is byte counter.
 0EF5 B5 00      BRD      LDA ACC,X     Get the LSB.
 0EF7 69 00               ADC $00       Add the carry.
 0EF9 95 00               STA ACC,X     Result into mantissa.
 0EFB CA                  DEX
 0EFC D0 F7               BNE BRD       Back to complete addition.
 0EFE 90 04               BCC BRE       No carry from MSB so finish.
 0F00 66 01               ROR MSB       A carry, put in bit seven,
 0F02 E6 05               INC BEXP      and increase the binary exponent.
 0F04 A9 00      BRE      LDA $00       Clear the OVFLO position, then
 0F06 85 00               STA OVFLO     get out.
 0F08 60                  RTS
  .                        .
  .                        .            Empty memory locations here.
  .                        .
 0F16 A9 00      ARND     LDA $00       Remainder of BCD-to-floating
  .                        .
  :                        :            point routine is here.
```

Listing 2. Modifications to the BCD-to-Floating-Point Binary Routine.

```
$0E54 18                  CLC           Clear carry for addition.
 0E55 A5 05               LDA BEXP      Get binary exponent.
 0E57 69 20               ADC $20       Add $20 = 32 to place binary
 0E59 85 05               STA BEXP      point properly.
 0E5A EA                  NOP
 0E5B EA                  NOP
$0D53 A0 20      BR7      LDY $20       Y will limit the number of
 0D55 A5 01      BR10     LDA MSB       left shifts to 32.
 0D57 30 0D               BMI BR11      If mantissa has a one in its
 0D59 18                  CLC           most-significant bit, get out.
 0D5A A2 04               LDX $04
 0D5C 36 00      BR9      ROL ACC,X     Shift accumulator left one bit.
 0D5E CA                  DEX
 0D5F D0 FB               BNE BR9
 0D61 C6 05               DEC BEXP      Decrement binary exponent for each
 0D63 88                  DEY           left shift.
 0D64 D0 EF               BNE BR10      No more than $20 = 32 bits shifted.
 0D66 60         BR11     RTS           That's it.
```

be called by *both* the BCD to floating-point binary routine and the binary to decimal routine described here. So my first task was to rewrite the divide-by-ten routine as a subroutine. I also discovered that the divide-by-ten routine described in the previous article did not give sufficient precision. In any case, the divide-by-ten routine was completely revised and appears in Listing 1 in this article. It uses the location $0000, called OVFLO, as a "guard" byte to give the necessary precision. It actually starts at $0EC5, but our listing starts at $0EBF to indiciate a few changes that must be made in the original listing to insert the subroutine.

Some other minor modifications to the program are given in Listing 2. Although the BCD to Floating-Point Binary program will work without these changes, it will work better if you introduce the changes shown in Listing 2. The development of the program described in this article enabled me to find some places to improve the other routine. The modifications are simple and short.

The Conversion Routine

The program to convert a normalized floating-point binary number and its exponent to a BCD number and then output the result is given in Listing 3. A 32-bit binary to BCD conversion subroutine is called by this program and it is found in Listing 5. A flowchart of the entire process is given in Figure 1. The normalized floating-point binary mantissa is operated on by a series of "times ten" or "divide by ten" operations until the binary point is moved from the left of the mantissa to the right of the 32 bit mantissa. In other words, we multiply by ten or divide by ten until the binary exponent is 32. Then the mantissa represents an integer and can be converted to a BCD number using the subroutine in Listing 5. The algorithm for this latter routine is from Peatman's (John B)

Listing 3. A Floating-Point Binary to BCD Routine.

```
$0B00 A5 01     BEGIN    LDA MSB       Test MSB to see if mantissa is zero.
 0B02 D0 0E              BNE BRT       If it is, print a zero and then
 0B04 20 9B 0F           JSR CLDISP    get out. Clear display.
 0B07 A9 30              LDA $30       Get ASCII zero.
 0B09 20 A6 0F           JSR OUTCH     Jump to output subroutine.
 0B0C A9 0D              LDA $0D       Get "carriage return."
 0B0E 20 A6 0F           JSR OUTCH     Output it.
 0B11 60                 RTS           Return to calling routine.
 0B12 A9 00     BRT      LDA $00       Clear OVFLO location.
 0B14 85 00              STA OVFLO
 0B16 A5 05     BRY      LDA BEXP      Is the binary exponent negative?
 0B18 10 0B              BPL BRZ       No.
 0B1A 20 00 0D           JSR TENX      Yes. Multiply by ten until the
 0B1D 20 30 0D           JSR NORM      exponent is not negative.
 0B20 C6 17              DEC DEXP      Decrement decimal exponent.
 0B22 B8                 CLV           Force a jump.
 0B23 50 F1     BVC BRY  Repeat.
 0B25 A5 05     BRZ      LDA BEXP      Compare the binary exponent to
 0B27 C9 20              CMP $20       $20 = 32.
 0B29 F0 48              BEQ BCD       Equal. Convert binary to BCD.
 0B2B 90 08              BCC BRX       Less than.
 0B2D 20 C5 0E           JSR DIVTEN    Greater than. Divide by ten until
 0B30 E6 17              INC DEXP      BEXP is less than 32.
 0B32 B8                 CLV           Force a jump.
 0B33 50 F0              BVC BRZ
 0B35 A9 00              LDA $00       Clear OVFLO
 0B37 85 00              STA OVFLO
 0B39 20 00 0D  BRW      JSR TENX      Multiply by ten.
 0B3C 20 30 0D           JSR NORM      Then normalize.
 0B3F C6 17              DEC DEXP      Decrement decimal exponent.
 0B41 A5 05              LDA BEXP      Test binary exponent.
 0B43 C9 20              CMP $20       Is it 32?
 0B45 F0 2C              BEQ BCD       Yes.
 0B47 90 F0              BCC BRW       It's less than 32 so multiply by 10.
 0B49 20 C5 0E           JSR DIVTEN    It's greater than 32 so divide.
 0B4C E6 17              INC DEXP      Increment decimal exponent.
 0B4E A5 05     BRU      LDA BEXP      Test binary exponent.
 0B50 C9 20              CMP $20       Compare with 32.
 0B52 F0 0F              BEQ BRV       Shift mantissa right until exponent
 0B54 46 01              LSR MSB       is 32.
 0B56 66 02              ROR NMSB
 0B58 66 03              ROR NLSB
 0B5A 66 04              ROR LSB
 0B5C 66 0B              ROR TEMP      Least-significant bit into TEMP.
 0B5E E6 05              INC BEXP      Increment exponent for each shift
 0B60 B8                 CLV           right.
 0B61 50 EB              BVC BRU
 0B63 A5 0B     BRV      LDA TEMP      Test to see if we need to round
 0B65 10 0C              BPL BCD       up. No.
 0B67 38                 SEC           Yes. Add one to mantissa.
 0B68 A2 04              LDX $04
 0B6A B5 00     BRS      LDA ACC,X
 0B6C 69 00              ADC $00
 0B6E 95 00              STA ACC,X
 0B70 CA                 DEX
 0B71 D0 F7              BNE BRS
 0B73 20 67 0D  BCD      JSR CONVD     Jump to 32 bit binary-to-BCD
                                       routine.
 0B76 A0 04     BRM      LDY $04       Rotate BCD accumulator right until
 0B78 A2 04     BRP      LDX $04       non-significant zeros are shifted
 0B7A 18                 CLC           out or DEXP is zero, whichever
 0B7B 76 20     BRQ      ROR BCDN,X    comes first.
 0B7D CA                 DEX
 0B7E 10 FB              BPL BRQ
 0B80 88                 DEY
 0B81 D0 F5              BNE BRP
 0B83 E6 17              INC DEXP      Increment exponent for each shift
 0B85 F0 06              BEQ BRO       right. Get out when DEXP = 0.
```

Microprocessor Based Design (McGraw-Hill).

Of course, each time the binary number is multiplied by ten or divided by ten the decimal exponent is adjusted. Thus, we are left with a BCD number in locations $0020 - $0024 (five locations for ten digits) and a decimal exponent in $0017. The rest of the routine is largely processing required to give a reasonable output format. Since we don't want to print a group of non-significant zeros, the BCD number is rotated right until all the zeros are shifted out *or* the decimal exponent is zero, whichever comes first.

Next the routine starts examining the BCD number from the left and skips any leading zeros. Thus, the first non-zero digit is the first digit printed. Of course, if the number is minus (a non-zero result in location $0007) a minus sign is printed. Next the decimal point is printed, and finally the exponent is printed in the form "E XX." Thus, the format chosen always has the decimal point to the right of the significant digits, 3148159. E-6 for example. If you want scientific notation for non-integer results you can modify the output routine. It's simply a matter of moving the decimal point. The flowchart and the comments should allow you to understand and modify the code.

```
0B87 A5 20                LDA LBCDN      Has a non-zero digit been shifted
0B89 29 0F                AND $0F        into the least-significant place?
0B8B F0 E9                BEQ BRM        No. Shift another digit.
0B8D EA          BRO      NOP            Oops. These NOPs cover an
0B8E EA                   NOP            earlier mistake.
0B8F EA                   NOP
0B90 EA                   NOP
0B91 EA                   NOP
0B92 20 9B 0F             JSR CLDISP     This routine simply clears the
0B95 A5 07                LDA MFLAG      AIM 65 20-character display.
0B97 F0 05                BEQ BRN        If the sign of the number is minus,
0B99 A9 2D                LDA $2D        output a minus sign first.
0B9B 20 A6 0F             JSR OUTCH      ASCII " - " = $2D. Output
                                         character.
0B9E A9 0B       BRN      LDA $0B        Set digit counter to eleven.
0BA0 85 0B                STA TEMP
0BA2 A0 04       BRI      LDY $04        Rotate BCD accumulator left to
0BA4 18          BRH      CLC            output most-significant digits
0BA5 A2 FB                LDX $FB        first. But first bypass zeros.
0BA7 36 25       BRG      ROL BCDN
0BA9 E8                   INX
0BAA D0 FB                BNE BRG
0BAC 26 00                ROL OVFLO      Rotate digit into OVFLO.
0BAE 88                   DEY
0BAF D0 F3                BNE BRH
0BB1 C6 0B                DEC TEMP       Decrement digit counter.
0BB3 A5 00                LDA OVFLO      Is the rotated digit zero?
0BB5 F0 Eb                BEQ BRI        Yes. Rotate again.
0BB7 18          BRX      CLC            Convert digit to ASCII and
0BB8 69 30                ADC $30        output it.
0BBA 20 A6 0F             JSR OUTCH
0BBD A9 00                LDA $00        Clear OVFLO for next digit.
0BBF 85 00                STA OVFLO
0BC1 A0 04                LDY $04        Output the remaining digits.
0BC3 18          BRL      CLC
0BC4 A2          $FB      LDX $FB
0BC6 36 25       BRJ      ROL BCDN,X     Rotate a digit at a time into
0BC8 E8                   INX            OVFLO, then output it. One digit
0BC9 D0 FB                BNE BRJ        is four bits or one nibble.
0BCB 26 00                ROL OVFLO
0BCD 88                   DEY
0BCE D0 F3                BNE BRL
0BD0 A5 00                LDA OVFLO      Get digit.
0BD2 C6 0B                DEC TEMP       Decrement digit counter.
0BD4 D0 E1                BNE BRX
0BD6 A5 17                LDA DEXP       Is the decimal exponent zero?
0BD8 F0 48                BEQ ARND       Yes. No need to output exponent.
0BDA A9 2E                LDA $2E        Get ASCII decimal point.
0BDC 20 A6 0F             JSR OUTCH      Output it.
0BDF A9 45                LDA $45        Get ASCII "E".
0BE1 20 A6 0F             JSR OUTCH
0BE4 A5 17                LDA DEXP       Is the decimal exponent plus?
0BE6 10 0D                BPL THERE      Yes.
0BE8 A9 2D                LDA $2D        No. Output ASCII " - "
0BEA 20 A6 0F             JSR OUTCH
0BED A5 17                LDA DEXP       It's minus, so complement it and
0BEF 49 FF                EOR $FF        add one to form the twos
                                         complement.
0BF1 85 17                STA DEXP
0BF3 E6 17                INC DEXP
0BF5 A9 00       THERE    LDA $00        Clear OVFLO.
0BF7 85 00                STA OVFLO
0BF9 F8                   SED            Convert exponent to BCD.
0BFA A0 08                LDY $08
0BFC 26 17       BR1      ROL DEXP
0BFE A5 00                LDA OVFLO
$0C00 65 00               ADC OVFLO
0C02 85 00                STA OVFLO
0C04 88                   DEY
0C05 D0 F5                BNE BR1
```

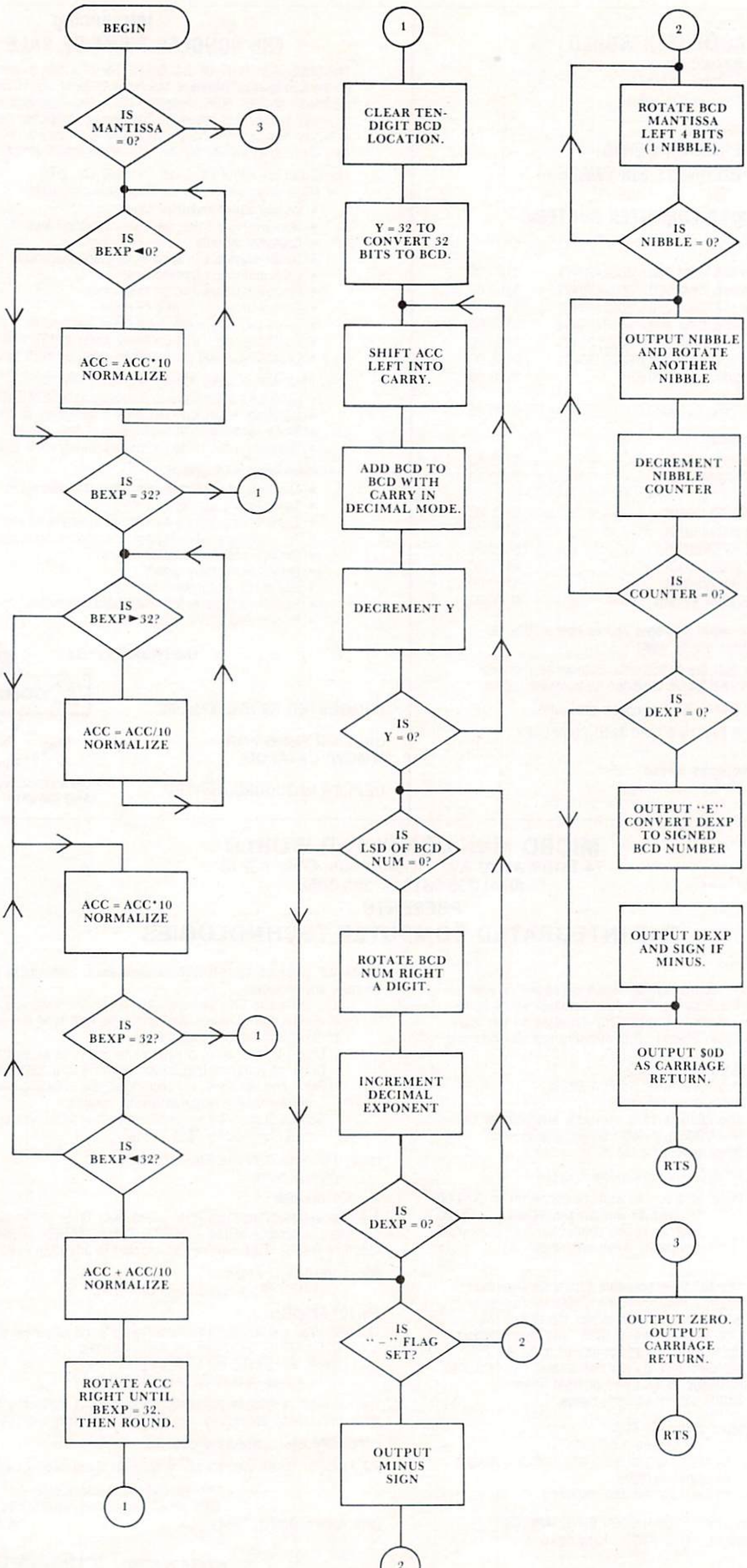


Figure 1. Flowchart of the Floating-Point Binary to BCD Routine.

```
0C07 D8                 CLD
0C08 18                 CLC
0C09 A5 00              LDA OVFLO      Get BCD exponent.
0C0B 29 F0              AND $F0        Mask low-order nibble (digit).
0C0D F0 09              BEQ BR2
0C0F 6A                 ROR A          Rotate nibble to the right.
0C10 6A                 ROR A
0C11 6A                 ROR A
0C12 6A                 ROR A
0C13 69 30              ADC $30        Convert to ASCII.
0C15 20 A6 0F           JSR OUTCH      Output the most-significant digit.
0C18 A5 00     BR2      LDA OVFLO      Get the least-significant digit.
0C1A 29 0F              AND $0F        Mask the high nibble.
0C1C 18                 CLC
0C1D 69 30              ADC $30        Convert to ASCII.
0C1F 20 A6 0F           JSR OUTCH
0C22 A9 0D     ARND     LDA $0D        Get an ASCII carriage return.
0C24 20 A6 0F           JSR OUTCH
0C27 60                 RTS            All finished.
```

Listing 4. Subroutine OUTCH For the AIM 65.

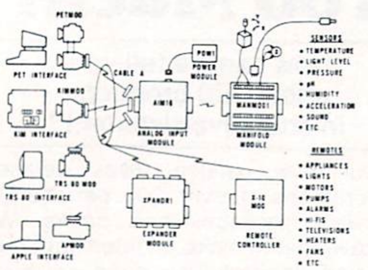
```
$0FA6 20 00 F0  OUTCH   JSR PRINT      AIM 65 monitor subroutine.
 0FA9 20 72 0F          JSR MODIFY     See previous article in COMPUTE!
 0FAC 20 60 0F          JSR DISPLAY    See previous article in COMPUTE!
 0FAF 60 RTS            RTS
```

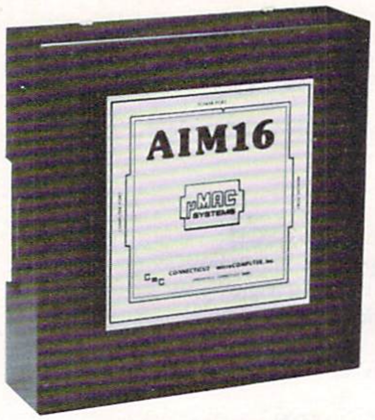
Listing 5. A 32 Bit Binary-to-BCD Subroutine.

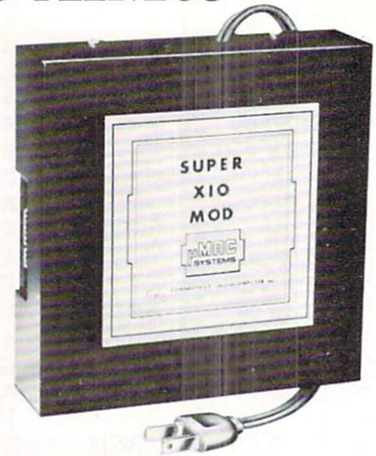
```
$0D67 A2 05    CONVD    LDX $05        Clear BCD accumulator.
 0D69 A9 00             LDA $00
 0D6B 95 20    BRM      STA BCDA,X     Zeros into BCD accumulator.
 0D6D CA                DEX
 0D6E 10 FB             BPL BRM
 0D70 F8                SED            Decimal mode for add.
 0D71 A0 20             LDY $20        Y has number of bits to be
 0D73 06 04    BRN      ASL LSB        converted. Rotate binary number
 0D75 26 03             ROL NLSB       into carry.
 0D77 26 02             ROL NMSB
 0D79 26 01              ROL MSB
 D7B A2 FB              LDX $FB        X will control a five byte
 0D7D B5 25    BRO      LDA BCDA,X     addition. Get least-significant
 0D7F 75 25             ADC BCDA,X     byte of the BCD accumulator,
 0D81 95 25             STA BCDA,X     add it to itself, then store.
 0D83 E8                INX            Repeat until all five bytes have
 0D84 D0 F7             BNE BRO        been added.
 0D86 88                DEY            Get another bit from the binary
 0D87 D0 EA             BNE BRN        number.
 0D89 D8                CLD            Back to binary mode.
 0D8A 60                RTS            And back to the program.
```

©

Microcomputer Measurement And Control For PET,APPLE,KIM and AIM65







The world we live in is full of variables we want to measure. These include weight, temperature, pressure, humidity, speed and fluid level. These variables are continuous and their values may be represented by a voltage. This voltage is the analog of the physical variable. A device which converts a physical, mechanical or chemical quantity to a voltage is called a sensor.

Computers do not understand voltages: They understand bits. Bits are digital signals. A device which converts voltages to bits is an analog-to-digital converter. Our AIM 16 (Analog Input Module) is a 16 input analog-to-digital converter.

The goal of Connecticut microComputer in designing the uMAC SYSTEMS is to produce easy to use, low cost data acquisition and control modules for small computers. These acquisition and control modules will include digital input sensing (e.g. switches), analog input sensing (e.g. temperature, humidity), digital output control (e.g. lamps, motors, alarms), and analog output control (e.g. X-Y plotters, or oscilloscopes).

Connectors

The AIM 16 requires connections to its input port (analog inputs) and its output port (computer interface). The ICON (Input CONnector) is a 20 pin, solder eyelet, edge connector for connecting inputs to each of the AIM16's 16 channels. The OCON (Output CONnector) is a 20 pin, solder eyelet edge connector for connecting the computer's input and output ports to the AIM16.

The MANMOD1 (MANifold MODule) replaces the ICON. It has screw terminals and barrier strips for all 16 inputs for connecting pots, joysticks, voltage sources, etc.

CABLE A24 (24 inch interconnect cable) has an interface connector on one end and an OCON equivalent on the other. This cable provides connections between the uMACSYSTEMS computer interfaces and the AIM 16 or XPANDR1 and between the XPANDR1 and up to eight AIM 16s.

Analog Input Module .

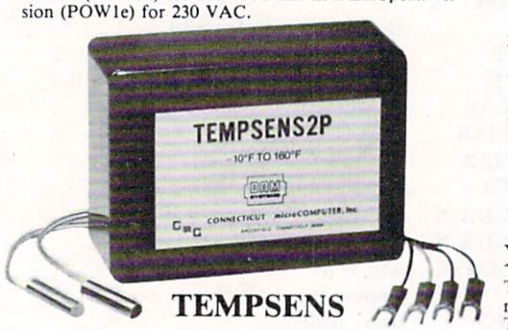
The AIM 16 is a 16 channel analog to digital converter designed to work with most microcomputers. The AIM 16 is connected to the host computer through the computer's 8 bit input port and 8 bit output port, or through one of the uMAC SYSTEMS special interfaces.

The input voltage range is 0 to 5.12 volts. The input voltage is converted to a count between 0 and 255 (00 and FF hex). Resolution is 20 millivolts per count. Accuracy is 0.5% ± 1 bit. Conversion time is less than 100 microseconds per channel. All 16 channels can be scanned in less than 1.5 milliseconds.

Power requirements are 12 volts DC at 60 ma.

POW1

The POW1 is the power module for the AIM16. One POW1 supplies enough power for one AIM16, one MANMOD1, sixteen sensors, one XPANDR1 and one computer interface. The POW1 comes in an American version (POW1a) for 110 VAC and in a European version (POW1e) for 230 VAC.



TEMPSENS

This module provides two temperature probes for use by the AIM16. This module should be used with the MANMOD1 for ease of hookup. The MANMOD1 will support up to 16 probes (eight TEMPSENS modules). Resolution for each probe is 1°F.

Remote Controller-Clock and Calendar

AN INEXPENSIVE CONTROL SOLUTION FOR

HOME SECURITY • ENERGY CONSERVATION • GREENHOUSES • ENVIRONMENTAL CONTROL • INDUSTRIAL CONTROL • LABORATORIES

SUPER X-10 MOD SPECS

1. Remote controller
   Controls up to 256 different remote devices by sending signals over the house wiring to remote modules. Uses BSR remote modules available all over the USA (Sears, Radio Shack, etc.). Does not require BSR control module. Does not use sonic link.
2. Clock/calendar
   Time of day - hours, minutes, seconds
   Date - month, day - automatically corrects for 28,29,30 and 31 day months. Day of the week.
3. Digital input/outputs
   8 inputs - TTL levels or switch closures. Can be used as a trigger for a stored sequence.
   8 outputs - TTL levels
   Power supply included 110VAC only.

XPANDR1

The XPANDR1 allows up to eight Input/Output modules to be connected to a computer at one time. The XPANDR1 is connected to the computer in place of the AIM16 or X10 MOD. Up to eight AIM16s or seven Aim 16s and one X10 MOD are then connected to each of the eight ports provided using a CABLE A24 for each module.

For your convenience the AIM16 and the X10 MOD come as part of a number of sets. The minimum configuration for a usable system is the AIM16 Starter Set 1 which includes one AIM16, one POW1, one ICON and one OCON. The AIM16 Starter Set 2 includes a MANMOD1 in place of the ICON. The minimum configuration for a usable system is the X10 MOD Starter Set which includes one X10 MOD, one ICON and one OCON. These sets require that you have a hardware knowledge of your computer and of computer interfacing.

For simple plug compatible systems we also offer computer interfaces and sets for many computers.

AIM16 .... 179.00
SUPER X10 MOD (110 VAC only) 249.00
POW1a (POWer module-110 VAC) 14.95
POW1e (POWer module-230 VAC) 24.95
ICON (Input CONnector) 9.95
OCON (Output CONnector) 9.95
MANMOD1 (MANifold MODule) 59.95
CABLE A24 (24 inch interconnect cable) 19.95
XPANDR1 (allows up to 8 Input or Output modules to be connected to a computer at one time) 59.95
TEMPSENS2P1 (two temperature probes, -10°F to 160°F) 69.95
LIGHTSENS1P1 (light level probe) 89.95

The following sets include one AIM16, one POW1, one OCON and one ICON.

AIM16 Starter Set 1a (110 VAC) 189.00
AIM16 Starter Set 1e (230 VAC) 199.00

All prices and specifications subject to change without notice. Our 30-day money back guarantee applies.

The following sets include one AIM16, one POW1, one OCON and one MANMOD1.

AIM16 Starter Set 2a (110 VAC) 239.00
AIM16 Starter Set 2e (230 VAC) 249.00

The following modules plug into their respective computers and, when used with a CABLE A24, eliminate the need for custom wiring of the computer interface.

PETMOD (Commodore PET) 49.95
KIMMOD (KIM,SYM) 39.95
APMOD (APPLE II) 59.95
TRS-80 MOD (Radio Shack TRS-80) 59.95
AIM65 MOD (AIM 65) 39.95

The following sets include one AIM16, one POW1, one MANMOD1, one CABLE A24 and one computer interface module

PETSET1a (Commodore PET - 110 VAC) 295.00
PETSET1e (Commodore PET - 230 VAC) 305.00

KIMSET1a (KIM,SYM,AIM65 - 110 VAC) 285.00
KIMSET1e (KIM,SYM,AIM65 - 230 VAC) 295.00
APSET1a(APPLE II - 110 VAC) 295.00
APSET1e(APPLE II - 230 VAC) 305.00
TRS-80 SET1a (Radio Shack TRS-80 - 110 VAC) 295.00
TRS-80 SET1e(Radio Shack TRS-80 - 230 VAC) 305.00
AIM65 SET1a(AIM65-110 VAC) 285.00
AIM65 SET1e(AIM65-230 VAC) 295.00

The following sets include one X10 MOD, one CABLE A24, one ICON and one computer interface module.

PETSET2(Commodore PET) 295.00
KIMSET2(KIM,SYM) 285.00
APSET2(APPLE II) 295.00
TRS-80 SET2 (Radio Shack TRS-80) 295.00
AIM65 SET2 (AIM65) 285.00
SUPER X10 MOD/XPANDR1 SET2 (if you already have a SET1) 295.00

Printer And Communication Interfaces For The CBM/PET

SADI For Serial Two-way Communication, Parallel Printers, and Serial Printers

SADI - The microprocessor based serial and parallel interface for the Commodore PET. SADI allows you to connect your PET to parallel and serial printers, CRT's, modems, acoustic couplers, hard copy terminals and other computers. The serial and parallel ports are independent allowing the PET to communicate with both peripheral devices simultaneously or one at a time. In addition, the RS-232 device can communicate with the parallel device.

Special Features for the PET interface include:
- Conversion to true ASCII both in and out
- Cursor controls and function characters specially printed
- Selectable reversal of upper and lower case
- Addressable - works with other devices

Special Features for the serial interface include:
- Baud rate selectable from 75 to 9600
- Half or full duplex
- 32 character buffer
- X-ON, X-OFF automatically sent
- Selectable carriage return delay

Special Features for the parallel interface include:
- Data strobe - either polarity
- Device ready - either polarity
- Centronics compatible

Complete with power supply, PET IEEE cable, RS-232 connector, parallel port connector and case. Assembled and tested.
SADIa (110VAC) $295
SADIe (230VAC) $325

ADA1600 • For Parallel NEC and Centronics Standard Printers

The ADA1600 is a low cost easy to use interface for the Commodore Computers. It allows the PET and CBM computers to use standard Centronics type printers (including the NEC 5530) for improved quality printing. The ADA1600 has a two foot cable which plugs into the PET IEEE port. Another IEEE card edge connector is provided for connecting disks and other peripherals to the PET. The ADA1600 is addressable and does not tie up the bus. The address is switch selectable. A four foot cable with a standard 36 pin Centronics connector is provided. A switch selects upper/lower case, upper/lower case reversed (needed for some Commodore machines) and upper case only for clearer program listings. Works with WORDPRO, BASIC and other software. No special programming is required. The case measures 3 1/2 x 5 3/4 inches. Comes complete, assembled and tested, with case and cables. Power is obtained from the printer or an external power supply may be used. Retail price for the ADA1600 is $129.

ADA1450 • Serial Printer Adapters

The ADA1450 is a low cost, easy to use serial interface for the Commodore Computers. It allows the PET and CBM computers to use standard serial printers for improved quality printing. The ADA1450 has a two foot cable which plugs into the PET IEEE port. Another IEEE card edge connector is provided for connecting disks and other peripherals to the PET. The ADA1450 is addressable and does not tie up the bus. The address is switch selectable. A six foot RS-232 cable is provided with a DB25 connector. Pin 3 is data out. Pins 5,6 and 8 act as ready lines to the printer. Pins 4 and 20 act as ready lines from the printer. These lines can be switched for non-standard printers. Baud rate is selectable to 9600 baud. A switch selects upper/lower case, upper/lower case reversed (needed for some Commodore machines) and upper case only for clearer program listings. Works with WORDPRO, BASIC and other software. No special programming is required. The case measures 3 1/2 x 5 3/4 inches. Comes complete, assembled and tested, with case, cables, power supply and software on cassette for graphing functions, formatting data etc. The ADA1450 has a female DB25 connector at the end of the RS-232 cable for most standard printers. The ADA1450N has a male DB25 at the end of the RS-232 cable for the DIABLO serial printers. Retail price for the ADA1450 or 1450N is $149.

ADA730 Parallel • For the Centronics 730 and 737 Printers

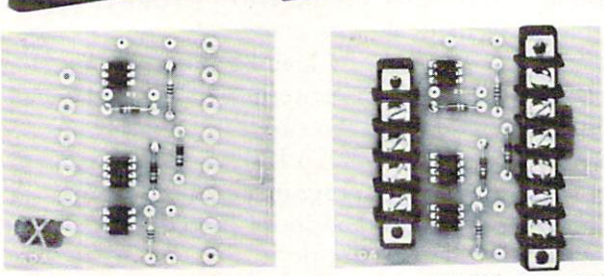
The ADA730 is a low cost easy to use interface for the Commodore Computers. It allows the PET and CBM computers to use Centronics type 730 and 737 printers. The ADA730 has a two foot cable which plugs into the PET IEEE port. Another IEEE card edge connector is provided for connecting disks and other peripherals to the PET. The ADA730 is addressable and does not tie up the bus. The address is switch selectable. A cable with a 36 pin card edge connector is provided. A switch selects upper/lower coase, upper/lower case reversed (needed for some Commodore machines) and upper case only for clearer program listings. Works with WORDPRO, BASIC and other software. No special programming is required. The case measures 3 1/2 x 5 3/4 inches. Comes complete, assembled and tested, with case and cables. Power is obtained from the printer or an external power supply may be used. Retail price for the ADA is $129.

Word Processor Program •

PET Word Processor. On tape - $39.50, On disk - 49.50

For 8K Pets 29.50
For 16K and 32K Pets 39.50

Compose and print letters, flyers, ads, manuscripts, etc. Uses disk or tape. 30 page manual included.



ADA400 RS-232 To Current Loop Adapter

RS-232 to current loop adapter (ADA 400). $29.50
Two circuits - 1 each direction. Run an RS-232 device off a computer's teletype port or vice versa. Optoisolated.

ADA4005 - Solder Pads 24.50
ADA400B - Barrier Strips with screw terminals 29.50

QUANTITY	DESCRIPTION	PRICE	TOTAL

SUBTOTAL
Handling and shipping — add per order $4.00
Foreign orders add 10% for AIR postage
Conn. residents add 7% sales tax
TOTAL ENCLOSED
NAME
COMPANY
ADDRESS
CITY
STATE ZIP
VISA O M/C O Expiration date
Card number
Mention this magazine and deduct 2% from the TOTAL

**In the US order from: Connecticut MicroComputer, Inc.
34 Del Mar Drive Brookfield, CT 06804 (203) 775-4595
In Canada order from: Batteries Included, LTD
71 McCaul St. F6 Toronto, Canada M5T2X1 (416)596-1405**

Connecticut microComputer, Inc.
34 Del Mar Drive, Brookfield, CT 06804
203 775-4595 TWX: 710 456-0052

THE apple® GAZETTE

Resolving Applesoft and Hires Graphics Memory Conflicts

Jeff Schmoyer

This article will attempt to divulge solutions to memory usage conflicts that can occur when an Applesoft program becomes large enough to start taking up residence in a Hires screen page. Of course the problems only appear when a program utilizing Hires graphics is executed. Throughout this article, numbers will be used in both decimal and hexadecimal (base 16). Hexadecimal numbers are represented with a dollar sign ($) preceding them, i.e. $800.

First, it is necessary to understand the memory layout of the region of RAM with which we are concerned. Applesoft programs may reside anywhere in memory from $800 to $BFFF in a 48K Apple II. If a disk is being used, the top boundary will be lower, generally $9600. The top boundary is not really important to this duscussion so it will be referred to as the top of memory. It makes no difference where it is.

The two Hires screens are in fixed positions in memory, the first located from $2000 to $3FFF, and the second from $4000 to $5FFF. Figure 1 is a map demonstrating what is known so far.

As may be seen in the drawing, if an Applesoft program is confined to the area from $800 to $1FFF, there is no conflict. This allows 6K of program space.

Now it's time to introduce another variable, variables! Not only does the program itself take up space, but as variables are allocated in the program, they too have to exist in memory.

String variables are no problem. They allocate space from the top of memory down. Plenty of unused space is available in this region.

Simple variables and numeric and string arrays on the other hand, start at the end of the program

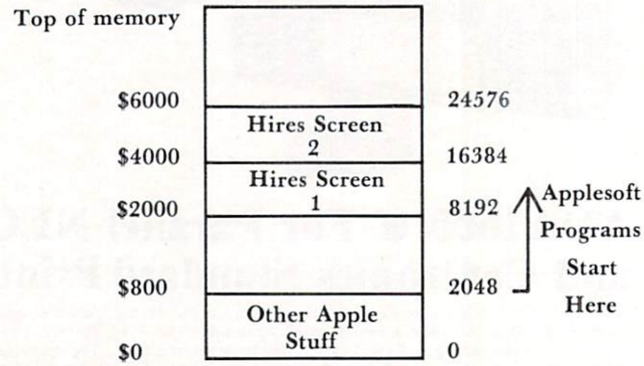


Figure 1: Partial Memory Map

and move up through memory. If there are enough of them, they will cruise right into the Hires page. If this happens, whenever the Hires screen is altered by an HGR or some other command, the variables in the screen space will be changed or erased!

At this point it can be seen that there is 6K of memory available for the program and simple variables and arrays altogether. Now the space is starting to get tight.

The first solution that comes to mind is simply to switch screens and use the second graphics page instead of Hires page 1. That will free up an additional 8K of memory yielding 4K total.

This is not really a bad way to go except that some Hires features are not fully supported for the second screen, such as the mixed text and graphics mode. For Hires page 2, the four lines of text at the bottom are always filled with garbage. (The lines are not actually full of garbage but that will have to be considered in a future article.) There is also the possibility of needing both Hires screens in the program for animation or some other purpose. So this solution may not be totally acceptable.

One acceptable solution deals solely with the variables. If it can be determined that the program itself does not infringe on the Hires territory, the variables may be dealt with separately. This determination may be made by loading the questionable program, entering HGR, entering TEXT, and then listing the program. (Make sure the program has been saved somewhere first.) If the end of the program is still intact, the program fits in the room available. If it is gone then move directly to the next solution. This one will not do it.

If the program fits and the variables do not, the variables may be easily moved to another region of memory. To do this, as the first line of the Applesoft program enter 0 LOMEN:16384. This line must be executed before any variables are allocated. In this

way variables are stored above Hires page 1 and out of the way. If both screens usage are required, 0 LOMEN:24576 may be alternately entered to start variable allocation above screen 2. Be sure to check that the additional program line did not extend the program past the Hires boundary.

If the program itself is too large for the available space, it must be moved to a more roomy area of memory, in this case above the Hires pages. There are two page zero locations which control where an Applesoft program starts, $67 and $68. By altering these locations and reloading the program, it can be run from the new location. These alterations may be made from the direct mode or by a startup program. As long as Applesoft is not reinitialized the changes will remain in effect.

In reality only location $68 in page zero need be changed since $67 will be set to 1 in any case. The necessary poke is POKE 104,96. 104 translates to location $68, while the 96 in hex is $60. This is the high order byte of the new program starting address, $6001. Alternatively POKE 104,64 may be used to locate the program at $4001. This operation has set up the new address for the program to be loaded and run from. For the programs to execute correctly at the new location, one other poke is necessary. The location preceding the program must be set to zero. This would be $6000 for the program at $6001 or $4000 for the program at $4001. POKE 24576,0 or POKE 16384,0 respectively will accommodate this change for programs at $6001 and $4001.

After the program is moved, a 'dead zone' is left in memory from $800 to $1FFF. Neither the program nor any of its variables will use this space. Its best use would be for machine language routines and tables.

To reiterate, for a program to load and run above Hires page 1, POKE 104,64: POKE 16384,0 is necessary. For a program to load and run above Hires page 2, POKE 104,96: POKE 24576,0 is necessary.

Remember, the program will not actually be moved by this operation. Only programs loaded and run after this point will be above the Hires screen. Also, reinitialization of Applesoft will reset the pointers to $800. Setting LOMEM as with the previous technique is not necessary and should not be done.

To recap, three techniques to avoid memory conflicts with Applesoft and the Hires screens were outlined. The first is to use Hires page 2 instead of Hires page 1. The second is to move the simple variables and arrays out of the way with the LOMEM command. The third is to change the program start pointers to reset the program load and run point above the Hires pages. There are other ways to accomodate the screens but these few should suffice in most cases. ©

Fill The Screen With Your Message: Generating Large Multi-Colored Characters Using Apple Low-Resolution Graphics

Francis A. Harvey
Rosann W. Collins
Theodore C. Hines
School of Education
University of North Carolina at Greensboro
Greensboro, North Carolina 27412

Programs written by beginning programmers can often be distinguished from more elaborate "commercial" programs by the fact that the commercial programs make such extensive use of color and graphics. Computers such as the Apple and Atari have very good graphics capability, but many users lack the time or programming background, or both, to take full advantage of these capabilities. As a result their programs, while they may be carefully designed and interesting, lack the pizzazz that children expect from computers as a result of their experiences with commercial programs and computer games at home and in game rooms.

As part of a series of utilities of this kind, we have developed a set of subroutines in Applesoft which will display the characters in any string on the screen as large, colorful letters. With these subroutines program instructions, prompts, positive reinforcement, and negative responses to user input can look just like those in "real" computer games. Very little modification of an existing program is required to convert screen output to this form.

Each character is defined (with a combination of PLOT, VLIN, and HLIN commands) on a matrix which uses seven blocks in the vertical dimension and which varies in width depending on the shape of the character. With the character set defined in this way, each line of text can have between six and nine characters, and a total of four lines of text can be displayed. Each letter is approximately one-fifth as

```
10  REM   ---- LETTER  MATCH  ---
20  REM   ---- 10/20/80 VERSION--
30  REM --------  BY   --------
40  REM ---- FRANCIS A. HARVEY -
50  REM ---- ROSANN W. COLLINS -
60  REM -- & THEODORE C. HINES -
65  REM --- COPYRIGHT 1980 -----
70  HOME : GR : GOSUB 5020
80  REM -------TITLE PAGE-------
90 Y = 3:A$ = "MATCH": GOSUB 6010
100 Y = Y + 12:A$ = "THE": GOSUB 6010
110 Y = Y + 12:A$ = "LETTERS": GOSUB 6010
120  PRINT : PRINT : PRINT
130  FOR I = 1 TO 1000: NEXT I
141  PRINT "BY          FRANCIS A. HARVEY"
142  PRINT "            ROSANN W. COLLINS"
143  PRINT "            THEODORE C. HINES"
145  FOR I = 1 TO 2500: NEXT I
146  PRINT : PRINT : PRINT
160 A$ = "COPYRIGHT OCTOBER 1980": GOSUB 4020
170  FOR I = 1 TO 4000: NEXT
180  PRINT : PRINT : PRINT : REM --CLEAR TEXT
190 E = 5: REM --- FOR DEMO PURPOSES
200  REM :E IS NUMBER OF ATTEMPTS
210  REM --- USER INSTRUCTIONS---
220  GR : GOSUB 5020
230 X = 0:Y = 0: REM --RESETS LETTER POSITION
240 A$ = "I TYPE": GOSUB 6010
250 Y = Y + 12
260 A$ = "A": GOSUB 6010
270 Y = Y + 12:A$ = "LETTER.": GOSUB 6010
280  FOR I = 1 TO 5000: NEXT I: GR
290  GOSUB 5020
300 Y = 3
310 A$ = "YOU TYPE": GOSUB 6010
320 Y = Y + 9
330 A$ = "THE SAME"
340  GOSUB 6010
345 Y = Y + 9:A$ = "LETTER.": GOSUB 6000
350  FOR I = 1 TO 4000: NEXT I
360 Y = Y + 10
370 A$ = "READY?": GOSUB 6000
380  HOME
390  INPUT "STRIKE 'RETURN' WHEN READY.";A$
400  REM
410  REM *********************
420  REM ---BEGIN MAIN PROGRAM---
430 L =  RND (1) * 26 + 1
440 C1 = C1 + 1: REM --COUNTS LETTERS TRIED
450 L =  INT (L) + 64
460 Y = 3
470 A$ =  CHR$ (L): GR : GOSUB 5020: GOSUB 6010
480  HOME
490  PRINT "TYPE THE SAME LETTER."
500  GET B$
510  REM ---DISABLE RETURN KEY
520  IF  ASC (B$) = 13 THEN 500
530  REM ---DISABLE SPACE BAR
540  IF  ASC (B$) = 32 THEN 500
550 Y = Y + 8
560 A$ = B$: GOSUB 6010
570  FOR K = 1 TO 500: NEXT
580  IF B$ =  CHR$ (L) THEN  GOSUB 2010
590  IF B$ <  >  CHR$ (L) THEN  GOSUB 1010:Y = 3: GOTO 500
600  FOR I = 1 TO 2000: NEXT I
610  IF C1 < E THEN 420
620  GR : GOSUB 5000:Y = 3
630  HOME
640 A$ = "THAT'S": GOSUB 6010
650 Y = Y + 12
660 A$ = "ALL": GOSUB 6010
670 Y = Y + 12
680 A$ = "FOR NOW.": GOSUB 6010
```

high as the screen and about one-eighth of a screen wide.

Color can be set within the program or randomly selected each time a line of characters is displayed. The upper-left corner of the matrix is defined as (X,Y), and each character is "drawn" from this reference point.

Each line of characters is passed to the subroutine as the string A$. An initial value of X, the horizontal beginning point of each character, is calculated which will center the characters on the line, and the characters are drawn one at a time.

The subroutines that draw each character automatically increment the value of X the appropriate number of spaces to the right. Messages longer than one line (e.g., "You are sharp!") can be subdivided; the value of the string A$ is set to the contents of each line, and Y is incremented by at least nine before calling the subroutine which centers and plots the characters.

The sample program listed demonstrates two ways in which these techniques can be used. The program, LETTERMATCH, was developed to familiarize primary school students with the letters of the alphabet and the computer keyboard. A randomly selected letter is displayed on the screen, and the user is asked to type the same letter. The GET command is used for input and all non-letter keys, the RETURN key, and space bar are disabled.

If the student enters the wrong letter, the computer responds with a large "TRY IT AGAIN," then clears the screen of the student's response and redisplays the original letter. The student responds until the correct letter is selected. When the student does enter the correct letter, the computer responds (again, in large, multi-colored letters) with one of five randomly selected positive responses, such as "RIGHT!" or "YOU ARE SHARP!". Each student is asked to identify five letters correctly.

```
690  FOR I = 1 TO 2000: NEXT I
700  GR : GOSUB 5000: REM —CLEARS SCREEN
710 Y = 3
720 A$ = "NEXT": GOSUB 6000
730 Y = Y + 12:A$ = "PERSON,": GOSUB 6000
740 Y = Y + 12:A$ = "PLEASE.": GOSUB 6000
745  FOR I = 1 TO 2000: NEXT I
750  PRINT
760  PRINT "TYPE ";: FLASH : PRINT "S";: NORMAL : PRINT " TO STOP."
770  PRINT "STRIKE ANY KEY TO GO ON."
780  GET Z$: IF Z$ <  > "S" THEN C1 = 0: GR : GOTO 380
790  GR : GOSUB 5000:Y = 3
800  HOME
810 A$ = "OK!": GOSUB 6000
820 Y = Y + 12:A$ = "GOODBYE"
830  GOSUB 6000
840 Y = Y + 12:A$ = "FOR NOW.": GOSUB 6000
899  END
1000  REM
1010  REM ——SUBROUTINE FOR WRONG ANSWERS
1020 Y = Y + 8
1030 A$ = "TRY IT": GOSUB 6010
1040 Y = Y + 8:A$ = "AGAIN.": GOSUB 6010
1050  FOR I = 1 TO 1500: NEXT I
1060 Y = 11: GOSUB 3000
1070  RETURN
2000  REM
2010  REM ——SUBROUTINE FOR RIGHT ANSWERS
2020 Y = Y + 12
2030 M =  INT ( RND (1) * 5) + 1
2040  ON M GOTO 2050,2060,2070,2090,2100
2050 A$ = "RIGHT!": GOSUB 6000: RETURN
2060 A$ = "OK!2:GOSUB 6010: RETURN
2070 A$ = "YOU ARE": GOSUB 6010
2080 Y = Y + 8:A$ = "SHARP!": GOSUB 6010: RETURN
2090 A$ = "GREAT!2:GOSUB 6010: RETURN
2100 A$ = "SUPER!": GOSUB 6010: RETURN
2110  RETURN
3000  REM —BLANKS REST OF SCREEN
3010  COLOR= 0
3020  FOR T = Y TO 39
3030  HLIN 0,39 AT T
3040  NEXT
3050  GOSUB 5020
3060  RETURN
4000  REM
4010  REM **********************
4020  REM ——CENTERS AND PRINTS
4030  REM ——REGULAR TEXT——
4040 Z = (40 -  LEN (A$)) / 2
4050  HTAB Z: PRINT A$
4060  RETURN
5000  REM
5010  REM **********************
5020  REM —PICKS RANDOM COLOR——
5030  COLOR=  INT ( RND (1) * 15) + 1
5040  RETURN
6000  REM  ***********************
6010  REM -LARGE PRINT SUBROUTINE
6020  REM ——A$ IS STRING TO———
6030  REM  ——  BE PRINTED ————
6040  REM ————CENTERS TEXT—
6050 X =  ABS (20 -  LEN (A$) * 2.5)
6060  FOR W = 1 TO  LEN (A$)
6070  IF  ASC ( MID$ (A$,W,1)) = 32 THEN X = X + 2: GOTO 6160
6080  IF  MID$ (A$,W,1) = "?" THEN  GOSUB 8010: GOTO 6160
6090  IF  MID$ (A$,W,1) = "!" THEN  GOSUB 8080: GOTO 6160
6100  IF  MID$ (A$,W,1) = "," THEN  GOSUB 8130: GOTO 6150
6110  IF  MID$ (A$,W,1) = "." THEN  GOSUB 8180: GOTO 6160
6120  IF  MID$ (A$,W,1) = "/" THEN  GOSUB 8230: GOTO 6160
6125  IF  MID$ (A$,W,1) = ";" THEN  GOSUB 8270: GOTO 6160
6130 P =  ASC ( MID$ (A$,W,1)) - 64
6140  ON P GOSUB 6200,6270,6350,6420,6490,6560,6620,6700,6750,6810,6860,69
     00,6950,7010,7070,7130,7190,7270,7360,7440,7480,7530,7590,7650,7700,7
     750
6150 X = X + 6
6160  NEXT
6170  GOSUB 5020
6180  RETURN
6200  REM ————PRINTS LETTER A
6210  PLOT X + 2,Y
6220  PLOT X + 1,Y + 1: PLOT X + 3,Y + 1
```

We have found that LETTERMATCH provides an excellent introduction to computers for quite young children. It can be modified (for example, by considerably shortening the delay loops) for use to introduce other users to computers and to sharpen keyboarding skills of users of any age.

LETTERMATCH occupies about 6400 bytes of memory. The subroutines which format and plot the characters (beginning in line 5000) occupy about 3700 bytes. Thus, any Applesoft program which leaves 3700 bytes of memory free when loaded can be modified to give large, multi-colored responses by merging these subroutines with the existing program (using the Apple RENUMBER program), making minor changes in the main program (adding GR, selecting colors, etc.), then modifying each PRINT statement to use the subroutine. For example, the line "200 PRINT "VERY GOOD!" would be changed to "200 A$ = "VERY GOOD!": GOSUB 6000."

The character set as developed includes the upper case letters A to Z and the question mark, exclamation point, comma, period, single quotation marks, and semicolon. The set could easily be expanded to include lower case letters, numerals, and other punctuation. The program randomly selects the color of each line of characters.

Copies of LETTERMATCH on diskette or cassette are available from the authors at the above address for the cost of duplication. While we reserve all commercial rights to these programs, we offer them free to any user for any non-commercial educational purpose. Other utilities of this kind which we have developed include routines for adding sound effects and music to programs, additional graphics (such as screen borders), and others. These will appear in later issues of **COMPUTE!** We hope that teachers and other computer users will find these procedures a useful addition to their program collection.

```
6230 VLIN Y + 2,Y + 6 AT X + 4
6240 HLIN X,X + 4 AT Y + 4
6250 VLIN Y + 2,Y + 6 AT X
6260 RETURN
6270 REM ----PRINTS B
6280 VLIN Y,Y + 6 AT X
6290 HLIN X,X + 2 AT Y
6300 HLIN X,X + 2 AT Y + 3
6310 HLIN X,X + 2 AT Y + 6
6320 VLIN Y + 1,Y + 2 AT X + 3
6330 VLIN Y + 4,Y + 5 AT X + 3
6340 IF X > 0 THEN X = X - 1: RETURN
6350 REM -----PRINTS C
6360 VLIN Y,Y + 6 AT X
6370 HLIN X,X + 3 AT Y + 6
6380 HLIN X,X + 3 AT Y
6390 PLOT X + 3,Y + 1
6400 PLOT X + 3,Y + 5
6410 X = X - 1: RETURN
6420 REM -----PRINTS D
6430 VLIN Y,Y + 6 AT X
6440 VLIN Y + 1,Y + 5 AT X + 3
6450 HLIN X,X + 2 AT Y
6460 HLIN X,X + 2 AT Y + 6
6470 X = X - 1: RETURN
6480 PRINT "1799": END
6490 REM --------PRINTS E
6500 VLIN Y,Y + 6 AT X
6510 HLIN X,X + 3 AT Y
6520 HLIN X,X + 2 AT Y + 3
6530 HLIN X,X + 3 AT Y + 6
6540 X = X - 1
6550 RETURN
6560 REM -------PRINTS F
6570 VLIN Y,Y + 6 AT X
6580 HLIN X,X + 3 AT Y
6590 HLIN X,X + 2 AT Y + 3
6600 X = X - 1
6610 RETURN
6620 REM ----------PRINTS G
6630 VLIN Y,Y + 6 AT X
6640 HLIN X,X + 3 AT Y + 6
6650 HLIN X,X + 3 AT Y
6660 PLOT X + 3,Y + 5: PLOT X + 3,Y + 4
6670 PLOT X + 2,Y + 4
6680 X = X - 1
6690 RETURN
6700 REM ----------PRINTS H
6710 VLIN Y,Y + 6 AT X
6720 VLIN Y,Y + 6 AT X + 3
6730 HLIN X,X + 3 AT Y + 3
6740 X = X - 1: RETURN
6750 REM ------------PRINT I
6760 HLIN X,X + 2 AT Y
6770 HLIN X,X + 2 AT Y + 6
6780 VLIN Y,Y + 6 AT X + 1
6790 X = X - 2
6800 RETURN
6810 REM -------PRINTS J
6820 HLIN X,X + 4 AT Y
6830 VLIN Y,Y + 5 AT X + 3
6840 PLOT X,Y + 5: HLIN X + 1,X + 2 AT Y + 6
6850 RETURN
6860 REM ------------PRINTS K
6870 VLIN Y,Y + 6 AT X
6880 PLOT X + 3,Y + 1: PLOT X + 2,Y + 2: PLOT X + 1,Y + 3: PLOT X + 1,Y +
     4: PLOT X + 2,Y + 5: PLOT X + 3,Y + 6
6890 X = X - 1: RETURN
6900 REM ------------PRINTS L
6910 VLIN Y,Y + 6 AT X
6920 HLIN X,X + 3 AT Y + 6
6930 X = X - 1: RETURN
6940 RETURN
6950 REM ------------PRINTS M
6960 VLIN Y,Y + 6 AT X
6970 VLIN Y,Y + 6 AT X + 4
6980 PLOT X + 1,Y + 1: PLOT X + 3,Y + 1
6990 PLOT X + 2,Y + 2
7000 RETURN
7010 REM ------------PRINTS N
7020 VLIN Y,Y + 6 AT X
7030 VLIN Y,Y + 6 AT X + 4
7040 PLOT X + 1,Y + 1: PLOT X + 2,Y + 2: PLOT X + 2,Y + 3: PLOT X + 3,Y +
     4
```

```
7050  PLOT X + 3,Y + 5
7060  RETURN
7070  REM ---------------PRINTS O
7080  VLIN Y,Y + 6 AT X
7090  HLIN X,X + 3 AT Y
7100  HLIN X,X + 3 AT Y + 6
7110  VLIN Y,Y + 6 AT X + 3
7120 X = X - 1: RETURN
7130  REM ---------------PRINTS P
7140  VLIN Y,Y + 6 AT X
7150  HLIN X,X + 3 AT Y
7160  HLIN X,X + 3 AT Y + 3
7170  VLIN Y,Y + 3 AT X + 3
7180 X = X - 1: RETURN
7190  REM ---------------PRINTS Q
7200  VLIN Y,Y + 6 AT X
7210  VLIN Y,Y + 6 AT X + 4
7220  HLIN X,X + 4 AT Y
7230  HLIN X,X + 4 AT Y + 6
7240  PLOT X + 3,Y + 5: PLOT X + 2,Y + 4
7250  PLOT X + 2,Y + 4
7260  RETURN
7270  REM ---------------PRINTS R
7280  VLIN Y,Y + 6 AT X
7290  HLIN X,X + 3 AT Y
7300  HLIN X,X + 3 AT Y + 3
7310  VLIN Y,Y + 3 AT X + 3
7320  PLOT X + 1,Y + 4: PLOT X + 2,Y + 5
7330  PLOT X + 3,Y + 6
7340  IF X > 1 THEN X = X - 1
7350  RETURN
7360  REM ---------------PRINTS S
7370  HLIN X,X + 3 AT Y
7380  HLIN X,X + 3 AT Y + 6
7390  HLIN X,X + 3 AT Y + 3
7400  VLIN Y,Y + 3 AT X
7410  VLIN Y + 3,Y + 6 AT X + 3
7420 X = X - 1
7430  RETURN
7440  REM ---------------PRINTS T
7450  VLIN Y,Y + 6 AT X + 2
7460  HLIN X,X + 4 AT Y
7470  RETURN
7480  REM ---------------PRINTS U
7490  VLIN Y,Y + 6 AT X
7500  VLIN Y,Y + 6 AT X + 3
7510  HLIN X,X + 3 AT Y + 6
7520 X = X - 1: RETURN
7530  REM ---------------PRINTS V
7540  VLIN Y,Y + 3 AT X
7550  VLIN Y,Y + 3 AT X + 4
7560  VLIN Y + 4,Y + 5 AT X + 1: VLIN Y + 4,Y + 5 AT X + 3
7570  PLOT X + 2,Y + 6
7580  RETURN
7590  REM ---------------PRINTS W
7600  VLIN Y,Y + 5 AT X
7610  VLIN Y,Y + 5 AT X + 4
7620  PLOT X + 1,Y + 6: PLOT X + 3,Y + 6
7630  VLIN Y + 4,Y + 5 AT X + 2
7640  RETURN
7650  REM ---------------PRINTS X
7660  VLIN Y,Y + 1 AT X: VLIN Y,Y + 1 AT X + 4
7670  VLIN Y + 5,Y + 6 AT X: VLIN Y + 5,Y + 6 AT X + 4
7680  PLOT X + 1,Y + 2: PLOT X + 3,Y + 2: PLOT X + 1,Y + 4: PLOT X +
  4: PLOT X + 2,Y + 3
7690  RETURN
7700  REM ---------------PRINTS Y
7710  VLIN Y,Y + 1 AT X: VLIN Y,Y + 1 AT X + 4
7720  PLOT X + 1,Y + 2: PLOT X + 3,Y + 2
7730  VLIN Y + 3,Y + 6 AT X + 2
7740  RETURN
7750  REM ---------------PRINTS Z
7760  HLIN X,X + 4 AT Y
7770  HLIN X,X + 4 AT Y + 6
7780  PLOT X + 4,Y + 1: PLOT X + 3,Y + 2: PLOT X + 2,Y + 3: PLOT X +
  4: VLIN Y + 5,Y + 6 AT X
8000  REM --PUNCTUATION ROUTINES-
8010  REM ---- QUESTION MARK----
8020  PLOT X,Y + 1: HLIN X + 1,X + 3 AT Y
8030  VLIN Y + 1,Y + 2 AT X + 4
8040  HLIN X + 2,X + 3 AT Y + 3
8050  PLOT X + 2,Y + 4: PLOT X + 2,Y + 6
8060 X = X + 6
8070  RETURN
8080  REM --- EXCLAMATION POINT
8090  VLIN Y,Y + 4 AT X + 1
8100  PLOT X + 1,Y + 6
8110 X = X + 3
8120  RETURN
8130  REM ----- COMMA -----
8140  VLIN Y + 5,Y + 6 AT X
8150  VLIN Y + 5,Y + 7 AT X + 1
8160 X = X + 4
8170  RETURN
8180  REM ----- PERIOD -----
8190  PLOT X,Y + 6
8200 X = X + 3
8210  RETURN
8220  RETURN
8230  REM ---SINGLE QUOTES---
8240  VLIN Y,Y + 1 AT X
8250 X = X + 2: RETURN
8260  END
8270  REM -----SEMICOLON---
8280  VLIN Y + 2,Y + 3 AT X
8290  VLIN Y + 2,Y + 3 AT X + 1
8300  VLIN Y + 5,Y + 6 AT X
8310  VLIN Y + 5,Y + 7 AT X + 1
8320 X = X + 4
8330  RETURN
8340  END
```

©

Decrementing The For... Next & Endless Loops

Derek Kelly

Scrolling text *backward* in a RUNning program can be programmed almost as easily as scrolling text forward. Scrolling text forward can be easily accomplished, as can be seen in the program in figure 1:

```
10 For I = 1 to 20
20 Print A$(I)
30 Get G$
40 Next I.
```

The GET command in Applesoft BASIC allows a pause-until-any-key-is-hit way for a user to control the progress of a program. Here, after every string printed (A$(I), a pause is inserted. After each pause, the printing of strings will proceed in a sequential manner, incrementing the counter for 'I' by 1 each time. This is forward scrolling.

But what if you want to go backward? Suppose you want the previous string by a count of 1, 2, to the first printed string? Can you go backward? The answer is Yes, and it's simple to accomplish. But like the various problems associated with For... Next loops generally, if not properly constructed, e.g., with no GOTO's that branch away totally from the loop, is that For... Next loops can involve endless loops, whether such loops run forward or backward.

In any well-constructed rogram module, there should be only one point of entrance to that module, and only one real exit. Some programmers who use GOTO, and related statements, have been known to construct programs where *under normal use* the program will "hang" somewhere, a somewhere often caused by a GOTO or a number of GOTO's which send the program around in circles. For example, consider programs A and B below. A has one entrance and exit, B has many:

```
10 GOSUB 100
 .

100 For I = 1 to 230: Print "!";: GOSUB 5
110 GOSUB 200: GOSUB 300
120 NEXT I
130 RETURN
```

Program A

```
10 For I = 1 to 33
20 If I = Int(I/11) Then GOTO 101
30 GET G$: If G$ = "?" Then GOTO
   163
 .
 .
 .
163 Print " = ": GOTO 194
101 For J = 1 To 3: If I = J
   Then GOTO 121: Next J
102 GOTO 30
```

Program B

Program A is called by a GOSUB from line 10. Program A does one set of functions in the program as a whole. When called, it does what it's supposed to, and then returns. Program B, on the other hand, GOTO's all over the place, and one can't predict that any one GOTO will even find its way back to the place that started the GOingTO process. Under normal conditions, and presupposing that the GOSUBs called by Program A are well-constructed, then we can reasonably predict that program A will not cause an endless loop, while Program B most certainly *may* contain an endless loop or two that will "hang" your computer.

The two short programs above affect the forward running of For... Next loops. If a user desires to decrement rather than increment a For... Next loop so as to reverse the order of a printout, and be able to review previously printed items, then other endless loop possibilities arise.

Take Program A as an example. I might want to add a pause to the program, so I add III Get G$: If G$ = "?" Then GOTO 130. This GOTO is OK because it takes place *within* the routine.

Now I want to add a provision to reverse the loop. Since the "NEXT I" statement increments a counter by 1 (or whatever), to reverse print, I must subtract 2 from the I counter to get the next previous record. So my program now looks like PROGRAM C:

```
100 For I = 1 to 230: Print "!";: GOSUB 5
110 GOSUB200 : GOSUB300)
111 Get G$: If G$ = "-" Then I = I-2
120 NEXT I
130 RETURN
```

Program C

As soon as Program C is RUN, you will see that it decrements to 1 then to -1 and so on. You don't want negative numbers, so another line of code will have to be added. In addition, when the counter gets to '-1', the computer — at least *my* APPLE — hangs and prints out a steady, unstoppable — except by RESET, stream of 1's... 1.1.1.1.1.1.1.1.1.1.... This problem can be avoided by changing line 111 to read:

111 Get G$: IF G$ = "-" Then I = I-2: If

I< = 0 Then I = 0 If I = 0 or less than zero, then the first item will be printed, as I will = 0 *before* the execution of the NEXT which will increment I by 1 (or more) to 1.

The simple program C, if improved to include the line 111 above, will be able to process the forward and backward scrolling in any program in which it is called.

In general, there is no formula that can guarantee that a program has no endless loops. It is always possible for some bug such as an endless loop to exist under certain conditions in any program. Since you can't be sure about the absence of bugs, you have to make do with the presence of controls to limit the possibility of harmful bugs. ©

HELP WANTED

Apple Authors

COMPUTE! is looking for applications articles aimed at beginners and intermediate programmers. We're specifically interested in programming hints, tutorials, articles written to help users get more out of their machine. We're not afraid of re-inventing the wheel (those useful programming tricks you learned two years ago are just as relevent to a beginner now), but we do insist on high quality, pertinant information. We pay on a published page basis, and that includes program listings as well. Here are our brief guidelines: Text should be typed in upper/lower case, double spaced, with your name and name of the article appearing at the top of each page. Program listings should be single spaced, with a new ribbon on your printer.

Apple User Groups

Please take the time to announce this at your next meeting. COMPUTE! is serious about supporting the Apple with the same high quality information we provide for other 6502 machines. To do that, we need your support as well.

Mailing address: COMPUTE!
Post Office Box 5406
Greensboro, NC 27403 USA
Telephone: (919) 275-9809

Got A Question You'd Like Answered?

Write: Ask The Readers,
c/o **COMPUTE!**,
P.O. Box 5406,
Greensboro, NC 27403

THE ATARI® GAZETTE

Super Cube

Steve Steinberg
Washington, DC

If you own an Atari and like to fiddle with graphics you're going to love this one. It's a program that allows you to draw cubes in different sizes and colors and also to use the cube itself as a cursor to make three dimensional bars, columns, and drawings reminiscent of the works of Escher. It can create some of the most impressive graphics displays you've yet seen on your monitor, and, to crow just a bit, it *can't* be done on an Apple.

Now, after all that buildup, a bit of mea culpa. While this program proves that a relative novice (never touched a computer until I bought my Atari last year) can make things happen on a home computer, the program is really nothing more than a synthesis of other people's work.

It all began when I started fooling around with the brilliant little program by David D. Thornburg that appeared in the May/June 1980 issue of **COMPUTE!** Mr. Thornburg's program enables you to draw a shaded three dimensional cube and change the colors by use of the keyboard. While playing with that one, Ray Daly of the Program Store in Washington, D.C., was kind enough to give me a program he wrote that puts a cursor on the screen and allows you to draw with a joystick. My question: why not change Ray's PLOT X,Y to GOSUB to a subroutine that will draw with a cube instead of with a point? My solution, with a few minor embellishments: to take David Thornburg's DRAW sequence and do just that.

Sound simple? It was, almost unbelievably so. Since then, I've been having a ball creating skyscrapers, harbors, cubist drawings and the like, and while I plead guilty to the charge of total plagiarism, it would be a shame not to share my fun with other Atari owners. Sorry, I can't tell you exactly how Ray's program works. I never really had to know. I can tell you that lines 600 to 670 set up the colors for the cube (sorry about that too, you can only get about 90 different color combinations!). Lines 720 to 830 are the cube drawing itself, with the input variable SQ setting the size. As far as the joystick commands are concerned, check them out in Atari's Basic Reference Manual; it's fairly easy to figure out what they do.

Now, for operating the program. The first thing the computer will ask you for is a dimension for the cube. Actually, you can enter numbers as large as 40 to 45, but this will cause problems if the cube or your drawing run off the screen, so until you've had a little practice, stick to dimension sizes between 1 and 10. Once you've entered your cube dimension, press RETURN and your Atari will go to the Graphics 7 mode.

The cursor is in the upper left hand corner of the screen. It may be a little difficult to see if you have a dark initial color, but you can change the display colors at any time by pushing the joystick button when the cursor is anywhere at the far left of the screen.

Next, using your joystick, move the cursor to wherever you want to begin your work of art and push the joystick button. Voila, a cube!

Whenever you let go of the joystick button, the cursor returns to the upper left hand corner of the screen. This keeps it out of the way of your drawing, since the cursor draws an erase line as it moves along. If you want to draw a bar or a column, or draw diagonally, hold the joystick button down while moving the cursor.

When you want to draw with a different sized cube, move the cursor to the bottom of the screen and press the joystick button. In the text window at the bottom of the screen, you'll be asked for a new dimension for the cube. Enter it, hit RETURN, and the cursor goes back to the upper left hand corner.

Certain three dimensional effects require that you go over what you have drawn. For example, to make a square shaped "O": Give yourself a cube dimension of, say, 6. Start somewhere near the center of the screen. Go up, straight up, about ten increments, across to the right another ten, down ten, and then draw carefully to the left, stopping when the left hand side of the drawing is just joined. Now, go back across to the right, stopping at the end of your original bottom bar line, and draw back up to the top.

Again, any time you want to change colors (it's more fun when whatever you're drawing is complete) either release the joystick button so that the cursor goes back to the upper left hand corner or just move the cursor to the far left of the screen and press the joystick button. Every now and then when you change colors, a cube will appear at the far left. If anyone figures out why this happens, please let me know.

This program is on the border line between 8K and 16K, so if you have only an 8K computer you will probably want to pack some of the program lines together, and eliminate the first two print statements. I've checked it out with only 8K in my Atari, and I can assure you it will work.

The program also provides a dramatic demonstration of how much memory graphics require. Try this: load the program, type RUN, then BREAK, and then PRINT FRE(0) and RETURN, which will tell you how much memory you have left. Now RUN the program again, enter an initial cube dimension, and when the program goes to Graphics Mode 7, type BREAK and then PRINT FRE(0), and RETURN again. Where did it all go?

One final note: As I've mentioned, I'm a novice at all this and Super Cube could certainly use some additional improvement. For example, how about using a second joystick to draw pyramids?

Or controlling the colors with the keyboard instead of making them random? I'd be pleased to hear from anyone who has any further ideas to offer. In the meantime, Picasso! Move over! A new generation of cubists is about to begin work!

Editor's Note: As you update Steve's program, send in your enhancements and we'll keep you posted. RCL

```
10 ? "}":POSITION 10,5:? "3-D DRAWING":?
 :? :? "PRESS JOYSTICK BUTTON TO DRAW"
20 PRINT :PRINT :PRINT "TO CHANGE CUBE D
IMENSION MOVE  CURSOR TO BOTTOM OF SCREE
N AND PRESS BUTTON"
30 ? :? :? "ENTER CUBE DIMENSION, THEN H
IT RETURN:(Note: Dimensions larger than
10 may  cause Errors)":INPUT SQ
40 GRAPHICS 7
50 PRINT "TO CHANGE COLORS PRESS BUTTON
WHEN    CURSOR IS AT EXTREME LEFT OF SCR
EEN"
70 GOSUB 600
80 IF STRIG(0)<>0 THEN GOSUB 500:GOTO 14
0
90 IF Y>75 THEN PRINT "ENTER NEW DIMENSI
ON FOR CUBE";:INPUT SQ:X=0:Y=0
120 POKE 77,0
130 IF STRIG(0)=0 THEN GOSUB 700
140 GOSUB 1000
150 X=X+XDIF:Y=Y+YDIF
200 IF X>143 THEN X=143:GOTO 300
210 IF Y>80 THEN Y=80:GOTO 300
300 IF X<0 THEN X=0:GOTO 400
310 IF Y<7 THEN Y=7
400 GOTO 80
500 COLOR 1:PLOT X,Y:FOR I=1 TO 5:NEXT I
:COLOR 4:PLOT X,Y:RETURN
600 A=INT(RND(1)*15)+1
610 B=INT(RND(1)*14)+2
620 IF B<4 THEN B=10
630 SETCOLOR 1,A,B
640 SETCOLOR 2,A,B-2
650 SETCOLOR 0,A,B-4
660 IF X<10 THEN GOTO 120
670 RETURN
700 IF X=0 THEN IF STRIG(0)=0 THEN GOSUB
 600
710 TRAP 80
720 COLOR 1
730 FOR I=0 TO SQ
740 PLOT X,Y+I:DRAWTO X+SQ,Y+I
750 NEXT I
760 COLOR 2
770 FOR I=1 TO INT(3*SQ)/5
780 PLOT X+I,Y-I:DRAWTO X+I+SQ,Y-I
790 NEXT I
800 COLOR 3
810 FOR I=1 TO INT(3*SQ)/5
820 PLOT X+SQ+I,Y-I:DRAWTO X+SQ+I,Y+SQ-I
+1
830 NEXT I
840 IF STRIG(0)<>0 THEN X=0:Y=0
850 RETURN
1000 WHAT=STICK(0):XDIF=0:YDIF=0
1100 IF WHAT=15 THEN RETURN
1110 IF WHAT=14 THEN YDIF=-1:RETURN
1120 IF WHAT=13 THEN YDIF=1:RETURN
1130 IF WHAT=11 THEN XDIF=-1:RETURN
1140 IF WHAT=10 THEN XDIF=-1:YDIF=-1:RET
URN
1150 IF WHAT=9 THEN YDIF=1:XDIF=-1:RETUR
N
1160 IF WHAT=7 THEN XDIF=1:RETURN
1170 IF WHAT=6 THEN YDIF=-1:XDIF=1:RETUR
N
1180 IF WHAT=5 THEN XDIF=1:YDIF=1:RETURN
```
©

Printing Characters In Mixed Atari Graphics Modes

Craig Patchett
2 Swan Terrace
Greenwich, CT 06830

For those of you who have been anxiously awaiting this appendum, I apologize. Time conflicts and the discovery of redefinable character sets have prohibited me (until now, of course) from writing it. For those of you who haven't, get hold of the Sept/Oct 1980 issue of COMPUTE! and read up on "Designing Your Own Atari Graphics Modes." This article won't be of much use to you until you do.

The problem, if you recall, is with printing characters on mode lines that are out of the usual range of that mode. For example, if we design a graphics mode such that the thirtieth line is mode two, we would get an error message if we attempted to print on that line. This is because the Atari thinks it is in the regular mode two, which only allows twelve lines of characters. We must therefore find another way to put the characters on the screen.

As you may already realize, the screen is just a type of window looking into a part of memory. If you change that memory, what you see on the screen also changes. The solution, therefore, is just to POKE the characters into the memory locations that correspond to the positions on the screen where we want them to appear.

Where is the screen in memory?

We already know where the display list is in memory; we used the variable BEGIN to point to it last time:

```
BEGIN = PEEK(560) + PEEK(561)*256 + 4
```

But, you may well ask, what does this have to do with the screen memory, or display memory as we will call it here? It just so happens that the first two memory locations in the display list point to the beginning of display memory in the following fashion:

```
DISMEN = PEEK(BEGIN) + PEEK(BEGIN + 1)*256
```

If you recall, we never used the first two memory locations in the display list last time; now you know why.

How do we calculate the exact memory locations to POKE into?

Each mode line used up a certain amount of memory. As you might guess, different modes use different amounts of memory per line. To be more exact:

MODE	0	1	2	3	4	5	6	7	8
MEM/LINE	40	20	20	10	10	20	20	40	40

So all we have to do is figure out how much memory is used before the mode line that we want to print on, and add that to DISMEM to determine where we want to start POKEing. As an example of how to do this, let's suppose we have a gaphics mode with four lines of mode 1, fifty lines of mode seven, three lines of mode four, and three lines of mode two (4X8 + 50X2 + 3X4 + 3X16 = 32 + 100 + 12 + 48 = 192); and we want to print on the second line of mode two. Checking the table above, we go:

4 lines of mode 1 = 4X20 =	80	
50 lines of mode 7 = 50X40 =	2000	
3 lines of mode 4 = 3X10 =	30	
1 line of mode 2 = 1X20 =	20	(remember, we only count the lines *above* the one we want to print on)
For a grand total of:	2130	

Therefore, memory location DISMEM + 2130 represents the first character in the second line of mode 2 *for this particular mode.* Memory location DISMEM + 2131 represents the second character, and so on up to DISMEM + 2149 for the twentieth character.

We know that POKEing the appropriate value into the appropriate location will cause the desired character to appear at the desired screen location. Since we already know how to determine the appropriate memory location, we now ask:

How do I calculate the appropriate value for a character?

It turns out that the value to poke for a given character corresponds to the order in which the character descriptions are stored in ROM (see "Designing Your Own Atari Character Sets" in the March 1981 issue of COMPUTE!). As a quick memory refresher:

ATASCII VALUE	VALUE TO POKE
0-31	64-95
32-95	0-63
96-127	96-127

For reverse characters, just add 128 to the value of the normal character.

My brain is in hibernation; how do I convert a character string to its appropriate values?

I'll leave you with the folowing self-explanatory subroutine that will take the (predefined) character string PRNTME$ and the starting memory location STARTHERE (also predefined and equal to DISMEM + offset) and POKE PRNTME$ into the appropriate (love that word!) memory locations. Enjoy!

```
30000 REM /*This loop will act on each c
haracter in PRINTME$*/
30010 FOR ME=1 TO LEN(PRNTME$)
30020 REM /*Find ATASCII value of charac
ter*/
30030 VALUE=ASC(PRNTME$(ME,ME))
30040 REM /*Subtract 128 temporarily if
it's a reverse character*/
30050 VALUE=VALUE-128*(VALUE>127):REM /*
see note below*/
30060 REM /*Make the appropriate value a
djustments*/
30070 VALUE=VALUE+64*(VALUE<32)-32*(VALU
E>31 AND VALUE<96)
30080 REM /*Convert back to reverse if n
ecessary*/
30090 VALUE=VALUE+128*(ASC(PRNTME$(ME,ME
)))127)
30100 POKE STARTHERE+ME-1,VALUE:REM /*re
member, ME starts at 0, not 1*/
30110 ? VALUE
30120 REM /*Go to next character*/
30130 NEXT ME
30140 REM /*All done, say goodbye*/
30150 RETURN
```

Note that (condition) equals 1 if the condition is true, Ø if it's not. Thus, X = 126:PRINT (X = 126) :PRINT (X = 127) will print a 1 followed by a Ø.

SPECIAL CORRECTION NOTE

The original article, *"Designing Your Own Atari Graphics Modes,"* was incorrect in that the graphics mode you use initially should be a full screen mode. In other words, if graphics mode six is the mode that uses the most memory out of the modes you will be mixing, you should use GRAPHICS 22, not GRAPHICS 6 (just add 16 to the mode number). In the example used in the article, line 10 should read:

1Ø GRAPHICS 17 and not 1Ø GRAPHICS 1

My apologies for any problems caused by this mistake. ©

Invaders From Outer Space

An Atari Program Using One Joystick

David H. Markley
Reynoldsburg, Ohio

This program is a simplified version of an earlier program I wrote called "Alien Landers" and is designed to run in 8K of memory. The object of the game described here, is to protect your galactic base from an invasion of allien saucers of unknown origin. Your weapon against the alien saucer is a sophisticated photon cannon which you control by a joystick connected to controller port #1. The saucer on the other hand, is equipped with a mysterious paralyzing death ray which when energized, will immobilize and possibly vaporize your photon cannon. The game is played by defending your base against the saucers and achieving the highest score possible before your 3 photon cannons are destroyed. The program keeps track of the highest score obtained during any individual game and displays the final game score and the high game score at the end of each battle. After starting the game using the RUN command, additional battles are initiated by holding the start button down until the new game begins.

This program utilizes many of the fine features of the ATARI personal computer, such as the exciting sound effects and the advanced player/missile graphics. The player/missile graphics are used to enable the program to provide good animation without the overhead of saving and restoring the background as the player or missile image moves through the playfield. Another ATARI feature used here is the player/missile collision registers. These registers are associated with the ATARI's graphics and indicate when a player or missile overlaps with another player, missile, or playfield image.

The program from line 30 thru line 600 provides background setup and player/missile initialization. Initialization begins by allocating space for the player/missile image buffers. The buffers are placed in a free area of memory just below the graphics memory with the base address located 2048 bytes (8 pages) from the top of memory. This leaves the top 1K of memory for the graphics 2 display list and mapping data.

The functional player/missile (P/M) graphics area begins with the missile image buffer. Since the program uses the double resolution mode of P/M graphics, the missile image buffer begins 384 bytes from the base address and is followed by four player image buffers. Each buffer occupies 128 bytes of memory. Invaders from Outer Space uses only two of the four available players which are in the form of a saucer (player 1) and a photon cannon (player 2). The images of the saucer and cannon are formed by placing a bit pattern of the shapes into their corresponding player image buffers.

Lines 600 and 700 initialize the game's counters and registers. The high game score counter initialized by line 600 is used to keep track of the highest score obtained for all games played during any program run. This counter is only initialized one time. Registers and counters which are initialized every time a new game is played are all located at line 700.

With the playfield and players enabled and the games counters and registers initialized, the program is ready to begin antimating the graphics by entering the game's antimation sequencer loop (lines 1000 thru 2000). This section of the program is used to control the movement of the saucer, cannon, and photon. It also determines when to fire the saucer's death ray and checks for P/M collisions.

The sequencer loop begins by determining the saucer's horizontal position in relation to the play field. Normally the saucer will travel across the field in a left to right direction in steps of 5 horizontal increments. If the saucer is within the range of fire of the cannon however, and is currently under attack (M>0) the saucer will take evasive action by moving randomly within a short distance of either side of its current position. With the position of the saucer determined, the sequencer's next task is to produce the saucer's sound effect. This effect is created by stepping through a series of six frequencies in which the frequency is changed one step for each loop cycle. Once the saucer has been placed in its new position, a test is made to see if it is in attack range of the cannon. If the test (line 1060) indicates that the saucer is within 15 increments of the cannon, the program will go to the death ray handler (line 3100).

The next sequencer task to be handled is the control of the cannon and its associated photon missile (lines 1100 thru 1220). The program begins by checking to see if the photon is ready to be fired. If it is ready to fire or has reached the top of the playfield (M<J +8), its sound is turned off and the joystick trigger is examined. If the trigger is not pressed, the program will continue through the sequencer loop and will examine the trigger each time it passes through the loop. When the trigger is finally pressed, the M pointer will be set to an address of the missile image buffer which corresponds to a vertical position directly above the cannon's muzzle.

The horizontal position of the photon is handled by calculating the horizontal position of the cannon's center and placing it into the cannon's horizontal positioning missile register. Line 1110 controls the movement of a photon which is enroute to its target. To move an object vertically using the P/M graphics, the image is first removed from its current position within the image buffer and then rewritten into its new location. The P/M graphics function in such a way that as the base address of the image data is moved to a lower numbered address in the buffer, the image will appear to move upward on the screen. After each photon movement, the player/missile collision register is checked to see if the saucer has been hit. When a hit is detected the program exits the sequencer loop and goes to the routine at line 3000 to handle the collision. If the saucer is not hit the sequencer loop enters the code which handles cannon movement. When the joystick is moved to the left or right the cannon is shifted 5 increments to the left or right of its current position each pass through the loop. With the completion of this code, the sequence loop is now ready for the next pass.

The routine starting at line 3000 is used to control hits on the saucer. It begins by removing the photon from the playfield by clearing the first 29 bytes of the missile image buffer. This area represents the area on the playfield in which the photon can collide with the saucer. When the buffer has been cleared, the missile position pointer M is reset to zero as an indication that it is ready to be fired again. To help produce the effect of more than one saucer, the saucer's color is randomly changed each time it is hit. Finally the score is updated and the background flag (BK) is set to indicate an above ground explosion. The program then goes to the explosion handler routine (line 4000).

The routine starting at location 3100 is used to generate the saucer's death ray. It begins by positioning the ray at the center of the saucer. This is done by calculating the center of the saucer and placing this position into the ray's horizontal position missile register. The sound and visual effects are handled by lines 3120 thru 3140. The ray is first drawn by setting bits 0 and 1 for every third byte in the missile image buffer. It is then erased by again accessing every third byte and clearing bits 0 and 1 to zero. As the ray is removed, the frequency of its associated sound generator is decreased. Upon completing the death ray, the program examines the P/M collision register corresponding to the ray missile and the cannon player. If the collision took place, an explosion effect is produced by moving the cannon to a position off the playfield and setting the explosion back ground flag for the bottom text window before calling the explosion handler at line 4000.

The explosion handler is used to produce the explosion effect for both the saucer and the cannon. To begin, the explosion handler resets the P/M collision registers. The collision registers are designed to be latching and must be reset each time a collision has been registered. After terminating the sound of the photon which may have caused the explosion, the program begins to produce the visual and audio effects of the explosion. This is done by stepping the variable X from 14 to 0 and using its value to control the intensity of the color and sound. The background which is determined by BK was selected by the calling program before the explosion handler was entered. When the explosion is complete, the playfield colors are returned to their normal hue.

When the explosion effect is complete, the program is ready to display the cannon number or score. Since the program displays all messages at the top of the screen using graphics 2, the saucer must be removed from the screen by setting its P/M color registers to zero. The saucer's original color is stored in X and will be again restored at the completion of the message display cycle. After a short delay while the message is being displayed, a test is made to determine if the last cannon has been hit. If not, the message is removed and the background is restored and the game continues. When the game is finally completed, the score is compared to the high game score. If the score resulting from the last game is higher, the high game score is updated. The program indicates the end of the game by alternately displaying the last game and high game scores. After each display of the high game score the start switch is examined to see if a new game is requested.

```
10 REM INVADERS FROM OUTER SPACE
15 REM COPYRIGHT (c) 1980
20 REM BY DAVID H. MARKLEY
```

Print title

```
25 GRAPHICS 2
30 PRINT #6;"     INVADERS"
35 PRINT #6;"       FROM"
40 PRINT #6;"    OUTER SPACE"
```

Start player/missile data list 2K from top of memory. Place in P/M base register and calculate beginning address of the player data lists.

```
50 I=PEEK(106)-8:POKE 54279,I
60 J=I*256+384
```

Clear P/M buffer area

```
70 FOR X=J TO J+384
80 POKE X,0
90 NEXT X
```

Generate play field

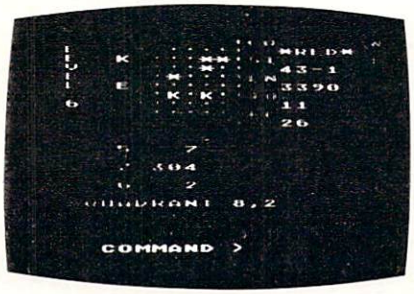
```
100 GRAPHICS 2
110 SETCOLOR 2,11,6
120 COLOR 3
130 POKE 84,0:POKE 85,4:? #6;".",".
140 POKE 84,5:POKE 85,2:? #6;"*    .
150 POKE 84,3:POKE 85,8:? #6;".",".
160 POKE 84,7:POKE 85,14:? #6;".
```

Enable player/missile DMA for double line resolution.

```
200 POKE 559,46
```

Place saucer image data into first player's image buffer area and set its color to green (200). Set the saucer's initial horizontal position to 70.

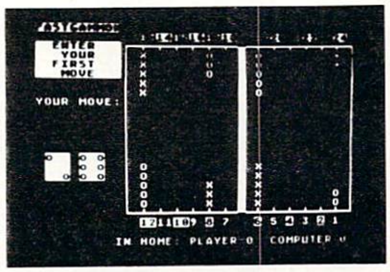
SOFTWARE FOR THE ATARI 800* AND THE ATARI 400*



TARI TREK™
By Fabio Ehrengruber

Get ready for an exciting trek through space. Your mission is to rid the galaxy of Klingon warships, and to accomplish this you must use strategy to guide the starship Enterprise around stars, through space storms, and amidst enemy fire. Sound and color enliven this action-packed version of the traditional trek game. Nine levels of play allow the player to make the mission as easy or as challenging as he wishes. At the highest level you are also playing against time. Damage to your ship can be repaired in space at a cost of time and resources if you can't make it back to base. TARI TREK gives you a lot of trek at a low price. This program is written entirely in BASIC and requires at least 24K of user memory. For the Atari 800 only.

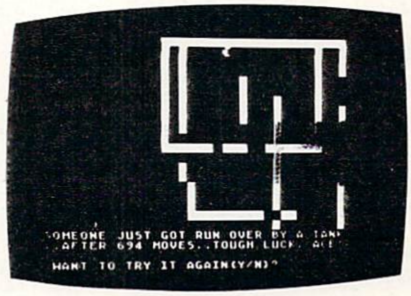
Cassette - $11.95 Diskette - $14.95



FASTGAMMON™
By Bob Christiansen

Play backgammon against a talented computer opponent. This is the latest and best version of the most popular backgammon-playing program for personal computers - FASTGAMMON. Roll your own dice or let the computer roll them for you. Adjust the display speed to be fast or slow. If you wish you can play a game using the same dice rolls as the previous game - a great aid in improving your skills at backgammon. Beginners find it easy to learn backgammon by playing against the computer, and even very good players find it a challenge to beat FASTGAMMON. The 12-page instruction booklet includes the rules of the game. Written in machine language. Requires only 8K of RAM and runs on both the Atari 400 and the Atari 800.

On cassette only - $19.95



TANK TRAP
By Don Ursem

A rampaging tank tries to run you down. You are a combat engineer, building concrete barriers in an effort to contain the tank. Use either the keyboard or an Atari joystick to move your man and build walls. If you trap the tank you will be awarded a rank based on the amount of time and concrete you used up. But they'll be playing taps for you if you get run over. There are four levels of play. Higher levels of play introduce slow curing concrete, citizens to protect, and the ability of the tank to shoot through any wall unless you stay close by. Music, color, and sound effects add to the excitement. Written in BASIC with machine language subroutines. Requires at least 16K of user memory. Runs on the Atari 800 and on an Atari 400 with 16K RAM.

Cassette - $11.95 Diskette - $14.95

QS FORTH™ By James Albanese. Step into the world of the remarkable FORTH programming language. Writing programs in FORTH is much easier than writing them in assembly language, yet FORTH programs run almost as fast as machine code and many times faster than BASIC programs. QS FORTH is based on fig-FORTH, the popular model from the FORTH Interest Group that has become a standard for microcomputers. QS FORTH is a disk-based system that can be used with up to four disk drives. There are five modules included:

1. The FORTH KERNEL (The standard fig-FORTH model customized to run on the Atari computer).
2. An EXTENSION to the basic vocabulary that contains some handy additional words.
3. An EDITOR that allows editing source programs (screens) using Atari type editing.
4. An IOCB module that makes I/O operations easy to set up.
5. An ASSEMBLER that allows defining FORTH words as a series of 6502 assembly language instructions.

Modules 2-5 may not have to be loaded with the user's application program, allowing for some efficiencies in program overhead. Full error statements (not just numerical codes) are printed out, including most disk error statements. QS FORTH requires at least 24K of RAM and at least one disk drive. For the Atari 800 only.

On diskette only - $79.95

★ ★ ★ ★ ★ ★ ★ ★ ★ ★

ASSEMBLER by Gary Shannon. Write your own 6502 machine language programs with this inexpensive in-RAM editor/assembler. Use the editor to create and edit your assembler source code. Then use the assembler to translate the source code into machine language instructions and store the code in memory. Simple commands allow you to save and load the source code to and from cassette tape. You can also save any part of memory on tape and load it back into RAM at the same or at a different location. The assembler handles all 6502 mnemonics plus 12 pseudo-ops that include video and printer control. Commenting is allowed and error checking is performed. A very useful feature allows you to view and modify hexadecimal code anywhere in memory. Instructions on how to interface machine language subroutines to your BASIC programs are included. ASSEMBLER requires 16K of user memory and runs on both the Atari 800 and the Atari 400.

On cassette only - $24.95

★ ★ ★ ★ ★ ★ ★ ★ ★ ★

6502 DISASSEMBLER by Bob Pierce. This neat 8K BASIC program allows you to disassemble machine code, translating it and listing it in assembly language format on the video and on the printer if you have one. 6502 DISASSEMBLER can be used to disassemble the operating system ROM, the BASIC cartridge, and machine language programs located anywhere in RAM except where the DISASSEMBLER itself resides. (Most Atari cartridges are protected and cannot be disassembled using this disassembler.) Also works as an ASCII interpreter, translating machine code into ASCII characters. 6502 DISASSEMBLER requires only 8K of user memory and runs on both the Atari 800 and the Atari 400.

Cassette - $11.95 Diskette - $14.95

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

WHERE TO GET IT: Call us at (213) 344-6599 for the name of the Quality Software dealer nearest you. If necessary you may order directly from us. Mastercard and Visa cardholders may place orders by telephone. Or mail your check or bankcard number to Quality Software, 6660 Reseda Blvd., Suite 105, Reseda, CA 91335. California residents add 6% sales tax. SHIPPING CHARGES: Within North America orders must include $1.50 for first class shipping and handling. Outside North America the charge for airmail shipping and handling is $5.00. Pay in U.S. currency.

```
300 POKE J+148,24
310 POKE J+149,126
320 POKE J+150,171
330 POKE J+151,126
340 POKE J+152,24
350 POKE 704,200
360 POKE 53248,70
```

Place photon cannon image data into second player's mapping buffer area and set color to orange. Set cannon's initial horizontal position to 160.

```
400 POKE J+347,24
410 POKE J+348,60
420 POKE J+349,24
430 POKE J+350,24
440 POKE J+351,60
450 POKE J+352,126
460 POKE J+353,255
470 POKE J+354,102
480 POKE 705,56
490 CNPOS=160
```

Set saucer player to double width and enable P/M graphics.

```
500 POKE 53256,1
510 POKE 53277,3
```

Initialize game counters and registers.

```
600 HIGH=0
700 M=0:SCORE=0:CN=1
```

This ends program initialization. Begin process loop. Determine saucer's position.

```
1000 FOR POS=15 TO 240 STEP 5
1010 SCPOS=POS
```

Vary saucer position if in range of cannon.

```
1020 IF ABS(POS-CNPOS)<20 AND M<>0 THEN
SCPOS=POS+(25-50*RND(0))
1030 POKE 53248,SCPOS
```

Produce saucer sound effect.

```
1040 FRQ=FRQ+1:IF FRQ>6 THEN FRQ=1
1050 SOUND 0,60/FRQ,10,5
```

If missile is not launched check if in range to activate saucer death ray.

```
1060 IF M<J+12 THEN IF ABS(CNPOS-SCPOS)<
15 THEN 3100
```

If missile is not launched or has reached the top of the screen, set missile flag and sound to zero. Check fire button, if pressed init missile position at cannon.

```
1100 IF M<J+8 THEN M=0:SOUND 1,0,0,0:IF
STRIG(0)=0 THEN M=J+93:POKE 53253,CNPOS+
3
```

If missile is launched then move it to next vertical position.

```
1110 IF M>0 THEN POKE M,0:POKE M-1,0:M=M
-8:POKE M,12:POKE M-1,12:SOUND 1,ABS(M+5
0-J)*0.5,8,15
```

Check missile collision register. If hit then goto 3000 for effect.

```
1120 IF PEEK(53257)=1 THEN 3000
```

Determine cannon horizontal position.

```
1200 IF STICK(0)=11 THEN IF CNPOS>50 THE
N CNPOS=CNPOS-5
1210 IF STICK(0)=7 THEN IF CNPOS<198 THE
N CNPOS=CNPOS+5
```

```
1220 POKE 53249,CNPOS
```
End process loop.
```
2000 NEXT POS:GOTO 1000
```
Saucer hit and explosion handler erase missile, change saucer color, and update earth score.
```
3000 FOR X=0 TO 28:POKE X+J,0:NEXT X:M=0
3010 POKE 704,INT(15*RND(0))*16+8
3020 SCORE=SCORE+25
```
Set explosion in sky.
```
3030 BK=4
3040 GOTO 4000
```
Saucer death ray handler. Position death ray position under saucer.
```
3100 POKE 53252,SCPOS+7
```
Fire ray and produce sound.
```
3120 SOUND 1,4,12,15:SOUND 0,0,0,0
3130 FOR X=25 TO 97 STEP 3:POKE J+X,3:NE
XT X
3140 FOR X=25 TO 97 STEP 3:POKE J+X,0:SO
UND 1,X-23,12,15:NEXT X
3150 SOUND 1,0,0,0
```
Check cannon's missile collision register, if a hit then update cannon counter and produce explosion.
```
3200 IF PEEK(53256)<>2 THEN 2000
3210 CN=CN+1:POKE 53249,0
3220 BK=2
```
Explosion handler program.
clear collision registers and disable missile sound.
```
4000 POKE 53278,255
4010 SOUND 1,0,0,0
```
Produce explosion sound and light.
```
4020 FOR X=15 TO 0 STEP -1
4030 SETCOLOR BK,3,X
4040 SOUND 0,50,0,X
4050 FOR D=0 TO 15:NEXT D
4060 NEXT X
```
Restore colors for sky and earth.
```
4100 SETCOLOR 2,10,6
4110 SETCOLOR 4,0,0
```
Display score.
```
4200 POKE 84,0:POKE 85,2
4210 POKE 53248,0
4220 IF BK=2 AND CN<4 THEN ? #6;"CANNON
";CN:GOTO 4240
4230 ? #6;"SCORE     ";:POKE 85,8:? #6;S
CORE
4240 FOR D=0 TO 200:NEXT D
```
Check if end of game.
```
4300 IF CN<4 THEN 4400
4310 IF SCORE>HIGH THEN HIGH=SCORE
4320 POKE 84,0:POKE 85,2
4330 ? #6;"HIGH     ";:POKE 85,7:? #6;HIGH
4340 POKE 53279,8:FOR D=0 TO 80
4350 IF PEEK(53279)=6 THEN 700
4360 NEXT D:GOTO 4200
```
Restore background.
```
4400 POKE 84,0:POKE 85,2:? #6;"   .",".
  "
4410 GOTO 1000
```
©

Unleash The Power Of Your Atari CPU

Ed Stewart
Uniontown, OH

Would you like to get as much as a 30% increase in speed from your Atari 6502 CPU? Would you also like to get this benefit without any additional capital expense? If your answer is no you probably don't like apple pie either, but if you answer yes read on and I will tell you how to accomplish such a feat with one simple BASIC POKE in the right place.

First a little background information about one of the many things that is going on inside your Atari computer. The particular thing that I want you to know something about is how display information reaches your TV screen. There is a specific hardware chip called ANTIC that has most of the responsibility for seeing that the display gets to your TV screen. ANTIC does this by operating independently from the main 6502 CPU in your computer. ANTIC is in fact a primitive CPU in it's own right. It executes a program which is located in RAM just as the 6502 executes a program in RAM or ROM. We can therefore call the Atari a multiprocessing computer since more than one CPU may be active at any time. A peculiar and somewhat unfortunate thing happens when a multiprocessing system such as the Atari is actively executing instructions—both CPUs desire access to memory simultaneously. The two CPUs cannot both access memory at the same time so one must wait until the other completes it's access request. This memory access conflict is common to all computers containing more than one CPU—from micros to macros—and is generally not something to be concerned about.

The ANTIC chip fetches it's data from memory using a technique called "Direct Memory Access" or DMA. Whenever this memory fetch is occuring the 6502 is temporarily halted. DMA is said to be "stealing" a portion of the computer's available time called a cycle. There are 1,789,790 cycles of computer time available per second. If DMA had not "stolen" that cycle of computer time the 6502 would not have been halted and therefore would have finished it's program instructions sooner. It is only logical to conclude that the more this DMA activity occurs in behalf of the ANTIC chip, the more our 6502 will be slowed down.

The ANTIC chip re-displays the entire TV display 60 times each second. During each of these 60 times many computer cycles are stolen from the 6502. During each of these 60 times the ANTIC chip also "interrupts" the 6502 and causes it to perform such tasks as updating various software timers and reading game controllers (joysticks and paddles). When the 6502 finishes what it must do in response to the ANTIC "interrupt" it may continue with what it was doing previous to being sidetracked by ANTIC. You should be getting the picture by now that although ANTIC is indispensable it causes a slowdown in the 6502 CPU, but how much?

I wrote a simple BASIC program for my Atari 800 in an attempt to answer this question. A FOR/NEXT loop was executed 100,000 times with no intervening statements as follows:

```
20 FOR I = 1 TO 100000:NEXT I
```

The first thing to measure was how long this loop executes with no ANTIC DMA active. A POKE 559,0 turned DMA off and the TV screen went black. A POKE 559,34 turned DMA back on and the original display was restored. The FOR/NEXT loop with *no* DMA required 148 seconds to complete. This same FOR/NEXT loop was executed in graphics modes 0-8 *with* DMA active and the execution times were observed as shown in table 1. The execution times *with* DMA increased from as little as 10% for graphics 3 to as much as 47% for graphics 8. It is reasonable to see that if you do a lot of number crunching and you don't need the TV screen or the software timers and game controllers then turn off the antic DMA for a while and you'll get your answer back sooner. It is also apparent from table 1 that your programs will execute faster if you are executing in graphics modes 3,4, or 5.

I hope you have learned a little bit more about the Atari computer and how the ANTIC DMA interferes with the 6502 CPU. You may in fact someday be able to unleash that latent power within during a computer chess tournament, and when someone asks how in the world you did it you can smile and say -me and my DMA.

GRAPHICS MODE	EXECUTION SECONDS	% INCREASE (over no-DMA)
NO DMA	148	
GRAPHICS 0	216	46
GRAPHICS 1	188	27
GRAPHICS 2	186	26
GRAPHICS 3	163	10
GRAPHICS 4	164	11
GRAPHICS 5	167	13
GRAPHICS 6	173	17
GRAPHICS 7	185	25
GRAPHICS 8	218	47

Table 1: Execution timings with and without DMA active for various graphics modes.

String Arrays in Atari BASIC

Charles Brannon

This article describes a method to simulate string array handling in Atari BASIC. If you already know what a string array is, skip ahead a bit. Otherwise, read on...

An array is essentially a list. For example, the numeric array A could hold a list of monthly amounts. A(1) would hold January, A(2) would contain February's monthly total, and so on, A(12) containing December's amount. To print the amounts from month one to month twelve, the BASIC statement:

```
FOR I = 1 TO 12:PRINT A(I):NEXT I
```

could be used. As the amounts were printed, the computer would optionally add the amounts up and print a yearly total. Numerical arrays also have many uses in mathematics.

If we wanted to print the name of each month along with the amount for that month, string arrays would come in handy. For January, the string array M$ might hold "Jan" (i.e. M$(1) = "JAN"). Once all the strings were defined, we could print the months along with the amounts:

```
FOR I = 1 TO 12:PRINT M$(I), A(I):NEXT I
```

Alternately, a string array could contain a list of player's names in a game. **PRINT P$(PLR)** could give the name of player number PLR.

It is a fairly well-known fact that Atari BASIC is not Microsoft BASIC (although a Microsoft BASIC for Atari should be released soon). The designers of the BASIC decided to use a different way of manipulating strings than Microsoft did. The statement **PRINT LEFT$(A$,4)** in Microsoft BASIC would produce the same effect as **PRINT A$(1,4)** would in Atari BASIC. Unfortunately, this notation precludes the use of string array notation.

Nevertheless, it is possible to produce the same effect as string arrays under Atari BASIC. The technique used here is similar to the one used in the string sort program in the back of the *Atari Basic Reference Manual.*

Essentially, the solution is to partition one very large string into several substrings. Each substring will be an element of the larger array. First, let us review the format that Atari BASIC uses to handle substrings. Say that A$ contains "Hello, how are you?". To print the word "how" we would specify A$(8, 10), the first argument, eight, being the starting position within the string and the second being the ending position of the substring, in this case, 11. We use this notation to recall or store any element from the main string. There are some non-Microsoft notations to specify an element in a string array. **PRINT A$(2,3,6)** would print the characters from three to six in the second element of A$. We will use an "unraveled" string to simulate this.

Say we want to set aside an area of memory for 10 strings, each string having a maximum length of 20 characters. Actually, this is required in Atari BASIC anyway. Since the DIM statement only takes one argument, the total number of characters in the string, we would enter in the program the statement: DIM A$(200), as 10 strings of 20 characters each would be 200 characters in total length. Now, to define each string, we must specify the starting location within the main string and how many characters to store at that location. If the string to be stored was T$, and E is the element that T$ is, then the statement to store T$ in A$ would be:

```
A$( (E-1)*20 + 1, E*20 ) = T$
```

If T = 2 (i.e. T$ would be the second string in A$), then the statement would reduce to: A$(21,40) = T$, and indeed, the area of A$ reserved for element 2 would be characters twenty-one to forty.

To recall an element from the string array, the statement need only be reversed. The element position of T$ still needs to be specified.

```
T$ = A$(E - 1)*20,E*20)
```

The "20" in the statements is of course the maximum number of characters in any element in the array. It could be specified a a variable at the start of your program.

There are still a few problems with this scheme. Although both CLR and RUN clear out the simple variables, the arrays and strings are left untouched. Therefore, before one can store a new element in the main string, a possible previous one must be cleared out. If the two previous statements were renumbered so as to be subroutines, then T$ could be set equal to twenty spaces first, GOSUB the storage routine, then store the actual value of the element. The example program should demonstrate this.

When printing the string, any character less than twenty characters will have trailing spaces. This could be corrected by a routine that strips off these spaces, but it might have a hard time if multiple spaces could be present in an actual element. Therefore, it might be advantageous to keep track of the length of each element so that only the proper number of characters would be printed. The length of each element in the array could be stored in a numeric array, say L. Therefore, the mini-subroutine to store a string element would be:

```
20000 L = LEN(T$): IF L>MAXLEN THEN L = MAXLEN
20010 L(E) = L : START = (E - 1)*MAXLEN
20020 A$(START, START + L - 1) = T$: RETURN
```

This subroutine requires that A$ and L be properly DIMensioned at the start of the program, MAXLEN equals the maximum length of each element, T$ contains the string to be stored, and E contains the number of which element T$ will become.

To recall an element, set E equal to the element number, and call the following routine. MAXLEN must be predefined (see above) and the strings and array L DIMensioned.

```
3000 START = (E - 1)*MAXLEN:T$ = A$(START,START +
  L(E) - 1):RETURN
```

To sum up, I have presented here a technique to simulate the use of string arrays. Undoubtedly it is not the only way, so be creative! Nevertheless, the two subroutines can be the foundation for programs of increased sophistication, as string arrays add a new dimension to programming efforts! ©

Atari Super Breakout

Robert Baker

If you own an Atari 800 computer don't forget there are other game cartridges besides STAR RAIDERS available. My favorite game (next to STAR RAIDERS of course!) is SUPER BREAKOUT. If you've visited any game arcades recently you may have seen an Atari Super Breakout machine. Well the SUPER BREAKOUT cartridge provides the same arcade games for your Atari 800 as well as standard BREAKOUT.

The cartridge provides four games: regular BREAKOUT, Progressive, Double, and Cavity. Each game can be played by one to eight players if you have enough paddle controllers. As expected, the major idea is to knock bricks out of a wall and score points determined by the brick colors.

Regular BREAKOUT is just like the Breakout game contained in the familiar Atari Pinball Breakaway TV game. You try to knock out all the bricks from 8 rows in the wall with five balls. If you knock out all the bricks, a new wall of bricks will appear. There is no limit on the number of times a new wall of bricks can be reset during a game.

With **Progressive**, the setup is somewhat different from regular BREAKOUT. When the game begins there are four rows of bricks at the top of the screen, followed by four blank rows, and then four rows of bricks. After game play begins, the brick walls move down toward the bottom of the screen at a rate determined by the number of times the ball is hit. At the same time, new rows of bricks enter the top of the screen at a progressively faster rate. This game will continue forever if you're good enough!

The **Double playfield** is the same as for regular Breakout except that there are two paddles and two balls are served. The paddles are stacked one on top of the other. The point values are the same as for other games except that each brick is worth double the normal amount whenever two balls are in play. In this game, the wall can only be reset twice after the initial wall is knocked out.

For **Cavity,** the playfield is just like regular Breakout except there are two "cavities" and each contains a ball. When the game begins, the balls bounce inside each cavity but are held captive. When enough bricks are removed to release a captive ball, that ball then enters play and starts to knock out bricks and score points. Point values are double with two balls in play or triple with three balls in play. You do get two paddles, as in Double, and the wall will only be reset twice.

There are no options to vary the number of balls or the paddle size like on the Pinball-Breakaway TV game. However, there is an option to suspend play and another to obtain an additional five serves if desired. After each game, scores are rated from "OOPS" to "BEST". Watch out—these games can be almost as addictive as STAR RAIDERS. They're also great party games since more than one person can play.

More Games For The Atari 400/800

Once you have an Atari 410 cassette drive for the Atari computer systems, there are a number of programs available from various sources. If you're interested in games then you might want to take a look at Mountain Shoot by **Adventure International.** This is a two player game where players take turns shooting cannons at each other. You enter the angle (0 to 90 degrees) to position the gun and the amount of powder (4 to 10) to be used. You even have to watch the wind direction and velocity that is always changing. Each time the game is run, a different terrain is generated between the two gun positions to make things even more challenging.

There are three levels of play: easy, hard, and very hard. The harder the level, the more accurate your shot must be to score a hit. In each level you can play with or without a powder limit. When playing without limits, the game ends when the first player hits his opponent. With a powder limit, you continue playing until *both* players have exhausted their powder. The first player to use up all their powder has to sit by and watch their opponent fire away at him. The player with the most hits, when both players run out of powder, is the winner.

When loading from tape, you first load a brief set of instructions. These remain on the television screen while loading the actual game program. While the program is loading don't forget to turn up the sound on the television. Otherwise you'll miss additional loading precautions and game instructions that are given audibly while the 1812 Overture plays in the background. The program takes full advantage of the two channel cassette drive of the Atari system. It sure beats sitting around doing nothing waiting for a program to load!

There are no printed instructions provided with the game but once played, they'd be unnecessary anyway. The graphics used by the game are not overly attractive. However, you do get sound effects and even a fanfare when you score a hit. The game does not check for a correct angle being entered; it will accept any positive number. If you're not careful you can destroy your own gun and score an additional hit for your opponent. On the other hand, the amount of powder is limited to 10 maximum even if you enter a larger number. All in all, this is a fairly good two player game for the Atari and it's easy enough for young children to play. It's not as elaborate as some of the games from Atari but it can be fun to play.

Thesis (P.O. Box 147, Garden City, MI 48135) offers a number of programs on tape for the Atari computers at $15 each. Their Casino I tape contains two games: BLACKJACK and SLOT MACHINE. Brief printed documentation is included, sufficient for these type of games.

The BLACKJACK game accommodates one to four players and uses full casino rules. It allows you to hit or stand, split pairs, or even double down. The game provides unique graphics that show the dealer as he shuffles and deals the cards each time the deck is exhausted. However, cards are only shown as a face value and a suit symbol. No graphics are used for the cards.

The game does check betting limits, but allows any fraction or full floating point number ($1.23456) to be entered as a bet. If a value entered has a number of digits past the decimal point, it does not get cleared right away and messes up the display slightly. Eventually, however, everything does get corrected. I guess if you enter a value less than a cent, then you deserve to get strange results.

BLACKJACK is a 16K program, so it does take a little while to load from tape. Playing the game is pretty straight-forward. You only enter single letter responses to select your desired action. What else can I say, It's a standard Blackjack game with reasonable graphics and sound.

The SLOT MACHINE game, that comes with BLACKJACK, shows a graphic representation of a slot machine (with parts of other machines on either side). The yellow START button on the ATARI is used to pull and release the handle of the slot machine. Some kind of "noise" is heard as the "wheels" spin and the game makes any appropriate pay-offs depending on the odds. On larger wins, you even get to see coins come out of a slot at the bottom of the slot machine. Since only three symbols (cherry, lemon, & gold bar) are used on the "wheels", there seem to be a fair number of pay offs or wins. This program is not quite as good as some I've seen for other machines but it is still interesting.

The CRIBBAGE game from THESIS is a 24K program and does take a while to load from tape. The game is a standard game of Cribbage, you against the computer. Little graphics are used other than for the cribbage peg board used for scoring. Cards are shown only as a face value and a suit symbol. Printed documentation is provided and includes brief rules on cribbage, as well as directions for using the program and how it works. The game offers two levels of play: beginner and intermediate. The beginner level does not penalize for any errors during play or scoring while the intermediate level will. However, the computer plays the same strategy at both levels of play.

If you've played cribbage, then you know the various steps in each hand during the game. In part of each hand, you alternate playing cards and scoring points, then later score points on your hand for various combinations (runs, flushes, pairs, etc.). In this computerized version, you must specify the card to be played, the current card count, and any score that you claim for your card played. The computer doesn't do any of the work for you. Again, when you later score your hand you also have to enter your own claimed score. Part of this is because of the intermediate level of play where you can be penalized for missing a possible score.

This game could be very good except for one major disadvantage. The author chose to use a very poor method of selecting the card suit when the user must specify his card. Instead of using an easy to remember key letter (like C for Clubs, H for Hearts, etc.) you must enter the actual graphic symbol for the suit. Thus, you have to remember that Control P is used for Clubs, Control comma is Hearts, Control period is Diamonds, and Control semicolon is Spades. Of course none of these are marked on the keyboard so you have to memorize them or have a reference card handy. Otherwise the game is rather good but slow playing. ©

Speeding Up The Player-Missile Demo

Larry Isaacs, Raleigh, NC

Chris Crawford's article[1] on the inner workings of some of the player-missile graphics was very interesting. I'm sure all those who tried the demo program noticed the difference in speed between horizontal movement and vertical movement. This provides a very good example of the difference between the execution speed of machine language and the execution speed of BASIC.

Horizontal movement of the "player" requires only a POKE statement. The function of the POKE statement is roughly equivalent to a single machine language instruction. This allows it to execute fairly fast. On the other hand, vertical movement of the "player" isn't nearly as simple. A FOR...NEXT statement is needed to move some data in memory. Also, vertical movement requires a more complex POKE statement. This POKE statement not only takes longer to execute, but is executed 7 times. These factors result in noticeably slower vertical movement than horizontal movement. If this FOR...NEXT loop could be replaced with some machine language, vertical movement could be brought to seemingly the same speed as horizontal movement.

The program in Listing 1 will illustrate this point. This program is a duplication of Chris Crawford's original demo with modifications for upward movement of the "player" to be done with the aid of machine language. Downward movement is still done with the FOR...NEXT loop to give a comparison. With this program you will find that it takes around three seconds to move the "player" from the bottom of the screen to the top. It will take around 17 seconds to move the "player" back to the bottom again. If the downward movement is too slow to bear, use the program in Listing 2. It has machine language for both upward and downward movement.

The machine language routines do not contain any absolute addressing, so they are relocateable. This means you can further modify the demo programs and the routines will still work. For those familiar with assembly language, here is the code for the two routines.

```
UP      PLA                 DOWN    PLA
        PLA                         PLA
        STA $CC                     STA $CC
        PLA                         PLA
        STA $CB                     STA $CB
        LDY #$01                    LDY #$06
UPLOOP  LDA ($CB),Y         DNLOOP  LDA ($CB),Y
        DEY                         INY
        STA ($CB),Y                 STA ($CB),Y
        INY                         DEY
        INY                         DEY
        CPY #$07                    CPY #$FF
        BNE UPLOOP                  BNE DNLOOP
        RTS                         RTS
```

As illustrated by this example, where the speed of an operation is concerned, it is faster to use machine language. However, it may not always be better to use machine language, and using it probably won't be easier than using BASIC.

Listing 1

```
1 GOSUB 1000:REM Load machine code
10 SETCOLOR 2,0,0:X=120:Y=48:REM Set bac
kground color and player position
20 A=PEEK(106)-8:POKE 54279,A:PMBASE=256
*A:REM Set player-missile address
30 POKE 559,46:POKE 53277,3:REM Enable P
M graphics with 2-line resolution
40 POKE 53248,X:REM Set horizontal posit
ion
50 FOR I=PMBASE+512 TO PMBASE+640:POKE I
,0:NEXT I:REM Clear out player first
60 POKE 704,216:REM Set color to green
70 FOR I=PMBASE+512+Y TO PMBASE+516+Y:RE
AD A:POKE I,A:NEXT I:REM Draw player
```

```
80 DATA 153,189,255,189,153
90 REM Now comes the motion routine
100 A=STICK(0):IF A=15 THEN GOTO 100
110 IF A=11 THEN X=X-1:POKE 53248,X
120 IF A=7 THEN X=X+1:POKE 53248,X
130 IF A=13 THEN FOR I=6 TO 0 STEP -1:PO
KE PMBASE+512+Y+I,PEEK(PMBASE+511+Y+I):N
EXT I:Y=Y+1
140 IF A=14 THEN D=USR(UP,PMBASE+511+Y):
Y=Y-1
150 GOTO 100
1000 DIM UPCODE$(22):UP=ADR(UPCODE$)
1010 FOR I=1 TO 5:READ A:NEXT I:REM Skip
 player data
1020 FOR I=UP TO UP+20
1030 READ BYTE:POKE I,BYTE
1040 NEXT I:RESTORE :RETURN
1050 REM Move player up code
1060 DATA 104,104,133,204,104,133,203
1070 DATA 160,1,177,203,136,145,203
1080 DATA 200,200,192,7,208,245,96
```

Listing 2

```
1 GOSUB 1000:GOSUB 1100:REM Load machine
 code
10 SETCOLOR 2,0,0:X=120:Y=48:REM Set bac
kground color and player position
20 A=PEEK(106)-8:POKE 54279,A:PMBASE=256
*A:REM Set player-missile address
30 POKE 559,46:POKE 53277,3:REM Enable P
M graphics with 2-line resolution
40 POKE 53248,X:REM Set horizontal posit
ion
50 FOR I=PMBASE+512 TO PMBASE+640:POKE I
,0:NEXT I:REM Clear out player first
60 POKE 704,216:REM Set color to green
70 FOR I=PMBASE+512+Y TO PMBASE+516+Y:RE
AD A:POKE I,A:NEXT I:REM Draw player
80 DATA 153,189,255,189,153
90 REM Now comes the motion routine
100 A=STICK(0):IF A=15 THEN GOTO 100
110 IF A=11 THEN X=X-1:POKE 53248,X:GOTO
 100
120 IF A=7 THEN X=X+1:POKE 53248,X:GOTO
100
130 IF A=13 THEN D=USR(DOWN,PMBASE+511+Y
):Y=Y+1:GOTO 100
140 IF A=14 THEN D=USR(UP,PMBASE+511+Y):
Y=Y-1
150 GOTO 100
1000 DIM UPCODE$(22):UP=ADR(UPCODE$)
1010 FOR I=1 TO 5:READ A:NEXT I:REM Skip
 player data
1020 FOR I=UP TO UP+20
1030 READ BYTE:POKE I,BYTE
1040 NEXT I:RETURN
1050 REM Move player up code
1060 DATA 104,104,133,204,104,133,203
1070 DATA 160,1,177,203,136,145,203
1080 DATA 200,200,192,7,208,245,96
1100 DIM DOWNCODE$(22):DOWN=ADR(DOWNCODE
$)
1120 FOR I=DOWN TO DOWN+20
1130 READ BYTE:POKE I,BYTE
1140 NEXT I:RESTORE :RETURN
1150 REM Move player down code
1160 DATA 104,104,133,204,104,133,203
1170 DATA 160,6,177,203,200,145,203
1180 DATA 136,136,192,255,208,245,96
```

©

THE OSI® GAZETTE

OSI C1P Control Functions

C. A. Stewart
Adrian, MI

In a previous issue of **COMPUTE!** a basic poke version of my control function was published. Since that time I have discovered a method to implement a RUN command with a single key stroke much like the PET run key.

The main routine resides in page 2 in this revision (I used page 0 in my previous version but the added functions required relocation) and the one key screen clear resides in page 0. Refer to figure #1 (Flow chart) for discussion of the program functions.

In normal operations locations #536 and #537 contain vectors set by system ROM to the input routine $FFBA in typical 6502 hi/lo order i.e. $BA in location #536 and $FF in location #537. (Note for new computerist the symbol $ in machine language signifies HEX number, not string and the symbol # signifies a decimal number). By changing the vectors in these locations we force the system into our routine first and then return control to the ROM, to implement in this example we poke #536 with #128 and #537 with #002 (POKE536,128:POKE537,2) in one command line! It should be noted that a break warm start will require this poke command line since a warm start re-initializes these vectors.

A useful basic program for HEX to DEC and DEC to HEX is included in listing #3 for readers without tables or a TI HEX calculator.

When the routine starts we go to the input subroutine $0280 which jumps to $FFBA (input a character) and compare to the following.

Control L	**Load command**
Control S	**Save command**
Control A	**Run Command**
Escape Key	**List command**
Rubout Key	**Screen Clear**

If any of the comparisons are true then the appropriate subroutine in ROM is called, otherwise normal program operation continues. I chose Control A for the Run function for two reasons. First the logical choice, Control r, is utilized for a remove in the cursor control package I have in ROM and because of its location next to the control key. The command keys can be changed to whatever the user requires by replacing the compare data with the appropriate key numbers. Control A = $01 and follows thru with control Z = #26. (see graphics manual.)

Listing #1 is the machine language routine. Listing #2 is the BASIC poke program. The machine language screen clear is callable in BASIC via the USR function. To use load and run, code erases itself, leaving the machine code in page 0 and 2, and doesn't require any normal usable memory.

```
D8              PHA
D9              LDA #$20;LOAD SPACE CHR
DB              LDX #$00;LOAD ACCX W/0
DD   HERE       STA $D000,X;STORE SPACE CH
R ON SCREEN
E0              STA $D100,X
E3              STA $D200,X
E6              STA $D300,X

E9              INX         ;INC X
EA              BNE HERE    ;BRANCH TO $DD I
F NOT EQUAL TO 0
EC              PLA
ED              RTS
```

```
280         JSR $FFBA;JUMP TO INPUT SUBRO
UTINE $FEED ON C4P
283         CMP #$0C ;COMPARE TO CONT L
285         BNE CONT S;BRANCH TO CONTROL
S
287         JSR $FF8B ;EXCUTE LOAD COMMAN
D
28A CONT S  CMP #$13  ;COMPARE TO CONTROL
 S
28C         BNE RUB   ;
28E         JSR $FF96 ;EXCUTE SAVE COMMAN
D
291 RUB     CMP #$7F  ;COMPARE TO RUBOUT
293         BNE ESC   ;
295         JMP $D8   ;EXCUTE SCREEN CLEA
R
298 ESC     CMP #$1B  ;COMPARE W/ESCAPE
29A         BNE RUN   ;
29C         JMP $A4B5 ;EXCUTE LIST
29F RUN     CMP #$01  ;COMPARE W/CONT A
```

OSI

SOFTWARE FOR OSI

OSI

A JOURNAL FOR OSI USERS!!

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

1) Using String Variables.

2) High Speed Basic On An OSI.

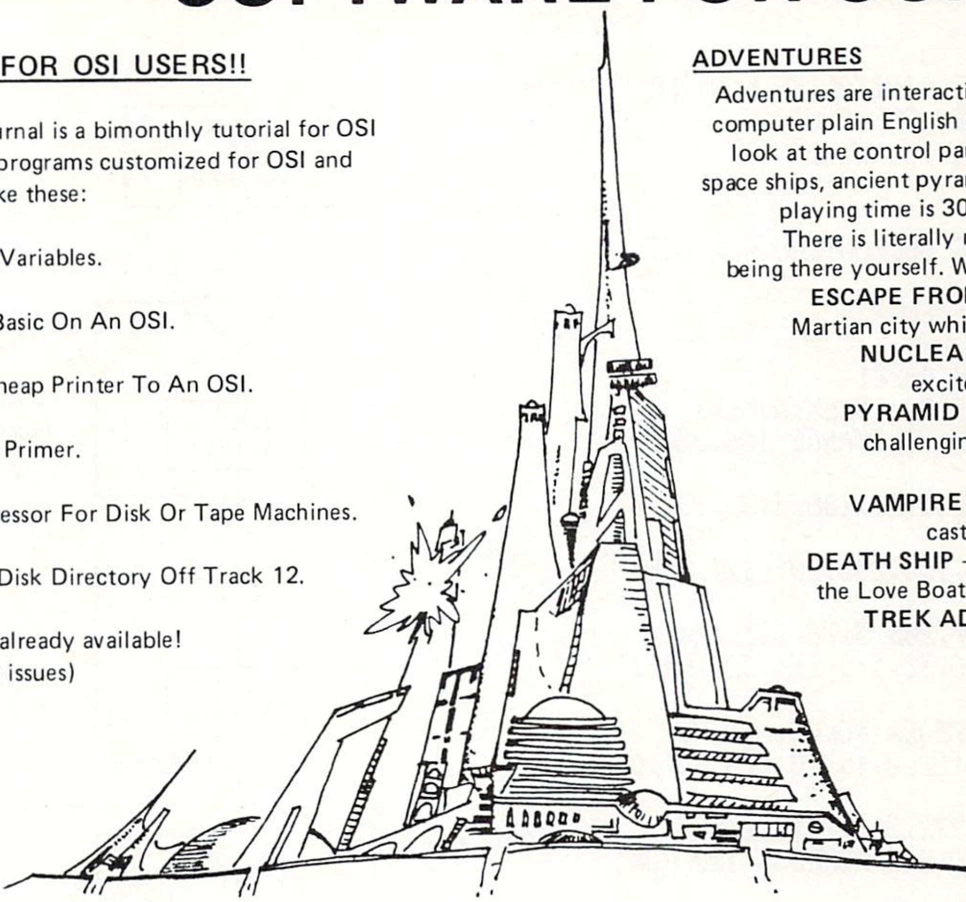
3) Hooking a Cheap Printer To An OSI.

4) An OSI Disk Primer.

5) A Word Processor For Disk Or Tape Machines.

6) Moving The Disk Directory Off Track 12.

Four back issues already available!
$9.00 per year (6 issues)



ADVENTURES

Adventures are interactive fantasies where you give the computer plain English commands (i.e. take the sword, look at the control panel.) as you explore alien cities, space ships, ancient pyramids and sunken subs. Average playing time is 30 to 40 hours in several sessions. There is literally nothing else like them – except being there yourself. We have six adventures available.

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

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

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

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

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

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

$14.95 each

NEW SUPPORT ROMS FOR BASIC IN ROM MACHINES

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

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

DISK UTILITIES

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

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

P.C. BOARDS

MEMORY BOARDS!! – for the C1P. – and they contain parallel ports!
Aardvarks new memory board supports 8K of 2114's and has provision for a PIA to give a parallel ports! It sells as a bare board for $29.95. When assembled, the board plugs into the expansion connector on the 600 board. Available now!

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

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

ARCADE AND VIDEO GAMES

ALIEN INVADERS with machine code moves – for fast action. This is our best invaders yet. The disk version is so fast that we had to add selectable speeds to make it playable.
Tape - $10.95 – Disk - $12.95

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

STARFIGHTER – a real time space war where you face cruisers, battleships and fighters using a variety of weapons. Your screen contains working instrumentation and a real time display of the alien ships. $6.95 in black and white - $7.95 in color and sound.

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

SCREEN EDITORS

These programs all allow the editing of basic lines. All assume that you are using the standard OSI video display and polled keyboard.

C1P CURSOR CONTROL – A program that uses no RAM normally available to the system. (We hid it in unused space on page 2). It provides real backspace, insert, delete and replace functions and an optional instant screen clear.
$11.95

C2/4 CURSOR. This one uses 366 BYTES of RAM to provide a full screen editor. Edit and change lines on any part of the screen. (Basic in ROM systems only.)

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

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

AARDVARK IS NOW AN OSI DEALER!

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

–THIS MONTH'S SPECIALS–

Superboard II	$279
C1P Model II	429
C4P	749

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

Video Modification Plans and P.C. Boards for C1P as low as $4.95

VISA

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

master charge

OSI

Aardvark Technical Services • 1690 Bolton • Walled Lake, MI 48088
(313) 669-3110

OSI

```
2A1          BNE END    ;
2A3          JSR $A477 ;INITALIZE ROUTINE
2A6          JSR $A5C2 ;RUN ROUTINE
2A9 END      RTS        ;
```

Listing 1

```
 10 REM MACHINE LANG SUBROUTINE FOR OSI
C1P/C4P
 20 REM CHARLES A. STEWART
 30 REM 3033 MARVIN DR.
 40 REM ADRIAN, MICH 49221
 60 FORX=640TO681:READA:POKEX,A:NEXT
 65 REM FOR C4P LINE 70 CHANGE 186,255 T
O 237,254
 70 DATA32,186,255,201,12,208,3,32,139,2
55,201
 80 DATA19,208,3,32,150,255,201,127,208,
3,76
 90 DATA216,0,201,27,208,3,76,181,164
 100 DATA201,1,208,6,32,119,164,32,194,1
65,96
 110 FORX=216TO237:READA:POKEX,A:NEXT
 120 DATA72,169,32,162,0,157,0,208,157,0
,209,157,0,210
 130 DATA157,0,211,232,208,241,104,96
 150 POKE11,216:POKE12,0:POKE536,128:POK
E537,2
 160 PRINT"*CONTROL VERSION #1":PRINT"*B
Y CHARLES A. STEWART"
 165 PRINT:PRINT
 170 PRINT"ESC LISTS":PRINT"RUBOUT GIVES
 SCREEN CLEAR
 180 PRINT"CONTROL S =SAVE":PRINT"CONTRO
L L = LOAD
 185 PRINT"CONTROL A RUNS PROGRAM
 200 NEW
```

Listing 2

```
 10 REM CHARLES A. STEWART
 20 REM 3033 MARVIN DR.
 30 REM ADRIAN MI 49221
 40 REM 517-265-4798
 50 REM NOVEMBER 22, 1980
 60 REM DEC TO HEX AND HEX TO DEC CONVER
SON PROGRAM
 100 DIMA$(16),S$(16):FORX=1TO16:READA$(
X):READS$(X):NEXT
 110 DATA0000,0,0001,1,0010,2,0011,3,010
0,4,0101,5,0110,6
 120 DATA0111,7,1000,8,1001,9,1010,A,101
1,B,1100,C,1101,D,1110,E
 130 DATA1111,F
 135 S$="0123456789ABCDEF"
 140 POKE11,0:POKE12,253:X=0:Y=0:W=0:Q=0
:I=0:E$="0"
 150 FORX=0TO40:PRINT:NEXT:PRINT"A> DECI
MAL TO HEX":PRINT
 160 PRINT"B> HEX TO DECIMAL":PRINT:PRIN
```

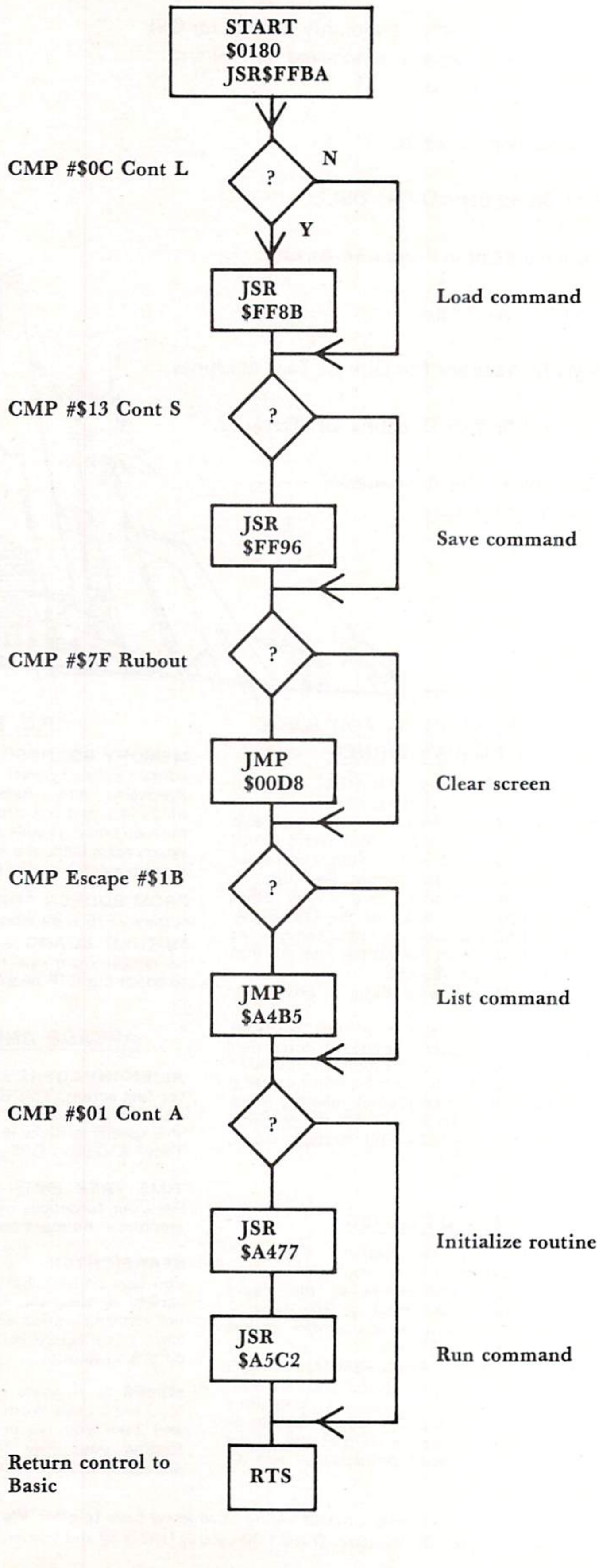


Figure #1

```
T"YOUR SELECTION":X=USR(X)
 170 IFPEEK(531)=65THENPRINT"DEC TO HEX
CONVERSION":GOTO3010
 180 IFPEEK(531)=66THENPRINT"HEX TO DEC
CONVERSION":GOTO2010
 190 GOTO150
 2010 PRINT:INPUT"HEX NUMBER";I$:IFLEN(I
$)>4THEN2010
 2020 IFLEN(I$)<>4THENI$=E$+I$:GOTO2020
 2040 FORX=1TO4:FORY=1TO16
 2050 IFMID$(I$,X,1)=MID$(S$,Y,1)THENB$(
X)=A$(Y)
 2060 NEXTY:NEXTX
 2070 B1$=B$(1)+B$(2)+B$(3)+B$(4)
 2080 PRINT:PRINTI$" IN BINARY=":PRINTB1
$
 2100 X=1:W=0:Q=LEN(B1$):I=0
 2120 Y$=MID$(B1$,Q,1):Y=VAL(Y$):I=Y*X:W
=W+I:X=X*2
 2130 Q=Q-1:IFQ<>0GOTO2120
 2140 PRINT:PRINTI$" IN DECIMAL=";W
 2150 PRINT:PRINT"TYPE ANY KEY TO CONTIN
UE":X=USR(X):GOTO2010
 3010 PRINT:INPUT"DECIMAL NUMBER";I$:I=V
AL(I$):Y$=" ":Y=65536
 3012 Y=Y/2
 3015 IFI>65535THENPRINT:PRINT"TOO LARGE
":GOTO2150
 3030 X=INT(I/Y):IFX=0THENY$=Y$+"0":GOTO
3050
 3040 Y$=Y$+"1":I=I-Y
 3050 Y=Y/2:IFINT(Y)=0THEN3200
 3060 GOTO3030
 3200 PRINT:PRINTI$"IN BINARY=":PRINTY$
 3210 X=2:Y=4
 3215 RE$=" "
 3220 A$=MID$(Y$,X,Y):FORW=1TO16:IFA$=A$
(W)THENRE$=RE$+S$(W):GOTO3240
 3230 NEXTW
 3240 X=X+4:IFX>14THENGOTO3260
 3250 GOTO3220
 3260 PRINT:PRINTI$" IN HEX=";RE$
 3265 PRINT"TYPE ANY KEY TO CONTINUE":X=
USR(X):GOTO3010
 OK
```

Listing 3

©

Double The Cassette Baud Rate Of Your OSI Superboard II/1P

Mr. James L. Mason
Jacobus, Pennsylvania

I was thrilled when I received my Superboard II. It was my first micro, but being experienced in BASIC Programming (using a phone-linked ASR 33 with GE time-sharing) I had several programs which I was anxious to try. The Superboard performed perfectly. It did everything OSI said it would, however, 2 disadvantages of Superboard soon made themselves apparent. The 25 x 25 character video format was not the easiest to read. Secondly, the baud rate at which programs are saved and loaded from cassette seemed painfully slow. Having a good working background in digital electronics, I thought it might be possible to improve upon these two features. Upon close examination I found the video hardware was too intimate with the software in ROM. Fortunately, modifying the cassette port circuitry was a piece of cake and I was able to cut load and save time by half.

The cassette port utilizes a 6850 programable **A**synchronous **C**ommunications **I**nterface **A**dapter (see figure 1). When using this chip, the communications rate is determined by two things, the frequency of the clock which is applied to the TXCLK and RXCLK pins of the ACIA and the control word which is written into the ACIA's control register. I hypothesied that by doubling the clock frequency I could double the baud rate.

ON the Superboard, a crystal oscillator generates the base timing signal by which the entire board is controlled. This signal drives a synchronous divider chain (see figure 2). The timing signal destined for the ACIA comes off the ÷ 32 tap of this chain. The signal is finally divided by a ÷ 24 circuit composed of U57, U58, and U63. The resultant frequency of 5120 Hz is applied to the TXCLK input of the ACIA. The ACIA must be programmed to utilize a clock frequency either 64, 16, or 1 times the baud rate. The 320 baud rate is realized by programming the ACIA for a 16x clock rate.

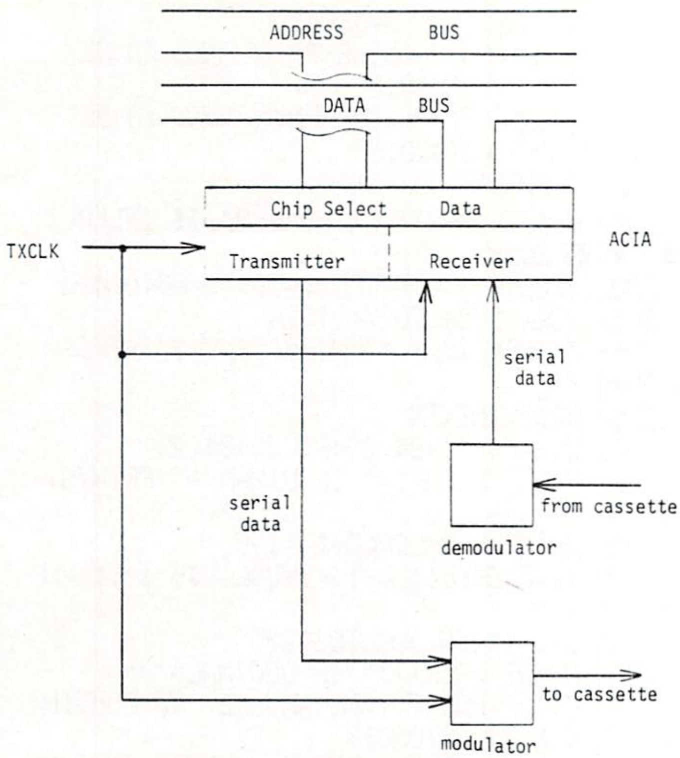


Figure 1. Cassette Port Block Diagram

To obtain a clock rate double of that which is used, I chose to sever the connection between U57 pin 2 and U59 pin 14 (see figure 3), then connect U57 pin 2 to U30 pin 11. I used a switch to maintain compatibility with my old 320 baud tapes.

As far as the ACIA was concerned, the modification was done. However, there is one more block between the ACIA and the cassette machine, namely the Modulator/Demodulator.

The modulator encodes the data in the form of tones. These tones are derived from the TXCLK (see Figure 4A). Since our new TXCLK is twice as fast, our tones will now be 2 times their original frequency. This poses no problem as far as modulation is concerned. It does, however, make a difference on the return trip. U69 determines what will be demodulated as a high or low tone (see figure 4B). A tone coming in will trigger the 74123 one-shot by it's rising edge (see figure 5). R57 is adjusted so that U69 will remain triggered until after the falling edge of the high frequency tone but not until the falling edge of a low frequency tone. The falling edge will clock the D flop U63 and propagate the state of U69. Because we now have shorter pulse widths, R57 must be adjusted to allow U69 to time-out during our new low tones. This was very simple to do. I simply saved a program using the new faster baud rate and attempted to load it back. While the program was trying to load, I adjusted R57 while watching the video monitor. I knew I had R57 adjusted properly when the program began appearing on the display, line by line. I experimented with R57 to find the points where data started to be garbled. The margin was surprisingly wide. Luckily, no software patch had to be made anywhere.

If you use a switch in your mod, remember you will have to readjust R57 each time you change baud rates. I see no reason why a DPDT switch couldn't be used to switch in a different resistor value for R57 along with switching the clock rate.

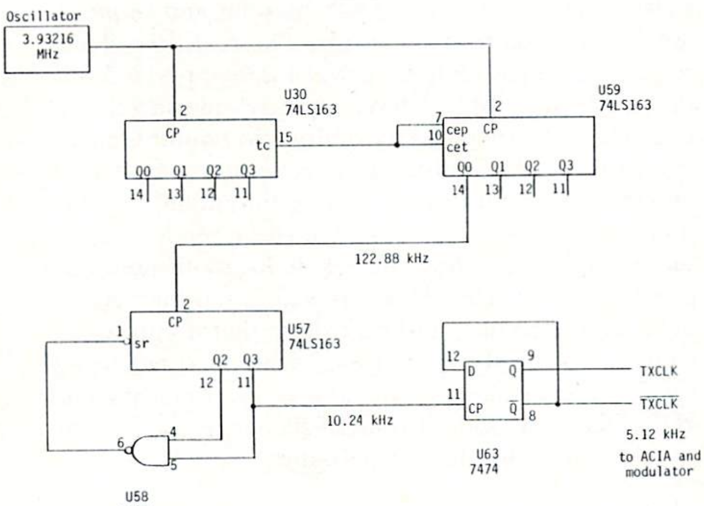


Figure 2. TXCLK Generation

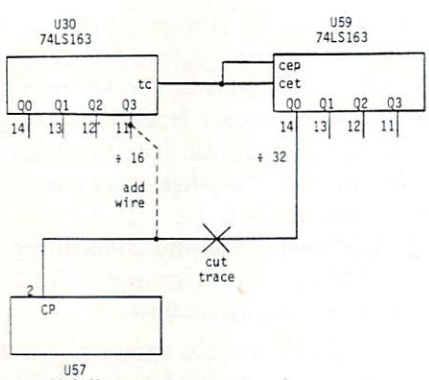


Figure 3. Installing The Modification

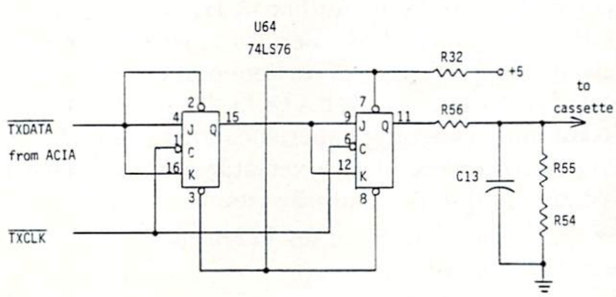


Figure 4A. Modulator

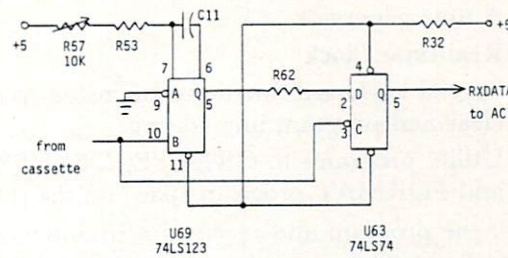


Figure 4B. Demodulator

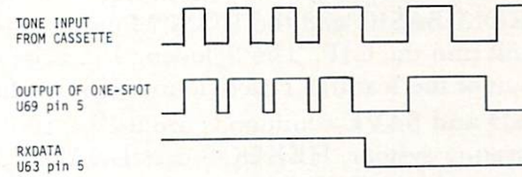


Figure 5. Demodulator Timing

©

Review

HEXDOS 2.3: A Disk Operating System For The OSI C1P or Superboard II

Ronald C. Whitaker
Salt Lake City, Utah

The day I received my OSI disk drive and 610 Expander board, I hooked them to my C1P and my homebuilt power supply, turned them on, and pushed "D" to boot up the disk. OSI's Pico-Dos came with the drive and expander board and booted up OK but would only allow me to load eight programs of up to 8K each. This was faster than cassette to be sure, but definitely lacking the features I wanted in a disk operating system. I lacked funds for OSI's OS65D and the additional 4K of memory it required to run on my system. The future looked dismal, indeed!

The day was saved by a single stroke of good fortune. Several months earlier a local dealer had loaned me a catalog from "The 6502 Program Exchange". While oriented mostly toward Apple and single board systems, they did have a few OSI compatible programs listed. One of these was HEXDOS 2.3 for the C1P and Superboard II by Steve Hendrix. The features promised by the catalog sounded too good to be true! These included:

1. An operating system and directory which occupied only the first two tracks on the disk

2. A tone generator
3. Real-time Clock
4. Special keyboard functions for instant screen clear and program line editing.
5. Utility programs to CREATE, DELETE, and FORMAT program space on the disk

I sent for the program and received it in about 10 days. My first attempts to use it were frustrating. It would function erratically or not at all. But in a quick exchange of correspondance with the author, the problem was resolved, my copy of the program was updated and has functioned flawlessly ever since. It is an exceptionally compact system due to its full use of ROM BASIC and the ROM Monitor software built into the C1P. The following is a brief discussion of the features I have found most useful.

1. **LOAD and SAVE** commands are the heart of any operating system. HEXDOS uses LOAD (filename) and SAVE (filename) to load and save programs by filename. Other commands open and close named data files, each consisting of a 2K block of memory which fills one disk track. Opening a data file reserves a 2K block of RAM for that file. Closing a file loads the contents of that 2K block to one track of the disk. Each file is designated as an input file or as an output file and up to 11 of each are allowed simultaneously, provided there is sufficient RAM memory to support them.

The most versatile and, potentially, the most useful LOAD and SAVE commands allow the user to load 2K bytes from a specified track to any location in RAM memory. The corresponding save command loads any 2K block of memory to be saved to a specified track number. Because of the directness of these commands, care must be used to avoid accidental over-writing of existing data or program memory. Using these direct LOAD and SAVE commands, I have written programs using record and file lengths of my own choosing.

2. **INPUT and PRINT** commands control I/O to or from disk data files, video screen, keyboard, 6850 ACIA, and ports reserved for printer and modem.

3. **SPECIAL KEY FUNCTIONS** make use of unused keys to provide instant screen clear, suspension of output until key is released, break in a BASIC program, and non-destructive forward and backward movement of the cursor for simple line editing. These functions are so useful that they are sorely missed on those rare occasions when I'm writing a program without HEXDOS.

4. **SINGLE STEP and TRACE** functions are added using simple POKE statements, and are very useful for debugging programs.

5. **The USR** function is used to provide several special functions:

a. Control of a tone generator with 256 different tones
b. Input of a character from one of 256 possible input devices
c. Return contents of the real-time clock
d. Direct jump to a machine code routine located anywhere in memory
e. Jump to ROM monitor
f. Jump to the last machine code routine loaded from the disk

Use of the tone generator and the real-time clock requires very simple hardware modifications which are well detailed in the HEXDOS manual and require only a few minutes to perform. The real-time clock requires a single jumper between pads provided on the 610 board. Since I have my C1P interfaced to a General Instrument's Programmable Sound Generator, I have not tried the tone generator, but it requires a single jumper and an audio amplifier. Additional hardware modifications include repositioning the break key to a less vulnerable location, automatic power-on reset (D/C/W/M appear when the computer is turned on), and disk drive motor control, which turns off the drive motor when it is not being used. None of these modifications are necessary for HEXDOS operation but all of them are useful additions to the computer's capabilities.

6. **UTILITY PROGRAMS:**

FORMAT will completely erase a disk and then format it for HEXDOS by loading the operating system onto track 0 and reserving track 1 for the directory.

CREATE names and reserves any number of tracks for use as data files or for storage of machine language programs, which require filenames beginning with $. These can then be loaded directly into any specified location in user RAM.

DELETE erases any program listed in the directory, deallocates the space on the disk reserved for it, and repacks the remaining files.

DISASSEMBLER lists machine code programs using standard 6502 mnemonics and identifies the addressing mode of the instruction.

HEXDOS 2.3 is an extremely compact, easy-to-use, and versatile operating system for the OSI C1P or Superboard II. Because it is so compact it can be placed at the beginning of every disk. Documentation is clear and complete, the best I have seen from any source. Because I knew nothing about data files or their use, I wrote to the author of HEXDOS, Steve Hendrix, for information. Besides a personal reply, I received two new pages of documentation explaining use of disk files. Using HEXDOS 2.3 has been a pleasant and rewarding experience for me and I strongly recommend it as a versatile and inexpensive alternative to other operating systems.

HEXDOS 2.3 is available for $25 from:
The 6502 Program Exchange
2920 West Moana Lane
Reno, NV 89509

©

THE PET® GAZETTE

Partition and Load

R. D. Young
Ottawa, Ontario

And another mystery is solved. This one began where I left off in "Relocate" (**COMPUTE!**, Issue 9, Feb. 1981, p. 103). I think that I have now consolidated Harvey B. Herman's memory partitioning and Charles Brannon's "Quadra-PET" into one routine that makes use of "Relocate" as well. I must acknowledge with gratitude Jim Butterfield's most helpful hints, particularly the various memory maps published over the months, and Harvey B. Herman's memory partitioning challenge in the first place, coupled with his version of "Relocate" that helped to translate for the Upgrade ROM Version.

Note that I have provided two listings to accomodate original and upgrade ROMs. I have not attempted to provide for the 80-column machine. The BASIC program performs two functions: it places a machine language program into the second cassette buffer, and it initializes the partitions and the pointer storage locations (1010 to 1017 decimal). The BASIC program ends with a NEW statement, leaving the PET in the lowest partition which is equivalent to the old standard 8K PET (FRE(0) = 7164).

I chose to use a BASIC program for this loading and initialization after trying several other alternatives. Its main advantage is that the machine language program could be made small enough to fit in the second cassette buffer with the advantages that second cassette buffer use offers. Some problems with loading and initialization were also alleviated. However, because of space restrictions, error checking of user input is not performed. In other words, if you exceed the input limits *** **CRASH** ***.

The machine language program provides control over four (4) memory partitions. It will function with 16K, 24K, or 32K PETs, providing, of course, that the proper inputs are used. That's twice now that I've mentioned inputs. Here is how the routine is used:

1. The BASIC program has left you in Area 1 (low).
2. Use SYS 826 to call the routine. A flashing cursor will appear below the SYS826 on the screen. This is asking you for your input (number 1 to 7) according to the following options:
   - 1 - go to Area 1 (low 7K)
   - 2 - go to Area 2 (next 8K)
   - 3 - go to Area 3 (next 8K)
   - 4 - go to Area 4 (high 8K)
   - 5 - load Area 2
   - 6 - load Area 3
   - 7 - load Area 4
3. To load Area 1, go to Area 1 and LOAD in the normal way.
4. Key in and SAVE programs from Area 1.
5. Clear all partitions by POKE 135,maximum:POKE 123,4:NEW — or upgrade ROM equivalents.

There are a few precautions to be observed. Load/ Verify and device flags are not set by this routine. To load from Tape #1, for example, you must set these flags manually (see "Hooray for SYS", **COMPUTE!**, Issue 8, Jan. 1981, p. 96), or you must have previously loaded a program from Tape #1. This routine assumes that BASIC pointers in locations 122 and 134 (original ROM) are never changed. And lastly, going from one partition to another will result in the loss of your program *if* the program is less than 256 bytes long. Also remember to observe the input limits for the memory size in your machine.

Should you want or need to find out where the partitions are, the pointers to the end of BASIC in each partition restored in locations 1010 to 1017 decimal as follows (original ROM):

1010	124	Area 1
1011	125	Area 1
1012	124	Area 2
1013	125	Area 2
1014	124	Area 3
1015	125	Area 3
1016	124	Area 4
1017	125	Area 4

In fairness to all, I should mention that my PET is equipped with the original ROM only, and has 16K of memory. The routine functions with my PET, but there could still be some bugs remaining for obvious reasons.

I am looking forward to seeing any comments on the routine in **COMPUTE!**. For now, I'm fresh out of mysteries. ©

```
1Ø REM *** PARTITION & LOAD LOADER ***
2Ø REM *** AND INITIALIZATION ***
25 REM *** ORIGINAL ROM VERSION ***
3Ø REM *** BY R.D. YOUNG ***
4Ø REM
5Ø REM
1ØØ REM *** LOAD 2ND CASSETTE BUFFER ***
11Ø PRINT"Ā"
12Ø FOR I=826 TO 990: READ A: POKE I,A: NEXT
13Ø REM
14Ø REM *** INITIALIZATION ***
15Ø POKE 1Ø12,4: POKE 1Ø14,4: POKE 1Ø16,4
16Ø POKE 1Ø13,32: POKE 1Ø15,64: POKE 1Ø17,96
17Ø POKE 135,32: POKE 8192,Ø: POKE 16384,Ø: POKE 24576,Ø
18Ø NEW
19Ø REM
1ØØØ DATA 32,223,241,56,233,48,17Ø,141,241,3,189,215,3,133,123,188
1Ø1Ø DATA 2Ø1,3,32,176,3,165,124,157,241,3,165,125,157,242,3,152
1Ø2Ø DATA 17Ø,189,241,3,133,124,189,242,3,133,125,174,241,3,189,2Ø8
1Ø3Ø DATA 3,133,135,224,4,16,12,165,125,197,123,24Ø,3,76,1Ø6,197
1Ø4Ø DATA 76,83,197,169,147,32,234,227,32,174,245,173,126,2,56,237
1Ø5Ø DATA 124,2,141,241,3,165,123,141,124,2,24,1Ø9,241,3,141,126
1Ø6Ø DATA 2,169,55,141,24,129,141,25,129,169,1,141,13,2,169,13
1Ø7Ø DATA 141,15,2,76,195,243,165,135,2Ø1,32,2Ø8,3,162,1,96,2Ø1
1Ø8Ø DATA 64,2Ø8,3,162,3,96,2Ø1,96,2Ø8,3,162,5,96,162,7,96
1Ø9Ø DATA 1,3,5,7,3,5,7,32,64,96,128,64,96,128,4,32
11ØØ DATA 64,96,32,64,96
```

```
1Ø REM *** PARTITION & LOAD LOADER ***
2Ø REM *** AND INITIALIZATION ***
25 REM *** UPGRADE ROM VERSION ***
3Ø REM *** BY R.D. YOUNG ***
4Ø REM
5Ø REM
1ØØ REM *** LOAD 2ND CASSETTE BUFFER ***
11Ø PRINT"Ā"
12Ø FOR I=826 TO 989: READ A: POKE I,A: NEXT
13Ø REM
14Ø REM *** INITIALIZATION ***
15Ø POKE 1Ø12,4: POKE 1Ø14,4: POKE 1Ø16,4
16Ø POKE 1Ø13,32: POKE 1Ø15,64: POKE 1Ø17,96
17Ø POKE 53,32: POKE 8192,Ø: POKE 16384,Ø: POKE 24576,Ø
18Ø NEW
19Ø REM
1ØØØ DATA 32,2Ø7,255,56,233,48,17Ø,141,241,3,189,214,3,133,41,188
1Ø1Ø DATA 2ØØ,3,32,175,3,165,42,157,241,3,165,43,157,242,3,152
1Ø2Ø DATA 17Ø,189,241,3,133,42,189,242,3,133,43,174,241,3,189,2Ø7
1Ø3Ø DATA 3,133,53,224,4,16,12,165,43,197,41,24Ø,3,76,117,197
1Ø4Ø DATA 76,93,197,169,147,32,216,227,32,166,245,173,126,2,56,237
1Ø5Ø DATA 124,2,141,241,3,165,41,141,124,2,24,1Ø9,241,3,141,126
1Ø6Ø DATA 2,169,55,141,24,129,141,25,129,169,1,133,158,169,13,141
1Ø7Ø DATA 111,2,76,185,243,165,53,2Ø1,32,2Ø8,3,162,1,96,2Ø1,64
1Ø8Ø DATA 2Ø8,3,162,3,96,2Ø1,96,2Ø8,3,162,5,96,162,7,96,1
1Ø9Ø DATA 3,5,7,3,5,7,32,64,96,128,64,96,128,4,32,64
11ØØ DATA 96,32,64,96
```

©

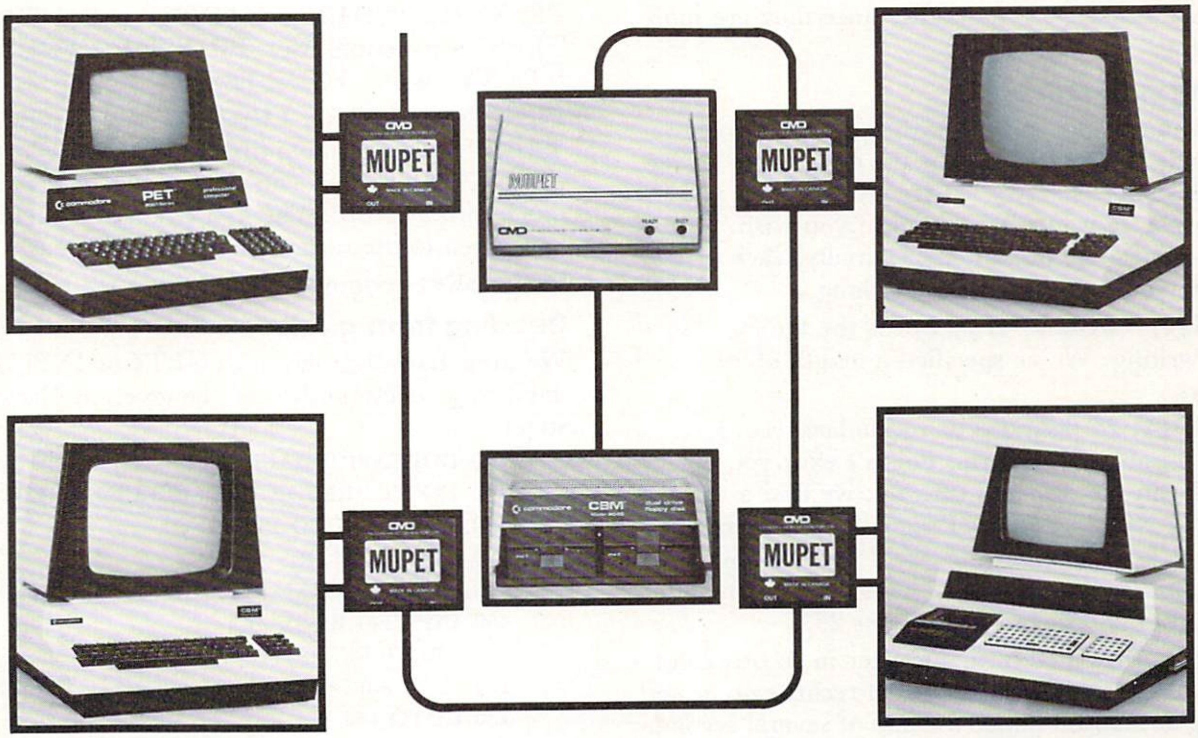
Last Minute Correction From R. D. Young

With reference to my program, "Partition and Load", dated February 15, 1981, there is a correction to be made in the program listings. In the DATA statement at line 1030 of both listings, the '4' between '224' and '16' must be changed to a '5'.

There is also one other precaution to be observed, although it is a function of the imbedded "Relocate" routine. A program loaded with this routine should not have a line 77 as part of the program. Any line 77 will be deleted because of the dynamic RETURN feature used to reset line links after the LOAD.

Now The PET's Know How To Share

MUPET MULTIUSER SYSTEM



EDUCATION

* allows direct access to any programmes or data files
* reduces media cost - one set of disks needed for up to 8 users
* disk based languages such as Assembler, Pascal, Word Processing, available to all users

BUSINESS

* true multi-station wordprocessing with access to common data files
* cost per station drastically reduced
* printer spooling is supported

MANUFACTURING

* using optional DOUBLE MUPET, both local and common IEEE devices are supported
* test equipment can be controlled locally, while reports and programmes are managed universally

MUPET
USES

NO
NONE
NOT ANY
NOT ONE BYTE

OF
SOFTWARE

****

YOU CAN RUN
ANY COMBINATION
OF PROGRAMMES
OR LANGUAGES
AT THE SAME TIME

- word processing
- Pascal
- Assembler
- Future???

* MUPET supports any combination of CBM hardware — present or future — which supports the IEEE bus
* MUPET plugs in externally. No need to open the PET/CBM cabinet
* MUPET supports all existing and future CBM disk based software
* MUPET supports printers as well as disks
* MUPET supports **all** versions of Commodore Basic

PET/CBM Trademark of Commodore Business Machines

CMD

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

Relative File Mechanics

Jim Butterfield, Toronto

I'll use the Basic 4.0 commands, since they are more convenient.

Creating a Relative File

To create a new Relative file, we may use Direct Commands if we wish. This is the easiest way to see how it all happens — try it Directly and later you can incorporate it into a program if you wish. Suppose we want a file of 25 items (initially) each of which may be up to 20 characters long.

DOPEN#1, "RFILE",L20 opens the file for relative writing. We've specified a length of 20 characters.

RECORD#1, 25 positions us to the last record desired, number 25, but this doesn't exist yet. The error light on the disk will turn on; we find a RECORD NOT PRESENT signalled. If we tried INPUT or GET at this time, we'd be in serious trouble. But we can still write, and this will create all records up to and including number 25.

PRINT#1 writes a Return character in record 25 of the file, and incidentally creates all records up to and including 25. You'll notice a delay of several seconds while this happens.

DCLOSE#1 wraps up the file in the usual way.

Writing to a relative file

If you have gone through the DOPEN/RECORD/PRINT/DCLOSE sequence above, we can use this file to write some data. Once again, let's use Direct statements to allow us to watch things happening.

DOPEN#1, "RFILE" opens our file for reading or writing. Note that we don't need to specify the file type (REL) or whether we wish to read or write (we might do either). And we must not specify the length L — that's only for creating the file.

RECORD#1,10 positions us so that we can read or write record number 10. There's no error light this time, since record 10 is in place. Note that if our record number was a variable such as X, we'd need to put it in brackets, e.g., RECORD#1,(X). Now let's write a few records starting at item ten.

PRINT#1, "HELLO" writes six characters (HELLO plus the RETURN character) to record number 10; the disk automatically positions to the start of record 11. This positioning is not triggered by the Return character — it's done by recognizing the end of PET's transmission (technically speaking, the EOI line).

PRINT#1, "THERE"; uses a semicolon to supress Return so only five characters are written to record 11. End of transmission is correctly detected, however, and the disk positions to record 12.

PRINT#1, "A" + 7CHR$(13) + "B" writes a single record, even though there are two Returns. When we are reading, it will take two INPUT# statements to get the information, since INPUT stops on a Return character.

PRINT#1, "THIR" + "TEEN" writes THIRTEEN in the usual way. **PRINT#1, "FOUR"; "TEEN"** writes FOURTEEN.

PRINT#1, "MORE THAN TWENTY CHARACTERS" will generate a disk error, "Overflow in Record", which tells you that you're trying to fit too much into a record. The first twenty characters will be written to the record, the rest discarded.

DCLOSE#1 terminates our writing session.

Reading from a relative file

We must have a program to GET# or INPUT# our data; these commands can't be given as Direct statements.

```
100 DOPEN#1,"RFILE"
110 INPUT "RECORD NUMBER DESIRED";R
120 IF R = 0 GOTO 190
130 RECORD#1,(R)
140 IF DS>0 THEN PRINT DS$:GOTO 110
150 INPUT#1,R$
160 PRINT R$
170 IF ST = 0 GOTO 150
180 GOTO 110
190 DCLOSE#1
```

Use the above program to browse through the items we have just written. You'll find some interesting things. For example, what do you see in records you have never written (say, record 4)? What happens if you try for record number 200? Does it seem to matter if you have written a Return to the record or not?

A new role for ST, the Status Byte

About the ST test in line 170: you may recall that we created a record with more than one item. Record 12 had two items (called "fields") each followed by a Return character. An INPUT# statement will stop on the first Return and would not see the second field.

But ST works in a very useful way: it is set to 64 (end of file) at the end of every record. That means that if we might have more than one field in a record, we can check ST to see if we have read the last one.

One drawback of this mechanism is that you can't use ST to tell you if you are at the end of a relative file; it flags EOF after every record. How do you tell when you're at the end? Use the RECORD command; when you get an error 50, RECORD NOT PRESENT, you'll know you are past the end.

Enlarging a file

Easy. Use RECORD followed by PRINT# in the same manner as when you created the file. The new records will automatically be written up to the record number you have specified.

Conclusion

There are new rules to learn. But Relative files are easy to use, and add power and speed to your programs. ©

Odds and Ends

Charles A. McCarthy
St. Paul MN

Microsoft Basic for the PET, and probably for other 6502 machines, treats multiplication $z = x^*y$ asymmetrically. The execution time depends upon the binary representation of the factor y, and not on x. This can be inferred from studying the coding of the multiplication routine, and is proved by experiment. On my PET, the execution time for one multiplication $z = x^*y$ is about 3840 + 30 A + 230 B microseconds where A is the number of non-zero bits in y and B is the number of non-zero bytes in y.

As a practical consequence, if you know that one of your factors must have only one or two non-zero bytes, it should be placed as y. When $x = 3$ and $y = \pi$, *the execution time is about the same as when* $x = \pi$ and $y = \pi$, and is about 20% longer than when $x = \pi$ and $y = 3$. ©

COPLOT

Anselm Wachtel
Pittsburgh, PA

This program, entitled "COPLOT" incorporates two previously published routines: 1. A plotting routine by J. S. Barney, Pet User Notes, July/August 1978, page 1; 2. An automatic selection of plotting limits routine by Wm. D. Johnston, Microcomputing, May 1980, page 26.

The former raises the normal resolution by a factor of 8 (singe-pixel resolution) in any one direction. I have chosen the vertical (Y) direction for the function values, and both for the positioning of the coordinate leaders.

The latter receives as input the numeric variable limits and the number of divisions which are desired on the axis. It then returns a new rounded-up interval for the limits, as well as the value of the interval in the same units. In the present program, it is used twice, once for each axis.

As in any program of this type, calculation of f(x) can lead to illegal quantities such as division by 0 or log(-x), etc. Since the function is entered (or edited) in line 120 as DEFNA(X) = . . . by the user, there is no way to anticipate which values will cause an error message in the course of scanning from the low to the high X-bounds, and the user has to anticipate this by applying elementary considerations. For instance, if DEFNA(X) = 1/(X*X – 5*X'14) then a ?DIVISION BY 0 ERROR will occur at x = – 2 if the X bounds are, say, – 5 and 3, and at x = 7 if they are 3 and 10. If the function is log of the above, then the entire range from – 2 to + 7 is illegal. The function is evaluated at line 290, and an appropriate line 285 such as IF X*X – 5*X – 14 = 0 THEN NEXTI should take care of that.

Notwithstanding the high resolution which leads to attractive looking displays, the program is not intended to allow an exact evaluation of function values, e.g. replace graph paper ruled in mm or 1/20 inch divisions. Theoretically, the values are located at the *left* ends of the bars and this, moreover, can create the illusion of a displacement in cases where a symmetrical curve is known to peak at some known value, because the viewer (unintentionally) tends to look at the center of the bars. In order to avoid overprinting of labels, the number of divisions are limited to 8 in the abscissa and 10 on the ordinate. At least on the abscissa, anything denser would create confusion even in the case of integer-labels, because these are not in good registry with the high-resolution leaders, especially considering the leading space which the PET places in front of positive numbers, including 0.

All things considered, this program should prove to be a valuable aid to any one desiring a quick sketch of some function within any reasonable limits.

```
0 REM PLOT ROUTINE BASED ON PROGRAM BY ¬
     ¬J.S.BARNEY - SEE PET USER NOTES
10 REM  JULY/AUGUST, 1978, PAGE 1,
     ¬ AND AUTOMATIC SELECTION OF ¬
     ¬PLOTTING LIMITS
20 REM BY WM. D. JOHNSTON, SEE MICROCOMP
     ¬UTING MAY,1980, PAGE 26.
30 REM ADAPTED FOR SCALING OF COORDINATE
     ¬S AND FUNCTIONS BY A.WACHTEL,
40 REM PITTSBURGH, PA.  AUGUST,1980.
50 REM ENTER FUNCTION IN LINE 120. EDIT ¬
     ¬AT END AFTER HITTING 'F'.
60 REM INPUT NEW X-INTERVAL AT END ¬
     ¬AFTER HITTING ANY OTHER KEY.
70 D$="↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓":REM 20 ¬
     ¬CURSOR DOWN.
80 X$="→→→→→→→→→→→→→→→→→→→→→→→→→→→→→→→→"
     ¬:REM 32 CURSOR RIGHT.
90 Y$="↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑":REM 20 ¬
     ¬CURSOR UP.
100 V$="$RF@CDE#":REM SHIFT"$RF@CDE#"
110 U$="%TGB]HY'":REM SHIFT "%TGB]HY'"
120 DEFFNA(X)=SIN(X)
130 DIM X(32),Y(32),S(12)
140 INPUT"ĥLOWER, UPPER X-BOUNDS";A1,B1
150 AX=A1:BX=B1
160 PRINT"↓# OF DIVISIONS ON ABSCISSA,
     ¬ ORDINATE"
170 PRINT"(8 OR LESS ON ABSCISSA,
     ¬ 10 OR LESS"
180 PRINT"ON ORDINATE TO PERMIT NUMERIC ¬
     ¬LABELS)."
190 INPUTLA,LO
200 IFLA<=8ANDLO<=10THEN220
210 PRINT"↓TOO CROWDED FOR NUMERIC ¬
     ¬LABELS!↓":GOTO160
220 PRINT"                    ↓↓↓rWORKING"
230 L=LA:GOSUB530
240 DU=256*C/R:REM SCALING FACTOR FOR ¬
     ¬ABSCISSA DIVISIONS.
250 AD=ABS(A-A1)*256/R
260 AA=A:BA=B:CA=C
270 CF=(B1-A1)/31:REM SCALING FACTOR ¬
     ¬FOR X
280 FOR I=1TO32:X(I)=A1+(I-1)*CF
290 Y(I)=FNA(X(I)):NEXT:REM STORE ¬
     ¬FUNCTION VALUES IN AN ARRAY
300 GOSUB430
310 DF=160/(MAX-MIN):REM SCALING FACTOR ¬
     ¬FOR F(X) ONTO 20*8=160 UNITS.
320 L=LO:A1=MIN:B1=MAX:GOSUB530
330 REM NEW INPUT FOR AUTO.SELECTION OF ¬
     ¬PLOTTING LIMITS SUBROUTINE.
340 DW=160*C/R:REM SCALING FACTOR FOR ¬
     ¬ORDINATE DIVISIONS.
350 OD=ABS(A-A1)*160/R:REM A,A1,
     ¬R ARE DIFFERENT FROM THOSE IN ¬
     ¬LINE 210
360 GOSUB970:REM PRINT COORDINATES.
370 FORI=1TO32:Y1=(Y(I)-MIN)*DF
380 Y%=Y1/8:V%=Y1-8*Y%+1
390 PRINT"h→→→→→→";D$;LEFT$(X$,I);LEFT$(
     ¬Y$,Y%);MID$(V$,V%,1);:NEXT:
     ¬REM PLOT.
400 GETZ$:IFZ$=""THEN400
410 IFZ$="F"THENPRINT"ĥ":LIST120
420 PRINT"ĥ":GOTO140
```

```
430 MAX=Y(1):MIN=Y(1):REM FIND LIMITS ¬
     ¬OF F(X) FOR SCALING
440 FOR I=2TO32:IFY(I)>MAXTHENMAX=Y(I):
     ¬GOTO460
450 IFMIN>Y(I)THENMIN=Y(I)
460 NEXT:RETURN
470 REM LINES 530-910 ARE THE AUTOMATIC ¬
     ¬PLOT LIMIT SELECTION PROGRAM WHICH
480 REM USES: A1,B1=RAW LOW,HIGH LIMITS
490 REM AND  L=NUMBER OF DIVISIONS/AXIS
500 REM AND RETURNS:
510 REM A,B=ROUNDED LOW,HIGH LIMITS ¬
     ¬WHICH ARE >=A1,B1, AND
520 REM C=(ROUND) INCREMENT BETWEEN ¬
     ¬DIVISIONS
530 M=12
540 FORI=1TO12:READS(I):NEXT
550 DATA1,1.5,2,2.5,3,4,5,6,7,8,9,10
560 RESTORE
570 A2=A1:B2=B1:R=B2-A2
580 C2=ABS(R/L):P=LOG(C2)/LOG(10)
590 IFP>0THEN610
600 P=P-1
610 N=SGN(P)*INT(ABS(P))
620 F=C2/10^N
630 FORJ=1TOM
640 IFF>S(J)THEN660
650 C=S(J)*(10^N):K=J:GOTO710
660 NEXTJ
670 PRINT"ABORT--MOST PROBABLE CASE IS ¬
     ¬AN ERROR
680 PRINT"IN THE TABLE OF STANDARD ¬
     ¬INCREMENT MANTISSAS
690 STOP
700 K=K+1:C=S(K)*(10^N)
710 I=A2/C
720 I=SGN(I)*INT(ABS(I)):T=I:A=T*C
730 D=ABS((A-A2)/R)
740 IFA2<0THEN800
750 IFR>0THEN840
760 IFD<=.0001THEN780
770 A=A+C
780 C=-C
790 GOTO840
800 IFR>0THEN820
810 C=-C:GOTO840
820 IFD<=.0001THEN840
830 A=A-C
840 T2=A+(C*L)
850 IFR<0THEN900
860 IFT2>B2THEN910
870 IFABS((B2-T2)/R)<=.0001THEN910
880 IFK<MTHEN700
890 N=N+1:GOTO620
900 IFT2>B2THEN870
910 B=T2:RETURN
920 REM LINES 970-1160 PRINT THE ¬
     ¬LABELED COORDINATES, USING L AND ¬
     ¬THE
930 REM QUANTITIES A,B,C RETURNED BY
940 REM THE PREVIOUS SUBROUTINE. THESE ¬
     ¬WERE OBTAINED TWICE AND THOSE FOR ¬
     ¬THE
950 REM ABSCISSA ARE THEREFORE STORED ¬
     ¬AND SUPPLIED BY THE MAIN PROGRAM
960 REM(LINES 70-420)AS LA,CA,ETC...
970 PRINT"ĥ";:FORI=1TO20:PRINTTAB(7)"%":
     ¬NEXT
980 PRINT"h↓↓↓";TAB(4)"Y";D$;LEFT$(X$,
     ¬18);"X"
990 FORI=0TOL-1
1000 YW=I*DW-OD:YW%=YW/8:W%=YW-8*YW%+1
1010 IFYW%<0THEN1050
1020 IFA+C*I>=A2ANDA+C*I<=B2THEN1040
1030 IFA+C*I>B2THEN1060
1040 PRINT"h";D$;LEFT$(Y$,YW%);TAB(0)A+C
     ¬*I;TAB(7)MID$(V$,W%,1)
1050 NEXT
1060 PRINT"h>>>>>>>";D$;:FORI=1TO32:
     ¬PRINT"$";:NEXT
1070 FORI=0TOLA-1
1080 XU=I*DU-AD:XU%=XU/8:U%=XU-8*XU%+1
1090 IFXU%<0THEN1150
1100 IFAA+CA*I>=AXANDAA+CA*I<=BXTHEN1120
1110 IFAA+CA*I>BXTHEN1160
1120 PRINT"h>>>>>>>";D$;:IFI>0THENPRINT"
     ¬<";
1130 PRINTLEFT$(X$,XU%);MID$(U$,U%,1);
1140 IFAA+CA*I>=AXANDPOS(0)<38THENPRINT"
     ¬↓↓<<<"AA+CA*I;
1150 NEXTI
1160 RETURN
READY.
```

©

ROM Expansion For The Commodore PET

F. Arthur Cochrane
E. I. du Pont de Nemours & Co., Inc.
Savannah River Laboratory
Aiken, SC 29808

This paper was prepared in connection with work under Contract No. DE-AC09-76SR00001 with the U.S. Department of Energy.

Summary

The Commodore PET is a low-cost personal computer housed in a self-contained unit complete with keyboard and a video display screen. The PET is portable and can sit on a desk or tabletop and operates on normal household current. The PET is simply plugged in and turned on to activate its BASIC interpreter. The BASIC interpreter is a computer program that is permanently built into every PET. In contrast, a program loaded from a tape, disk, or the keyboard is placed into Read/Write memory (RAM) and is lost when the PET is turned off.

The PET has three empty sockets for expansion read-only memory units (ROMs) which have memory addresses 36864 ($9000) through 49151 ($BFFF). A method for using single-supply, erasable, programmable, read-only memories (EPROMs) with the Commodore PET microcomputer has been developed. These EPROMs will allow the development of special-purpose firmware for the PET that does not have to be reloaded each time the PET is turned on and will permit an inexperienced operator to use the PET.

Current Firmware For The Pet

There are several ROMs available for the PET:

1. Commodore's WordPro-2 is a 4K ROM for addresses $B000-$BFFF.
2. Commodore's WordPro-3 is a 4K ROM for addresses $A000-$AFFF.
3. The BASIC Programmer's Toolkit™ is a 2K ROM for addresses $B000-$B7FF.
4. The Database Management System (United Software of America, New York, N.Y.) is a 2K ROM for addresses $A000-$A7FF.
5. The Keyed Random Access Method (KRAM 1.0; United Software of America) is a 2K ROM for addresses $A800-$AFFF.
6. The Monjana/1 CBM™ Monitor (Elcomp Publishing, Inc., Chino, Calif.) is a new 2K monitor ROM which is addressed for $9000-$97FF.
7. The Jinsam™ database is a 2K ROM for $9000-$97FF.

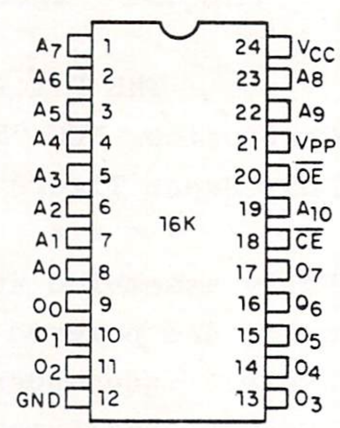
The Spacemaker™ allows two ROMs to be plugged in at the same time; a single-pole switch selects one or the other of the ROMs. The Database Management and KRAM 1.0 ROMs can thus be put on a Spacemaker. Better yet, a small wirewrap board could be built with three 24-pin sockets wired in parallel: the first socket for a ribbon cable input; the second, for the Database ROM; and the third socket for the KRAM ROM. A Spacemaker could then be used with these two ROMs and with another ROM such as the WordPro-3 ROM.

Use Of Eproms

The three empty sockets for ROM expansion are 4K sockets connected as shown in Figure 1.

There are three EPROMs, the Intel™ 2716 (2K, Figure 2), the TI™ 2516 (2K, Figure 2), and

2716



PIN NAMES

A_0-A_{10}	ADDRESSES
CE/PGM	CHIP ENABLE/PROGRAM
OE	OUTPUT ENABLE
O_0-O_7	OUTPUTS

FIGURE 2. Pin Configuration and Pin Names of the Intel 2716 2K EPROM

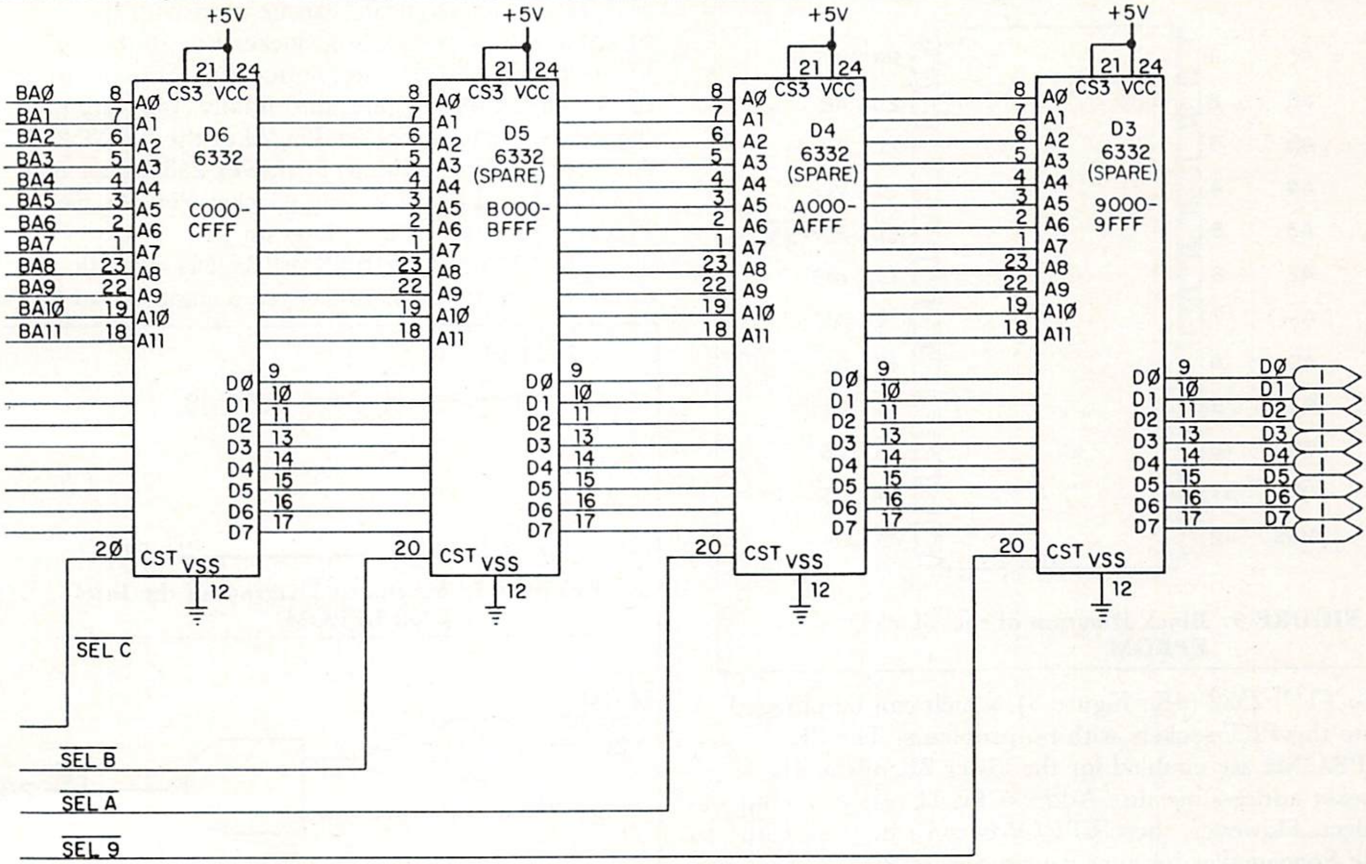


FIGURE 1. Schematic Diagram of PET ROM Modification

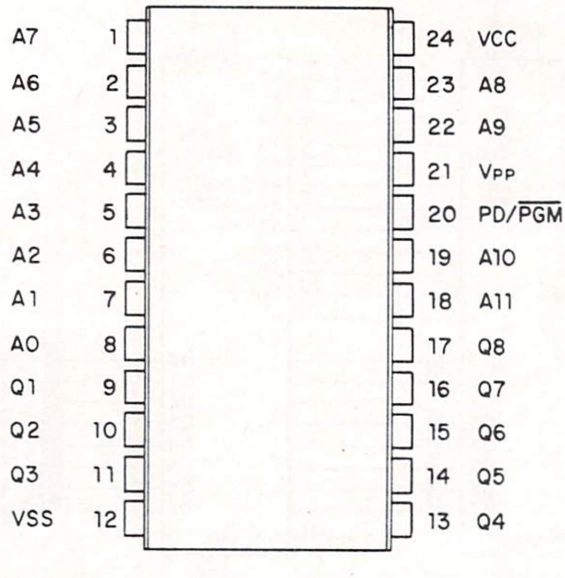


FIGURE 3. Block Diagram of the TI 2532 EPROM

the TI™ 2532 (4K, Figure 3), which can be plugged into the PET sockets with no problems. The 2K EPROMs are enabled for the lower 2K of the 4K socket address because Address Bit 11 acts as a chip select. However, these EPROMs can't be used with the Spacemaker because it uses Socket Pin 21, a high-enable chip select on 2K and 4K ROMs, to select one of its two sockets.

We have developed circuitry to overcome these difficulties and allow combinations of EPROMs with the Spacemaker. The circuits described provide a full 12K of expansion ROM space.

Intel also makes the 2732 EPROM 4K (Figure 4) which can't be used directly with either the PET or Spacemaker; however, with the switch of a wire and an inverter, the 2732 EPROM can be used with both the PET and Spacemaker. To use the 2732 EPROM, connect from Socket Pin 18 to Socket Pin 21 of the 2732, which connects All. Then go from Socket Pin 21 through an inverter to Socket Pin 18 of the 2732, which allows use with the Spacemaker. See Figure 5 for a schematic diagram.

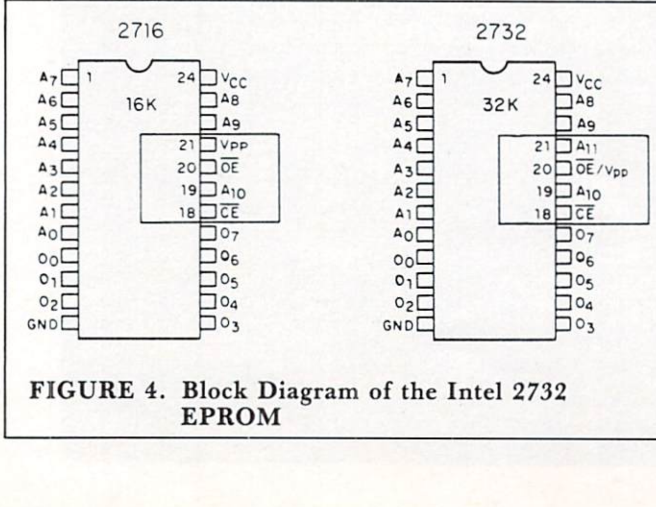


FIGURE 4. Block Diagram of the Intel 2732 EPROM

The TI 2532 eprom can be used with the Spacemaker by connecting Socket Pin 20 through an inverter to a NAND gate, connecting Socket Pin 21 to the same NAND gate, and finally, the output of the NAND gate to Socket Pin 20 of the TI 2352. Also Socket Pin 21 (Vpp) of the TI 2532 must be connected to a 5-volt supply (Socket Pin 24). See Figure 6 for a schematic diagram.

The 2716 and 2516 2K EPROMs can't be used directly with the Spacemaker, so a small board has to

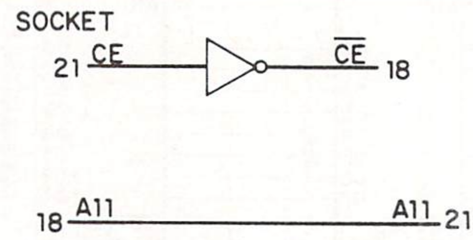


FIGURE 5. Schematic Diagram of the Intel 2732 EPROM

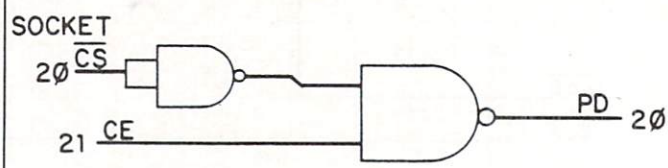


FIGURE 6. Schematic Diagram of the TI 2532 EPROM

be built that includes enough decoding to allow one 2K EPROM for the lower 2K of the 4K available from the socket, and another 2K EPROM for the upper 2K address space. A three-to-eight decoder is used by Decoding Pin 21 (the high chip select from the Spacemaker), Pin 20 (the low chip select from the PET), and Pin 18 (Address Bit 11). Pins 18 and 20 of the EPROMs are connected together (as both are chip selects) and are connected to the decoder output as given in Table 1. Figure 7 shows a PC board with (from left to right) the ribbon cable input, the BASIC Programmer's Toolkit, a 2716 ROM, and the 3-to-8 decoder.

Table 1 Truth Table for Decoder

C (21)	B (20)	A (18)	
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	Select lower 2K 2716
1	0	1	Select upper 2K 2716
1	1	0	
1	1	1	

C (21) is the high chip select from the Spacemaker.
B (20) is the low chip select from the PET.
A (18) is Address Bit 11 from the PET.

Additional Space Available

The PET has four ROMs (three 4K ROMs and one 2K ROM) for its 14K operating system. The 2K ROM is addressed from $E000 to $E7FF. The PET uses $E800 to $E8FF for the keyboard scan chip (6520), the IEEE interface (6520), and the USER PORT (6522) chips.

The address space $E900 to $EFFF is not used and is available for expansion. It can be used with a modification of the method for 2K EPROMs and a 2716 EPROM. The method is the same as that described above, except that the 2716 EPROM is not enabled for the first 256-bytes of its address space. This can be done with the 3-to-8 decoder enabling one of the chip selects and Address Lines 8, 9, and 10 through a three-input NOR gate. This enables the other chip select. Using the NOR gate will disable the EPROM for the first 256 bytes of the 2K range (Figure 8).

FIGURE 8. Schematic Diagram of the $E900 Address Space

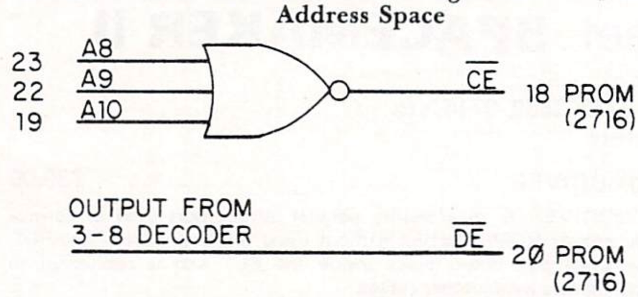


Figure 9 shows the inside of a PET with two Spacemakers (the vertical boards in the $A000 and $B000 ROM socket, and a PC board modified to use $E900-$EFFF.

All the boards mentioned in this article are connected to the PET or Spacemaker socket with a 24-pin dip socket to dip socket ribbon cable. The extra circuitry is powered from Pin 24 (5 V) and from Pin 12 (ground).

Conclusions

We have programmed an EPROM which includes the PET disk operating system (DOS), the support program (the WEDGE), a screen print routine, an expansion for the machine language monitor, and a routine for repeat keys for the PET. This EPROM is enabled into the system with one SYS command.

Acknowledgements

The following company trademarks are acknowledged:

PET and CBM - Trademarks of Commodore Business Machines, Inc., Norristown, Pa.

Toolkit - Trademark of Palo Alto ICs, a division of Nestar Systems, Palo Alto, Calif.

Jinsam - Trademark of Jini Micro-Systems, Inc., Bronx, N.Y.

Spacemaker - Trademark of CGRS Microtech, Inc., Langhorne, Pa.

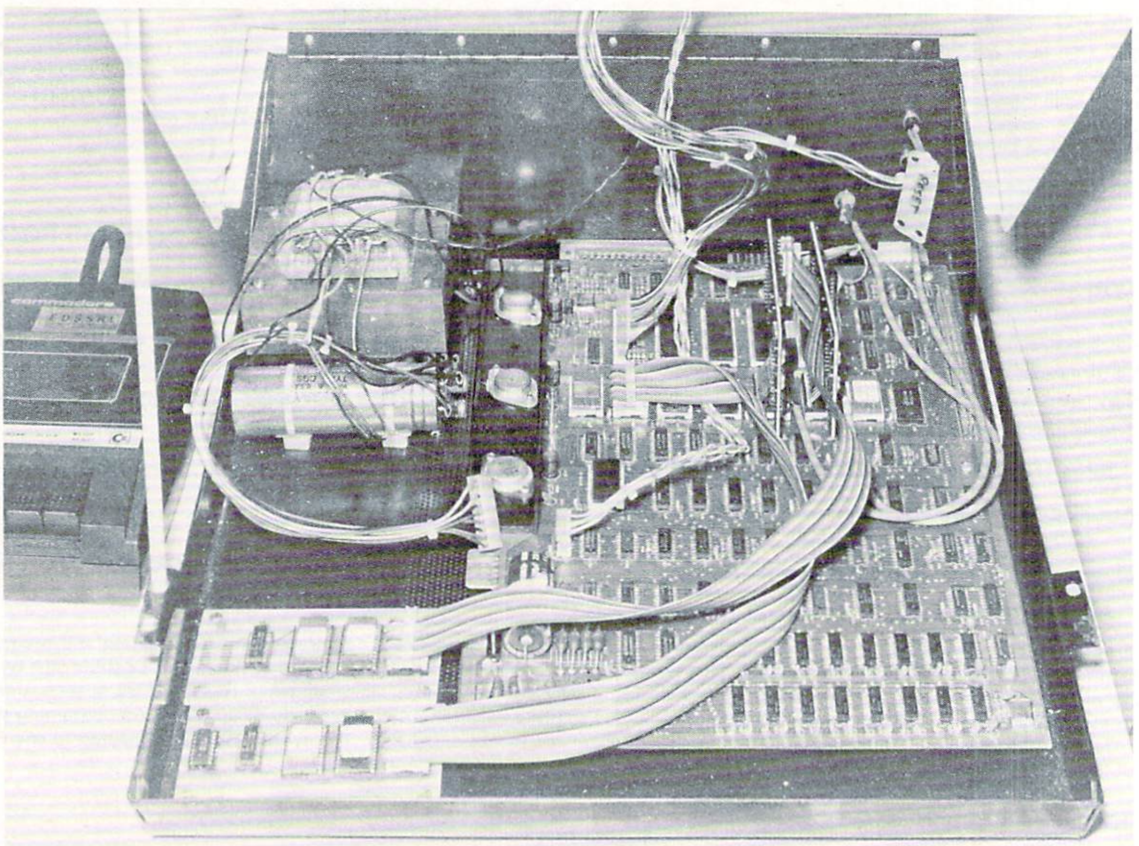
Intel - Trademark of TM Corporation, Santa Clara, Calif.

TI - Trademark of Texas Instruments, Houston, Texas.

REFERENCES

1. **Component Data Catalog,** Intel Corp., Santa Clara, California (1979).
2. **The MOS Memory Data Book,** Texas Instruments Inc., Dallas, Texas (1979).
3. **Logic Diagram Dynamic PET ROMs,** Commodore Business Machines Inc., Norristown, Pennsylvania (1980 rev.).

FIGURE 9. Interior of the Commodore PET



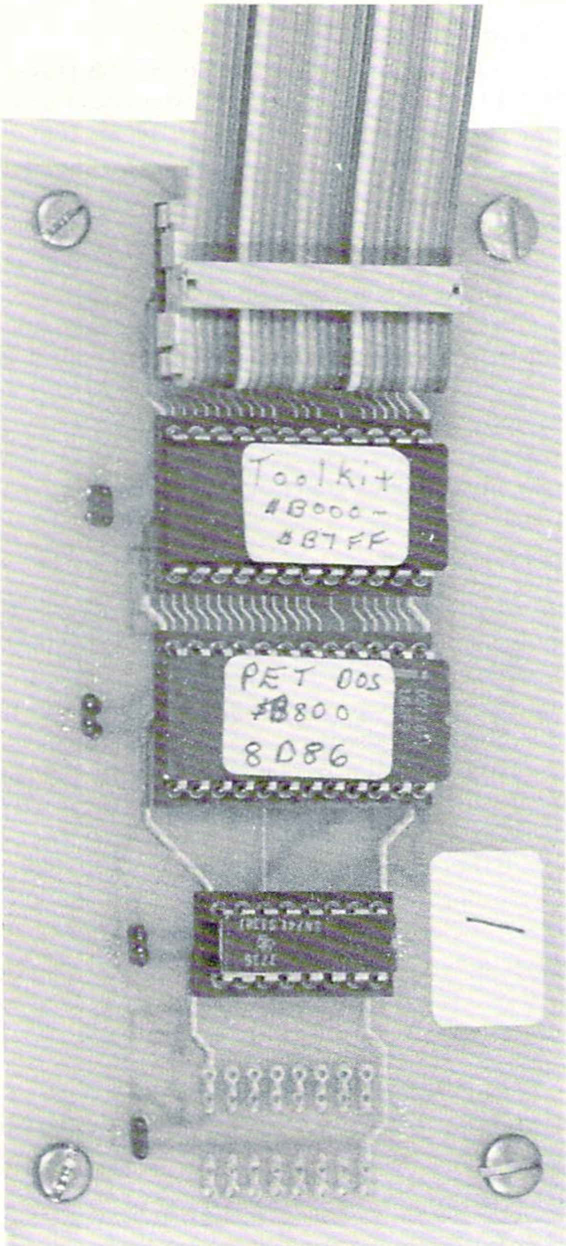


FIGURE 7. PC Board layout ©

Working With Basic 4.0

Jim Butterfield, Toronto

Machine Language Monitor

If you want to get printed output from the Machine Language Monitor in Basic 4.0, you'll have to call the monitor with SYS 54386.

Open your printer first, or course, with OPEN 4,4 and set the output with CMD4 — then give the SYS call. You can't give the usual SYS 1024 or SYS 4 commands: these get to the MLM by means of a Break command, and on the 4.0 system, a break cancels the CMD.

There's a good reason for this. The BRK (Break) instruction is normally used for debugging: it stops a program in mid-execution and allows you to examine registers and memory, and to make changes if you wish. If your program under test mistakenly set the CMD value to something unwanted, you'd lose control unless the BRK set it back. So... the Break action resets output to the screen.

When you want printed output from the MLM, however, you want to enter it without resetting the output. Use the call above, which enters the monitor directly and skips the reset part.

For normal output to the screen, the usual SYS 4 will still do the job nicely.

DOS Commands

It's nice to be able to say things like SCRATCH "WORKFILE", either as a direct statement or in a program. At first glance, however, it seems that variables won't work: you can't say SCRATCH X$, where X$ defines the file name. Similarly, you can designate drive zero within a command as D0 or drive one as D1, but you can't use DN to designate drive N, where N is a variable of value 1 or 0.

It all becomes easy when you learn the syntax. Just put the variable name in parentheses and it all works. So you can say, SCRATCH (X$), D(N) and you'll remove the file whose name is defined in X$ from the drive whose number is defined by N. This kind of syntax applies generally to the DOS commands.

IEEE-488 Timeout

Most of us will never meet a timeout on the GPIB; it only happens when certain slow devices are fitted. If you do have such a device and you get a timeout, the only solution was to go back and keep trying until the device becomes ready.

On Basic 4.0, you're allowed to defeat the timeout feature if you wish and tell the PET to wait forever. You do this by setting location 1020 (hexadecimal 03FC) to a negative value: POKE 1020, 255 will do the trick nicely in Basic. Reset the location to zero when you want to restore normal timeout action.

Review:

Papermate Wordprocessor

Paul W. Sparks
Alexandria, Va.

This is a review of PAPERMATE, developed by Michael Riley and sold by A B Computers for $29.95. (115 E. Stump Road, Montgomeryville, PA 18936).

The main reason that I became interested in this software is that I have an 8K PET with upgraded memory to 32K, both Commodore and Computhink disk units and both a Commodore and a non-standard printer. With a system like this, other available word processors are not as attractive because they cannot be used with the entire system and may require extra ROM boards to install. However, after I received this system I think that even users with a more conventional setup will be interested in using PAPERMATE. It can be used with old and new ROMs, is written in BASIC and ML, and can use either a CBM or an ASCII printer. It also uses a CBM disk or can be user modified to use another disk unit. It has variable line spacing within text, variable margins within text, shift for upper case in either ROM set, all caps look, auto repeat for all keys, center text within text, right justify, multiple tab control, edit, delete, insert, block handling and out of text, header, footer, page numbering, scrolling either direction, and most words on the screen shift to avoid splitting. And it's fast.

Let's review the PAPERMATE commands. First of all there are program commands and in-text commands. The program commands are given by exiting the text to a list of one letter entries to perform functions including block handling and disk operations. The in-text commands can be added or changed at any time and are primarily printer commands such as tabs and justification.

Program Commands:

SAVE: This command allows a text to be saved on disk. It will allow for either CBM disk or tape. Other units may be used if programmed by the user.

LOAD: This command allows loading a text from disk or tape.

WRITE: This command puts the user into the mode to enter text from the keyboard. The text may be edited at all times and the text is always available unless it is replaced by a LOAD which will put another text in memory. In-text commands may be added as you go along.

PRINT: This command will print out your text on the printer of your choice. All in-text commands will be translated into printer variables. You have the ability to stop the printout at any time and there is a feature to pause the printout at the end of each page so that if you are using single sheets they may be replaced.

FORMAT: This feature will cleanup the screen presentation if you have been inserting or deleting a lot of single words or letters in text.

KEY DEFINITION: This will allow you to change the output of one or more keys of your choice to whatever you want. The STOP key is already predefined to give a "." while in the WRITE mode. If you don't like your keyboard you can "change" it.

COMMAND: This feature allows you to give your CBM disk any command that would be used with a PRINT#15 (command channel mode).

The following commands are all block commands. The block is defined in the text by using the bracket characters to define the beginning and the end of the block.

TRANSFER: This command will transfer the defined block to be inserted at the point that the cursor was left (in the margin) before going to the program command section.

DELETE: This command will delete all text that is included in the block.

MEMORIZE: This command will allow the user to place on disk the defined block of text (to be used later).

APPEND: This command will add any block of text to the end of the file presently in memory. This block can then be moved anywhere desired using the TRANSFER command.

INSERT: This command will insert blanks equal to the size of the defined block and move the text down out of the way. It makes space.

In-Text Commands:

MARGIN: The left and right margins may be set and changed at any time. A command is available to decrease the left margin setting by so many spaces.

LINE CONTROL: Line spacing (double, triple, and more) can be controlled and changed in text. The program will allow for the option to right justify. There is a command to center the next (number supplied by the user) several lines of text. There is also a semi-automatic mode to use hyphens. One of the best features is the ability to set up multiple TABs and to change them at will.

PAGING: The page size may be specified (normally 66 lines). A header to be printed at the top of each page and a page number at the bottom of each page can be used. Two other paging commands are a pause feature so that paper may be changed at the end of each document and a command that will force the printer to go to the end of the page if a given number of lines are not available. This is very useful with tables or figures so that they are printed without being broken up between two pages.

LINKING: A title for each file is used. This will automatically be setup for disk use. A command that will call out the next file to be linked in a printing operation is available and any number of files may be joined in this manner.

FORM DOCUMENTS: There are several commands that allow the user to setup a form letter or any other type of recurring document and use a mailing list, etc. to fill the required information. The files may be set up using PAPERMATE or a separate program may be used. The file requirements are delineated.

GRAPHICS: There are two special commands that allow control characters to be sent to the printer. Therefore the secondary address on the CBM printer may be changed and enhanced graphics may be utilized.

MISC: There is a command that allows the user to insert non-printable remarks in the text.

There are a few non-command features that should be noted. All keys have a simple repeat feature that is activated by holding the key down for more than a second. The system will coexist with both disk DOSs and old and new BASIC. I do not know how it will work with BASIC 4.0 *(Editor's Note: We checked with the vendor, and Papermate runs on 4.0 and 80 column machines. RCL)* A simple method is described to convert for an 80 column CBM so that would indicate that it can be used with the new operating system. I would want to verify that before I bought it for that purpose. I have used it with both versions of the disk operating system successfully. One other neat little function is that if you type the REV key then the text will be typed in upper case for letter characters only and numbers, punctuation, etc. will be printed normally.

So what is wrong with it? As with many software packages I have seen, the documentation leaves a little bit to be desired. Some of the pages did not print very well. Many of the explanations were written by and for someone already familiar with the system and although they make good sense now they were quite hard to figure out at first. I believe that this system is equivalent to WORDPRO II. One must keep track of what file ties into where if you are going to insert blocks here and there. That is a function that only the much more expensive systems will perform but PAPERMATE will nicely chain files together for printing just by a single command at the end of each file. One last potential problem; if you are a novice programmer you will have considerable difficulty following Michael Riley's programming. He has done an excellent job of utilizing some very unique skills to get a lot of power from a comparatively short program and it is not easy to follow. I have changed several areas for my own idiosyncrasies and even though there is a list of variables, and program sections are defined, it was not easy and there are many pitfalls.

To summarize, the best recommendation that I can give is that this is the first program that I have been able to get my wife to use in the three years that I have owned my PET. You ought to consider this alternative if you are thinking of getting a word-processor program for your PET/CBM. ©

Disabling the STOP key

Jim Butterfield, Toronto

The STOP key can be disabled so that a program cannot be accidentally (or deliberately) stopped.

METHOD A is quick. Any cassette tape activity will reset the STOP key to its normal function, however.

```
Original ROM:  Disable STOP with POKE 537,136
               Restore STOP with POKE 537,133
Upgrade ROM:   Disable STOP with POKE 144,49
               Restore STOP with POKE 144,46
4.0 ROM:       Disable STOP with POKE 144,88
               Restore STOP with POKE 144,85
```

Method A disconnects the computer's clock (TI and TI$). If your program needs these, use method B.

METHOD B is more lengthy, but does not disturb the clocks. This method prohibits cassette tape activity.

```
Original ROM:  100 R$ = "20>:??:9??8 = 09024<88
                 >6"
               110 FOR I = 1 TO LEN(R$)/2
               120 POKE I + 900,ASC(MID$(R$,
                 I*2- - 1))*16 + ASC(MID$(R$,
                 I*2)) - 816 : NEXT I
     After the above has run:
               Disable STOP with POKE 538,3
               Restore STOP with POKE 538,230
Upgrade ROM:   100 R$ = "20>:??:9??8 = 9;004<31
                 >6"
               110 FOR I = 1 TO LEN(R$)/2
               120 POKE I + 813,ASC(MID$(R$,I*
                 2 - 1))*16 + ASC(MID$(R$,I*2)) -
                 816 : NEXT I
     After the above has run:
               Disable STOP with POKE 145,3
               Restore STOP with POKE 145,230
4.0 ROM:       100 R$ = "20>:??:9??8 = 9;004<58
                 >4"
               110 FOR I<1 TO LEN(R$)/2
               120 POKE I + 852,ASC(MID$(R$,I*
                 2 - 1))*16 + ASC(MID$*R$,I*2)) -
                 816 : NEXT I
     After the above has run:
               Disable STOP with POKE 145,3
               Restore STOP with POKE 145,228
```

How they work: Method A skips the clock update and stop key test. Method B builds a small program into low memory which allows the clock update and stop key test to be performed, but then nullifies the result of this test. The small program for method B is contained in R$ in "pig hexadecimal" format. Machine language programmers would read this as: 20 EA FF (do clock update, stop key test) A9 FF 8D 9B 00 (cancel stop test result) 4C 58 E4 (continue with keyboard service, etc.) ©

Dissecting C. W. Moser's ASSM/TED 1.0

Francis Turco

Carl Moser's excellent assembler/text editor for the 6502 Microprocessor has been reviewed superficially in several publications.[1,2] So far, no one has done an in-depth write-up for a PET owner who wants to understand or modify his copy. The manual provided by Moser is adequate, but sketchy in some areas. I, for one, would like to see some articles by users who have figured out solutions to problem areas.

For example, PET owners find out (on page 35 of the manual) that "At present, the ASSM/TED does not contain a printer subroutine...". In another area, the ASSM/TED is designed for a "standard" PET and utilizes the audio cassette drives for off-line storage. The manual (Sections 6 & 7) discusses configuring the ASSM/TED for disk operation and using it with disk. This discussion is too brief to be understandable by a novice assembly language programmer.

In still another area, the editor has many powerful capabilities and will accept a full line of characters (65 typed characters) but the sense of the shift key is reversed. That is, shift gives lower case letters. Unshifted gives upper case letters. This proves to be cumbersome when typing a letter or manuscript from the PET keyboard.

In an effort to shed some light for others, who like myself, are trying to understand and modify their copy of ASSM/TED and perhaps stimulate some of you to share your findings, I am submitting some areas that I have uncovered in Moser's Assembler.

Figure 1 shows a memory map of the assembler/text editor. The assembler is written for a 16K PET and fills almost all useable memory space. As the figure shows, the assembler and text editor are co-resident and occupy the space from $2000 thru $3FFF. Commodore's monitor occupies the area from $0400 thru $076C. This leaves enough memory for a relocatable file ($1F00 thru $1FFF), a lable file ($1800 thru $1EFF), and approximately 4K for user programs ($0770 thru $17FF).

Table I is a list of addresses of major routines. This is a fun table -- try some experimenting with it. For example, RUN 8390 will assemble your program. RUN 8390 LIST will assemble and list. RUN 8470 will print your program. Table II provides a list of addresses of the pseudo opcode routines, while Table III contains some interesting areas that will be helpful to someone modifying his assembler.

Carl Moser's ASSM/TED is a very good program and will allow the PET owner to convert his PET into a 6502 development station with a little effort on his part. If the PET is equipped with a line printer off the IEEE port, the owner can easily get around the first problem area and get a listing of his source code and/or his assembly. This subject will be treated in PART II of this article.

1. **Compute**, Fall 1979, p. 100, "6502 Macro Assembler and Text Editor SYM Version" by Harvey Herman
2. **The PAPER**, Vol. II, Issue 6, August 1979, "Relocating Macro Assembler/Text Editor 1.0 by R. Busdieker

Figure 1. ASSM/TED 1.0 Memory Map

HEX	DEC	
3FFF 2000	16383 8192	ASSEMBLER & TEXT EDITOR by C. W. MOSER
1FFF 1F00	8191 7936	RELOCATABLE FILE (256 BYTE BUFFER)
1EFF 1800	7935 6144	LABEL FILE (SYMBOL TABLE)
17FF 0770	6143 1904	USER'S TEXT FILE (SOURCE CODE)
076C 0400	1900 1024	COMMODORE'S MONITOR (876 BYTES)
03FF 0000	1023 0	RESERVED FOR COMMODORE'S OPERATING SYSTEM

Table I
MAJOR ASSEMBLER ROUTINES

HEX	DEC	ROUTINE	
2033	8243	CLEAR	user's text file
208A	8330	BREAK	to monitor
2098	8344	AUTO	line number
20A0	8352	GET	program from tape
20A6	8358	FORMAT	text file
20B6	8374	MANUSCRIPT	line numbers output/ not output
20C6	8390	ASSEMBLE	source code
20FF	8447	RUN	program previously assembled
2116	8470	PRINT	text file
2AFB	11003	OUTPUT	create a relocatable object file
2E52	11858	LABELS	prints out label file
31EE	12782	PASS	execute the second pass of assembly
333E	13118	NUMBER	re-number text file
3467	13415	PUT	program out to tape
3559	13657	FIND	character string specified
355F	13663	EDIT	change source code
3844	14404	HARD	print routine (not functional on PET)
3873	14451	COPY	lines of text
39B9	14777	MOVE	lines of text

39C2	14786	DELETE	lines of text
39EF	14831	SET	boundaries of text file, label file & buffer
3A80	14976	DUPLICATE	files from tape 1 to tape 0
3AB6	15030	ENTER	file name in the diskette directory
3AC7	15047	LOOK UP	file name in the diskette directory
3B50	15184	SHIFT	upper/lower case

Table II

PSEUDO OPCODE ROUTINES

HEX	DEC	ROUTINE	
2919	10521	.DS	Designate Storage
2964	10596	.EJ	Eject
297B	10619	.RS	Resolve address & Store
2980	10624	.CE	Continue with Errors
2985	10629	.OS	Object Store option
298A	10634	.OC	Object store option Clear
298F	10639	.CT	Continues on Tape
2994	10644	.LS	List option Set
2999	10649	.LC	List Option Clear
299F	10655	.SI	Store Internal address
29A8	10664	.SE	Store External address
29B3	10675	.BA	Beginning Address
29F3	10739	.MC	Move Code
2A1D	10781	.BY	Bytes
2A57	10839	.DI	Designate Internal
2A60	10848	.DE	Designate External
2AB7	10935	.EN	End
3378	13176	.RC	Resolve Code
3D1E	15646	.ES	Output macro generated object code
3D23	15651	.EC	Supress macro generated object code
3D6A	15722	.MD	Macro Definition
3E0C	15884	.ME	Macro End

Table III

INTERESTING AREAS

HEX	DEC	ROUTINE
2000	8192	Cold start of ASSM/TED 1.0
203F	8255	Command Line Interpreter
207A	8314	Initializes Pointer for Text File
2090	8336	Warm start of ASSM/TED 1.0
2190	8592	Same as 8599 + carriage return
2197	8599	Prints out the double slash after listing
2602	9730	Reads remainder of entered command - For Example: PRINT 100 200 or FORMAT CLEAR
26AB	9899	Jump Table for Major Assembler Routines (Commands)
271C	10012	Pseudo Opcode Table
27AA	10154	Mnemonics Table
2E89	11913	Xfers Pointer for Lable File to Zero Page
		Initialize Pointer for Lable File
2F96	12182	Stores a Zero Pointer +2
32DB	13019	Prints character that is in accumulator (same function as 65490 in BASIC ROM)
330B	13067	Prints carriage return
331A	13082	Prints 2 spaces
331D	13085	Prints 1 space
3323	13091	Converts accumulator to Hex & prints it
354F-3558	13647-13656	Permanent Copy of Value of Boundaries for Text, Label & Buffer (See also 14889)
37E2	14306	Moser suggests this location for a JSR to a line printer routine written by the user. The routine at 13019 would call this subroutine.
3A29	14889	Prints out the boundaries & the present end of data (See also 13647)
3F00-3FFF	16128-16383	Relocated Page 1 variables
3F35-3F85	16181-16261	Keyboard Buffer

©

Pet File I/O In Machine Language

Raymond A. Diedrichs

The PET's I/O scheme is very flexible. A BASIC language program can effortlessly specify that the PET read data from an input device (tape, disk, or keyboard) and send processed data to an output device (printer, plotter, or CRT). At all times the PET knows which devices are for input, which are for output, and when and how each should be used.

But there are occasions when we need an assembly language program. Must we give up the PET's facile I/O because we choose to use machine language? No, indeed. It is as easy to perform standard PET I/O in machine language as it is in BASIC, and this article will explore the necessary techniques.

Let's first be certain that we understand exactly what it is that we specify when we write BASIC language I/O statements. Consider this simple-minded program:

```
10 REM EXAMPLE ONE
20 INPUT A
30 PRINT A
```

The I/O actions ("INPUT" and "PRINT") are indicated, but the I/O devices are not. The PET therefore uses the default devices: the keyboard for input in line 20 and the CRT for output in line 30.

A second example uses explicit device indication:

```
10 REM EXAMPLE TWO
20 OPEN 1,2,0,"SAMPLE"
30 INPUT #1,A
40 INPUT B
50 PRINT A,B
```

In line 20 logical file 1 is created for input of file "SAMPLE" from cassette unit 2. In line 30 file 1

(''#'') is the input source, so the PET seeks input from cassette 2, which is the device associated with the file. In line 40 the PET once again uses the keyboard for input, and in line 50 the output is sent to the CRT.

There are multiple I/O devices in example two, but the PET copes by temporarily ignoring all devices except the pair which are in current use for input and output. If a file number is attached to the I/O statement, the file device is made current; if no file is indicated, the PET reverts to the default device.

When we use machine language, the default input and output devices remain the same as in BASIC. Therefore, when

```
JSR $FFCF
```

is executed, the next character in the input stream is returned in register A. Likewise, when

```
LDA #ASCII-CHARACTER
JSR $FFD2
```

is executed, the character in register A is displayed on the CRT in the next print position.

To use devices other than the default in machine language, we must open a logical file for each device and supply the same information that is supplied by the equivalent BASIC OPEN statement. The following routine performs this function:

```
!
! MACHINE LANGUAGE OPEN STATEMENT
    [NEW ROM PET]
!
    LDA #FILE-NUMBER
    STA $D2
!
! IF THE DEVICE HAS A BUFFER, DECLARE THE
    BUFFER START ADDRESS
! IF NO BUFFER IS NEEDED, SKIP THIS SECTION
!
    LDA #.LOW.ADDRESS-OF-DEVICE-BUFFER
    STA $D6
    LDA #.HI.ADDRESS OF DEVICE BUFFER
    STA $D7
!
! IF THE FILE HAS A NAME, THE NAME MUST
    RESIDE SOMEWHERE IN MEMORY
! AS AN ASCII CHARACTER STRING. DECLARE THE
    LENGTH OF THE NAME IN
! CHARACTERS AND THE STARTING MEMORY
    LOCATION OF THE NAME STRING.
! IF THE FILE DOESN'T HAVE A NAME, DECLARE
    A NAME LENGTH OF 0
    LDA #FILENAME-CHARACTER-STRING-LENGTH
    STA $D1
    LDA #.LOW.ADDRESS-OF-FILENAME-
      CHARACTER-STRING
    STA $DA
    LDA #.HI.ADDRESS-OF-FILENAME-CHARACTER-
      STRING
    STA $DB
!
! CALL A PET SYSTEM SUBROUTINE WHICH OPENS
    FILE USING THE
! INITIALIZED SYSTEM VARIABLE SET
!
    JSR $F524
```

This set of system variables can assume any values which make sense in the equivalent BASIC OPEN statement. The secondary address for a cassette file, for example, can be 0 (for READ), 1 for (WRITE), or 2 (for WRITE with EOT).

When input or output is required from a file device, we must make that device a current I/O device:

```
!
! ROUTINE TO MAKE A DEVICE THE CURRENT
    INPUT DEVICE [NEW ROM PET]
!
    LDX #FILE-NUMBER
    JSR $FFC6
!
! ROUTINE TO MAKE A DEVICE THE CURRENT
    OUTPUT DEVICE [NEW ROM PET]
!
    LDX #FILE-NUMBER
    JSR $FFC9
```

If, for example, a printer is attached in this manner, then the output routine at $FFD2 sends output directly to the printer on the IEEE-488 bus; all 'LISTEN' and 'UNLISTEN' commands are included. When the cassette is attached for input or output, the PET performs all cassette motor control, data buffering, prompting (''PRESS PLAY AND RECORD''), and logging (''SEARCHING FOR SAMPLE'').

When the attached device is no longer needed (or temporarily not needed — I/O can be mixed just as example two showed), restore the default devices in this manner:

```
JSR $FFCC
```

The file device is now detached, and the file can be closed:

```
!
! ROUTINE TO CLOSE A LOGICAL FILE [NEW ROM
    PET]
!
    LDA #FILE-NUMBER
    JSR $F2AE
```

In this article, we have seen that standard PET I/O in machine language is simple, uniform, and economical of user program space. ©

How To Get Started In Machine Code

And Not Go Crazy With A Routine For Two Joysticks

Elizabeth Deal

This article is for people who are taking their first steps in machine code. The ideas are illustrated by a relocatable subroutine for two joysticks that are attached to the Pet exactly as suggested by Harvey Herman in **COMPUTE!**, vol. 1, #4, p. 89. It is written for a new-ROM Pet, with untested modifications for the old-ROM Pet.

My goal for the joystick routine was quick interpretation of joystick positions into numeric keypad equivalents of those positions. The details of conversion have been described by Harvey Herman. Location 59471 ($E84F) contains 255 ($FF) when joysticks haven't moved or another 8-bit number when they have. Four bits belong to each joystick. A Basic routine that examines the 8-bit number and converts it to a numeric keypad equivalent takes a long time. The process is instantaneous in machine code. I've added a feature of wait option. Some games require motion to occur before continuing while others do not. Instead of having to write or omit an equivalent Basic statement the wait option is entered into the program. Finally, I wanted this routine to be relocatable and coexistent with other machine code routines I have.

The January 1980 issue of **COMPUTE!** contained a plea from Mr. Schiller to make assembly listings more accessible to users. I share Mr. Schiller's concern. Note that the three different looking programs are all the same. The disassembled listing in figure 1 begins at hex location $7000 (28672 decimal). The double letter machine code sits between the address ($7000) and the assembler code (LDA © $34). The code on the first line consists of A9 34. Monitor listing in figure 2 also begins with A9 34 and so do the DATA lines in figure 3 where that pair of machine instructions has been changed to decimal 169 52.

How you feed your Pet is up to you. The quickest and most error prone way is the DATA lines. This method, however, facilitates moving code from one place to another. Entry via the Monitor is the easiest, but without a relocating program the code can't move. Follow the excellent instructions of Charles Brannon in his KEYPRINT routine (**COMPUTE!**, Nov.-Dec. 1980) to enter the code via the monitor. Save 69 bytes (program and table + 1).

Where to put the code is also up to you. The listing is at $7000. The code does not have to be at $7000. You may put it in the second cassette buffer (decimal 826 or hex $033A), behind your Basic program, at the top of your Pet or any other reasonable place. Watch out - the program flags the change in top of the Pet pointer by POKEing 894 with 1. This is to prevent Pet from losing its memory during successive corrections and reruns. Should you need to make a correction, reset the pointers to their original value and POKE 894 with any non-one value. Final caution — because of the way I implemented relocatability the subroutine must be entered by a SYS call. Independently of location and manner of entry of this code, you must tell Basic where the code is located by setting variable AD properly. My Basic program takes care of it, otherwise you're on your own.

The routine ends at $7033. Locations $7034 to $7043 contain a lookup table of key numbers corresponding to joystick motions. This table is referenced by the address in $11 and $12. Numeric keypad equivalents are returned to Basic in locations 824 and 825 ($0338 and $0339, at the tail end of the first cassette buffer).

The logic of the joystick routine will not tax your intelligence, thus you can spend all of it on learning some simple machine code. This is exactly what I did and will now describe some of my experiences.

Machine language coding is rough when you jump from a higher level language. You seem to chase your own tail going in circles for need of an instruction that will do what you want it to do. Little tools I took for granted in Basic, like addition or handling arrays become a pain when they have to be considered with the same level of seriousness as the payroll program. And if this doesn't send you to the looney bin, the counters that count to 256 and reset to zero when you least expect it and the two byte addresses coexisting with one byte accumulator will get you for sure. You will not find unconditional GOTO in the repertoire, you will not find subscripted variables (arrays), and you'll have to learn to count **backwards** in hex...

The fact is all these tools **are** in the language. You have to find them and improvise, and that's the fun part. What comes out is a different matter. When I first wrote this joystick routine, it worked but I had a nasty feeling that it was too big and complicated for such an undemanding task.

I got the nerve to contact Jim Butterfield and it was a very rewarding experience. Jim helped me come to grips with instructions. He pointed out the sloppy code and gave suggestions for making it better. He also gave me SUPERMON, which contains an assembler, disassembler and all sorts of other

goodies. That helped tremendously, as hand assembly is not one of my favorite activities.

Some of the code that might have looked like this:

```
CPX T      (compare T to X-reg.)
BEQ STP    (if result = 0, goto STP)
JMP PK     (if not, got PK)
STP:       ...
```

was fixed up to a reasonable

```
CPX T
BNE PK
STP:       ...
```

while avoiding branching around branches and stepping on your own feet.

```
STX KY1
JMP END  (direct or indirect)
```

was changed to;

```
STX KY1
BPL END
```

as an appropriate way to write an unconditional GOTO. I wanted to store a value in KY1 and go to END. It's unnecessary to use a JMP instruction in a program this size and use of JMP makes it harder to move the code. Indirect jump, while helping relocatability, is an overkill in a minor procedure. Again the solution is simple. If you know that the flag at that point in the program is, for instance, always positive, a simple "branch if plus" improvises the unconditional GOTO. You may want to read Butterfield's article in Nov.-Dec., 1980 **COMPUTE!** which describes addressing modes and a "reach of instructions".

These examples do not now exist in the program, as they have been superseded by better code. But they illustrate silly trouble one can get in. Needless to say, after Jim Butterfield helped me the code became clearer and shorter.

Now, there is more to programming than just uncluttered, short code, whether the code is in machine, Basic, Pascal, or any other language. And this was the most important bit of advice I got. Combine a program and its tables into a coherent structure. Its simplicity should leap at you. And simple means less work, and perhaps no backward counting in hex...

I will illustrate this concept with a simple Basic example. It shows how a little bit of code can do lots of work. First, the clumsy way. We may set up a table of 10 values from Herman's list and loop through the table until the value of the bit pattern on the user port equals one of the values in the array, in which case the loop index (J) becomes the result. Like this:

```
JS = value of 4 bits
PV( ): 3 9 11 10 13 15 14 5 7 6
FOR J = 0 TO 9
IF JS = PV(J) then K1 = J: get out
NEXT J
```

There is nothing wrong with this type of coding except that it is inefficient. The program may have to loop ten times for each joystick before finding one matching value. A much neater structure is the one used in the routine shown in this article. Its Basic equivalent is:

```
JS = value of 4 bits
KY( ): x x x 0 x 7 9 8 x 1 3 2 x 6 5
K1 = KY(JS)
```

Note the absence of the unnecessary loop. Note the reduction in code. We now have a table of sixteen values, ten of which are the numeric keypad values (0 to 9). In my program x's have been filled with 10 ($0A). Depending on the value of the four bits at the user port the program addresses that position in the table which corresponds to the port value. Thus if the four-bit value at the user port is 9 the program, by use of the Y register, picks up the 9th value in the table (counting from zero) which in this case is 1. This kind of coding is in the disassembled listing at $7023-7027 and $702E-7032.

The rest of the code has to do with housekeeping: testing the stop key at $FFE1 and shifting bits around. The value seen at the user port is saved on the stack (line $701F) so that subsequent shifting and masking will not destroy the value, as it is needed in testing both joysticks. It is brought back in $7028 and reused.

I would like to add two more ideas that will prevent you from going crazy. When you start coding leave lots of room between logical sections of your code and fill the gaps with $EA, that is, for no operation, it is a filler. (The less you know the more EAs you need. Start with fifty). As you discover mistakes you will be able to use the room while expanding code. When the program works, eliminate EAs to create a compact package of code. Try to get an assembler-disassembler. It's invaluable. It makes the work possible. You will be able to concentrate on thinking instead of hassling with hand assembly. SUPERMON, for instance, is available from Pet Program Exchange, P.O. Box 561, Montgomeryville, Pa. 18936. *(Editor's Note: Price is $1.00 for tape and $1.00 for program.)*

If your sanity is still intact after a couple dozen lines of code you'll have a good laugh as you come to grips with that very strange, but useful, language of your Pet.

Changes for old-ROM Pets:

1. In BASIC program:

replace PEEKs and POKEs to locations 52 and 53 by 134 and 135.

first DATA line has two 17-s and one 18,

Replace 17 by 8. Replace 18 by 9.

third DATA line has two 17-s. Replace by 8.

2. In the ASSEMBLY listing: replace $11 by $08 in four places and replace $12 by $09 in one place.

3. In the MONITOR listing: same thing as in the assembler.

```
B*
     PC  IRQ  SR AC XR YR SP
.;  784B 7E11 30 78 5E 34 F0

.
7000 A9 34        LDA #$34
7002 18           CLC
7003 65 11        ADC $11
7005 85 11        STA $11
7007 90 02        BCC $700B
7009 E6 12        INC $12
700B A9 00        LDA #$00
700D 8D 43 E8     STA $E843
7010 20 E1 FF     JSR $FFE1
7013 AD 4F E8     LDA $E84F
7016 AE 37 03     LDX $0337
7019 F0 04        BEQ $701F
701B C9 FF        CMP #$FF
701D F0 F1        BEQ $7010
701F 48           PHA
7020 29 0F        AND #$0F
7022 A8           TAY
7023 B1 11        LDA ($11),Y
7025 8D 38 03     STA $0338
7028 68           PLA
7029 4A           LSR
702A 4A           LSR
702B 4A           LSR
702C 4A           LSR
702D A8           TAY
702E B1 11        LDA ($11),Y
7030 8D 39 03     STA $0339
7033 60           RTS
.?
.
.:  7034 0A 0A 0A 00 0A 07 09 08
.:  703C 0A 01 03 02 0A 04 06 05
.?
```

Figure 1

```
.
.:  7000 A9 34 18 65 11 85 11 90
.:  7008 02 E6 12 A9 00 8D 43 E8
.:  7010 20 E1 FF AD 4F E8 AE 37
.:  7018 03 F0 04 C9 FF F0 F1 48
.:  7020 29 0F A8 B1 11 8D 38 03
.:  7028 68 4A 4A 4A 4A A8 B1 11
.:  7030 8D 39 03 60 0A 0A 0A 00
.:  7038 0A 07 09 08 0A 01 03 02
.:  7040 0A 04 06 05 20 20 20 20
```

Figure 2

I'd like to thank Jim Butterfield for helping me take the plunge into the machine code...for the SUPERMON...lots of patience...and an incredible ability to share his knowledge.

```
100 REM-------------------------------
110 REM SUBROUTINE FOR TWO JOYSTICKS
120 REM ATTACHED AS IN H.HERMAN'S
130 REM ARTICLE - COMPUTE V.1,#4,P.89
140 REM
150 REM       BY ELIZABETH DEAL
160 REM
170 REM FOR LOADING MACHINE CODE INTO
180 REM    SECOND TAPE BUFFER REMOVE
190 REM    'REM' FROM LINE 250
200 REM TO PUT CODE AT TOP OF PET
210 REM    USE LINES 260-300
220 REM SEE ARTICLE FOR CHANGES TO
230 REM    OLD ROM PETS
240 REM-------------------------------
250 :REM SP=67:AD=826:FORI=ADTOAD+SP:
     ¬READV:POKEI,V:NEXT:GOTO500:
     ¬ <TO POKE823,0
260 IFPEEK(894)=1GOTO300
270 SP=67:AD=PEEK(52)+256*PEEK(53)-SP
280 FORI=ADTOAD+SP:READV:POKEI,V:NEXT
290 AH%=AD/256:AL=AD-256*AH%:POKE52,AL:
     ¬POKE53,AH%:POKE894,1:CLR
300 AD=PEEK(52)+256*PEEK(53)
310 :
320 DATA 169,52,24,101,17,133,17,144,2,
     ¬230,18,169,0,141.67,232
330 DATA 32,225,255,173,79,232,174,55,3,
     ¬240,4,201,255,240,241,72
340 DATA 41,15,168,177,17,141,56,3,104,
     ¬74,74,74,74,168,177,17
350 DATA 141,57,3,96,10,10,10,0,10,7,9,
     ¬8,10,1,3,2,10,4,6,5
360 :
370 REM-------------------------------
380 REM POKE823,W
390 REM   W=0  NO WAIT FOR JS MOTION
400 REM   W<>0 WAIT UNTIL MOVED
410 REM GET KEYPAD EQUIVALENTS OF JOY-
420 REM   STICK MOTION FROM 824 AND 825
430 :
440 REM 0=BUTTON      1=LEFT-DN  2=DN
450 REM 3=RIGHT-DN    4=LEFT     5=NONE
460 REM 6=RIGHT       7=LEFT-UP  8=UP
470 REM 9=RIGHT-UP
480 REM-------------------------------
490 :
500 POKE823,0
510 SYS(AD):PRINTPEEK(824),PEEK(825):
     ¬GOTO510
```

©

Machine Language: The Wonderful Wedge

Jim Butterfield

Adding new commands to Basic seems an impossible task at first glance. The Basic interpreter is frozen forever in ROM chips, and unless you're the adventurous type who can program your own EPROM chips, it seems that there's no way in.

It can be done. A small but important part of the Basic interpreter is located in RAM memory. It's written there during system initialization and is available for you to change.

The subroutine is called **CHRGET** (Character Get), and all 6502 Microsoft Basic implementations use it. Every time the Basic interpreter wants to get a character from the Basic statement it is executing, it calls CHRGET.

Here's where you can find subroutine CHRGET in some 6502 systems:

KIM -	C0 to D7 hexadecimal
SYM -	CC to E3
AIM -	BF to D6
OSI -	BC to D3
Apple -	B1 to C8
Early PET -	C2 to D9
PET/CBM -	70 to 87

Our description here will refer to the PET/CBM version, Upgrade and subsequent ROMs.

How it works

Let's look at the CHRGET subroutine in detail.

```
0070 E6 77  CHRGET  INC POINTER
0072 D0 02          BNE CHRGOT       ;skip next instruction
0074 E6 78          INC POINTER + 1
```

Locations 77 and 78 contain the address of the last Basic character obtained. The above coding bumps the pointer to the next address, adjusting the high order address if necessary.

```
0076 AD xx xx CHRGOT LDA xxxx        variable address
```

The address indicated with xxxx above normally points at your Basic program or at a direct Basic statement you have typed in. Note that the address itself has been modified by CHRGET, above.

```
0079 C9 3A          CMP #':          ;ascii colon or higher?
007B B0 0A          BCS EXIT         ;yes, exit subroutine
```

The above coding tests two things. If the new character is a colon, meaning end of Basic statement, we will exit with the Z flag set to one. If the new character is higher than ASCII 9 (hex 39), we will exit with the Carry flag set to one. The meaning of these flags will be discussed in a moment.

```
007D C9 20          CMP #'           ;is it a space?
007F F0 EF          BEQ CHRGET
```

We know that Basic ignores spaces; this is where it happens. If we find a space, we go back and get another character.

```
0081 38             SEC
0082 E9 30          SBC #$30
0084 38             SEC
0085 E9 D0          SBC #$D0
```

This seems to be a curious bit of coding: we subtract 256 from the A register, in two steps, which leaves it with its original value! The point is this: if the A register contains a value less than ASCII zero (30 hex), the Carry flag will be set to one; otherwise, it will be cleared to zero. The Z flag, too, will be affected: it will be set if we have obtained a binary zero.

```
0087 60             RTS
```

What the flags mean

The flags are often checked by the calling routines. The Z flag will be set on if we have found an ASCII colon (end of statement) or a binary zero (end of Basic line).

The Carry flag will be cleared to off if the character is an ASCII numeric, zero to nine (30 to 39 hex); otherwise it will be set on.

How the subroutine is called

CHRGET is called many times during the interpretation of a Basic program or a direct statement. It normally obtains data from the active program; but it is also used to obtain information from DATA statements or keyboard input during READ or INPUT activities. In such cases, the pointer at 77 and 78 is swapped out temporarily.

The Basic interpreter also frequently calls CHRGOT (address 0076) to re-obtain and check a previously obtained character.

From time to time, the pointer at 77 and 78 is used as an indirect address by the interpreter; when we start tampering with the coding of the subroutine, we must be sure to leave the pointer intact in its normal place.

Finally, there is a rare call that is made to the subroutine at address 7D (Compare to space); it doesn't happen often, but we must watch for it.

Keep in mind that the subroutine does not affect the X or Y registers.

Wedging it in

To fit in the extra features, we must "patch" the CHRGET program and connect it to our own code. The patch will destroy some of the existing code, of course, and we must carefully replace it.

There are two places we can insert the patch: at the beginning of CHRGET, or a little distance past CHRGOT. The first location will go into action only when a new character is called up by the interpreter. The second location would be invoked more often, since CHRGOT is called to recheck a previously obtained character.

Let's use the first location; and let's put in a simple do-nothing wedge for starters. Call up the Machine Language Monitor and set up the following memory locations as shown:

```
027A:    E6 77 D0 02 E6 78 4C
0282:    76 00 xx xx xx xx xx
```

The first six locations exactly match the coding at CHRGET. Now we'll put in the patch with:

```
0070:    4C 7A 02 02 E6 78 AD xx
```

Leave the Machine Language Monitor and play with Basic for a moment. Everything still works. It looks like we have found a way to penetrate Basic... but we haven't done anything yet.

A tiny example

Let's write a very small wedge to recognize an "@" sign and break to the monitor if it is seen. Not much in the way of power, but it will show how the technique is used. We'll continue to use the patch at 0070.

To get the character we plan to analyze, we'll have to use indirect, indexed addressing. The pointer is of course at 77 and 78, and we must set the Y register to zero. Since we must not affect the Y register, we must first save its contents, and restore them before we finish.

So our coding will follow the following pattern: STY WORK, to save Y; LDY #0, LDA (POINTER), Y to get the character; LDY WORK, to restore Y; CMP #'@, to check for the @ character in A; BEQ BREAK if we find it; and JMP CHRGOT to return if not. BREAK will have the BRK instruction to go to the Monitor. Let's do it.

```
027A:    E6 77 D0 02 E6 78 8C A0
0282:    02 A0 00 B1 77 AC A0 02
028A:    C9 40 F0 03 4C 76 00 00
```

We have arbitrarily picked address 02A0 as our Y Save location. Now the patch to implement the wedge:

```
0070:    4C 7A 02 xx xx xx AD xx
```

Return to Basic. Try statements which do not contain the @ sign, and others which do.

Final remarks

You're ready to try your hand at more ambitious wedge inserts. Be careful: remember to save X and Y if you use them, and restore them later. Keep in mind that the larger your wedge program, the slower Basic will run. Look for quick tests: for example, many wedge programs will exit instantly unless the statement was input as a Direct command... this can save a lot of time on a running program.

Watch that you don't conflict with other wedge programs, like Trace, Toolkit, or the DOS wedge program. It takes a lot of careful coding; but the results can be dramatic. ©

The Single-Board 6502

Eric Rehnke

The 6500 Family— just where is it going?

Six years ago a star was born. The 6500 family was the product of a chasm between some Motorola design engineers and their management. They broke away and joined MOS Technology, a calculator manufacturer, to help produce one of the most revolutionary pieces of silicon to ever roll off a drawing board. The main idea was to simplify a chip design as much as possible by removing everything but the bare necessities. This led to a chip which was much smaller than its contemporaries and was in turn much easier to produce. Since chip yields were high, the end price could be much lower than the competition. The 6502 was going for $25 when the 6800 had to sell for around ten times that much. Combine that low price with an elegant architecture and a chip which was both easy to learn and use and what did you get? A sure winner, that's what.

And if all that weren't enough, the factory was selling direct to individuals in single quantities. This alone was completely unheard of at the time. But, since MOS Technology management had to play a game of catch up to assure themselves of an adequate number of new design ins, it was a shrewd and well calculated move. It sealed the fate of the 6500 family. It would succeed. But MOS Technology was determined not to give their competition any breathing room.

At a time when most of the industry were selling evaluation kits to help folks become familiarized with their devices, MOS Technology brought out the KIM-1, a system with features that were unheard of. A scanning hex keyboard and display, a built in audio-cassette and terminal interface. All for $245...and it was assembled and burned in at the factory to boot! The marketplace LOVED it! Now we had an easy way to evaluate the 6502 without going through the system design phases on our own.

As an aside, shortly before the KIM-1 came out, I had designed a system which would let me evaluate the 6502 chip. I designed a hex keyboard and display that would allow me to enter and examine data in memory as well as start a program running. Since I knew nothing about software at the time, the system was designed using TTL gates and counters, stuff I was familiar with at that time. My design would have taken *AT LEAST 50 TTL chips* to implement. And there were no means of saving data to an audio cassette or communicating with a terminal. Well, you can imagine how I felt when I first saw a KIM-1 with all the stuff I wanted plus features I didn't even realize I needed. All assembled, tested and at a price I couldn't pass up. As I look back, I can't remember ever making a better investment.

The design cleverness of the KIM-1 only became apparent when users started doing the most amazing things with the system. The most memorable early contributions were those of Jim Butterfield from Toronto, Canada. I can still remember my fascination with his realtime moonlander program. I just couldn't believe the KIM-1 was performing these feats of "magic".

From the early days it became increasingly apparent that 6502 users were the elite of the hobby world. No other computer users were so creative and so willing to share their discoveries. Although Jim Butterfield deserves a lion's share of the credit for helping the KIM-1 become so popular, it was Rick Simpson, then Product Manager of the KIM-1 line, who really got things rolling by getting a few of the early KIM-1 users together. Rick thought up the idea for a KIM-1 user group newsletter and helped me get the original KIM-1 user notes started.

The 6500 family was proving very popular in real time systems such as video graphics machines because of its high throughput capability. It was one of the faster 8-bit chips and also very cost effective to design in because of the low number of support chips that were necesary to get a system together. This made it attractive for small dedicated systems use. Its elegant architecture and instruction set made it a programmer's dream. Since the influence of the DEC PDP-11 was so obvious, even down to some of the assembler mnemonics, people who were familiar with that popular mini could easily migrate downward to an 8-bit system and not have to start from scratch on the learning curve.

A short while after the 6500 family was introduced, thought was given to what should follow. Design specification work began on what was to be called the 6516 Psuedo-16 bit cpu. This processor had a rather unique design feature that gave the chip a dual personality! A single bit in the status register could be set to change the cpu from one that looked exactly like the 6502 to a psuedo-16 bit device with advanced instructions and even greater capability than the 6502.

But, it just wasn't in the cards for this advanced chip to ever become a reality. There was some talk of Synertek doing the chip, but, that never panned out either.

At this point in time, I don't see any real possibility of there being an expanded psuedo (or real) 16 bit cpu based on the 6502. Two years ago, such a chip would have been a hit. Today, it would be a disaster. Sorry bunky.

However, the 6500 family hasn't stopped growing. It just isn't growing in the same direction as some of the other semiconductor manufacturers' products. A lot of the development of the 6500 family of devices is being made in the area of single-chip microcomputer devices. These "computers-on-a-chip" will be increasingly important in the 80's as peripheral processors for the larger 16 bit cpu's. A system with 3 or 4 of the 16 bitters could conceivably have up to 30 or 40 of the single-chippers handling the more mundane system functions.

Admittedly, the thought of single-chippers probably isn't as glamorous as some of the new 16-bitters (such as the 68000), but, if you have a practical use for small dedicated systems around your home or lab, these small "all-in-one" devices will really fill the bill. Rockwell is planning to come out with single-chip devices that can talk to off chip EPROMS so you can easily develop your own programs and not have to buy a thousand of the little buggers just to automate your home energy management system, for example. And since the RAM, and I/O are on one chip (along with the cpu), the system design turns out to be a lot simpler.

And I've just heard a rumor that MOS Technology has announced a new chip called the 6508. Supposedly it has some on-board RAM and one 8-bit I/O port. But, I won't know for sure until the data sheet arrives. Now, if the full 16-bit address bus is present on the 6508, its eight bit I/O port could be used as a bank select for those applications requiring more memory than the normal 64K. Of course, it would be most useful at the lower end of the applications spectrum where the 6508 and an EPROM could form a very low-cost system.

So don't despair, all's not lost. The 6500 will live on.

6502 Word Processor (?)

For quite some time I have wanted a word processor for my HDE expanded KIM system. Now the text editor that comes with the Hudson Digital Electronics disk system is probably the best there is for editing assembler source files. But, it has some limitations which make it a bit awkward when handling articles or doing letter writing. Fortunately, another HDE system user, Chuck Kingston, also felt the need to have a processing capability and he went ahead and wrote one.

His word processor is actually in two parts — the input editor and the output formatter. The editor uses line numbers (which makes the files readable with the HDE editor) but the lines in the editor file don't necessarily correspond to the output of the text formatter. The column width set command (which can either be imbedded in the text file or set up upon entry to the text formatter) determines the printing buffer length. Words are pulled from the input file and stuffed into this printing buffer until it's filed or a new line command has been encountered in which cases, the printing buffer will be dumped to the printer. So the line numbers are essentially ignored when the text is output to the printer.

The editor includes commands for auto line numbering, block moves, string searching and replacing, tabbing and so forth. The text formatter allows margins and column size to be set, files to be chained in from disk, page numbers to be added, and includes lots of other features too numerous to mention.

I think that the main feature of this system which makes it more than just another text editor is the way multiple lines can be inserted anywhere in the file, even between words on the same line. Normal text editors will not let you insert more text on one line than the difference between the amount of text already contained on that line and the length of the line buffer. To insert any more you'd have to delete some of the text already there. Chuck Kingston's word processing system automatically saves any text overrun to new lines and lets you insert text to your heart's content — up to the limits of memory, of course.

This software has added a new dimension of usability to my computer and I can see it being used a lot in my line of work. Now I'm going to have to figure a way to get my system out to the patio so I can work in the sunshine.

If you've been using the HDE text editor, you'll have no problem learning this word processor. I haven't tried to teach my wife how to use this software yet so I can't say how long it would take to teach someone who is not a constant computer user.

At this time, the system is available for KIM systems with the 5'' HDE disk system. The price of $75 (NY residents must add appropriate tax) includes the program object code AND the manual on disk. That's right, the manual is included on the disk as a number of text files. Besides being less expensive to produce, it has the advantage of providing examples of about every command in the system for your inspection. Versions for the 8'' HDE (KIM and SYM) and 5'' HDE SYM systems should be available for the second quarter of 1981. AIM disk versions and cassette versions for the KIM, SYM, and AIM machines are planned for sometime in 1981.

If you do any writing at all, even just personal letters, you'll want a copy of this text editing software for your HDE disk system.

The text editing system is available from Charles Kingston, 6 Surrey Close, White Plains, NY 10607.

The reason for the question mark in the title of this article? Well, when I started to write it, I had wondered about what the difference was between a word processor and a text editor. I tried to figure out in my mind just what this particular system was and what it could be honestly called. After about two days of thought on the subject, I've decided that I don't really know the difference and it may not really matter anyway since everyone will have their own opinion. Since I can manipulate and insert text fairly comfortable with this particular editing software, I think it can be referred to as a word processor. If it were clumsy to operate I might call it a lousy text editor. So you decide...

WHAT IS THE DIFFERENCE BETWEEN A WORD PROCESSOR AND A TEXT EDITOR?????????

A Low Cost Controller Development System

The AIM 65, with an assembler ROM, makes a very usable and low-cost development system, as long as whatever is being developed is meant to reside on the AIM. However, if you should want to develop a dedicated controller system, the AIM doesn't really have the proper tools. What's needed is some way to communicate with the controller board to somehow download the object code into it for testing. But, up until now, there have been no low-cost add-on tools to enable the AIM to handle this kind of duty.

Well, luckily for those of us who need such tools, that is no longer the case. A company called R.J. Brachman Associates Inc. has come up with not only the controller development tools necessary but the controller board as well.

Let's talk about the controller first. Basically, it's a 4.5" x 6" p.c. board that contains a 6503 microprocessor, a 1MHZ xtal, two 6522 VIA chips, two 2114 RAMs, and a socket for a 2716 (2Kx8) EPROM. There's even a power supply on-board that will accept 9 to 18 volts of A.C. The I/O comes off the board from a 44-pin edge connector. As you know, the 6503 has a total addressing capability of 4K bytes — more than enough for a small controller system. The board layout is very clean and professional looking and a solder mask is included on both sides of the board. There's even an LED on the board to indicate power-on.

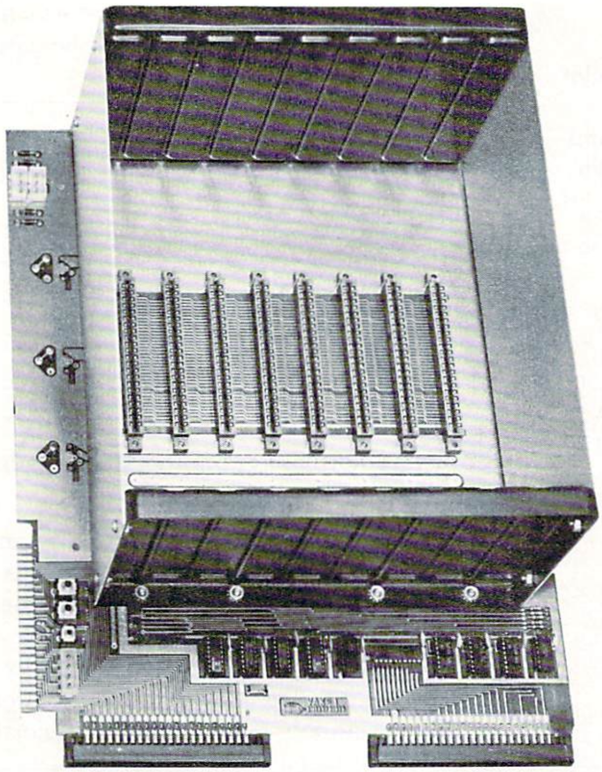
As part of the development package, they have an In Circuit Emulator that takes the place of the 6503 CPU in the controller and maps the lower 2K of the controller (everything but the EPROM) into the address space of the AIM. This means that you can directly access everything in the controller's memory space from the AIM. Think about that!

The controller hardware and software can be fully developed and debugged using the tools already resident in the AIM (like the assembler, the software trace and breakpoint routines). No need to download to another system and work in the blind anymore.

And when the system is developed, R.J. Brachman also has an EPROM programmer which plugs onto the controller and lets you make things more permanent. This is a very nicely integrated development system at a surprisingly low price. The In Circuit Emulator sells for $95, the EPROM programmer attachment sells for $45 and the controller board sells for from $25 for a bare pc board w/manual to $149 fully assembled with the on-board power supply. Pennsylvania residents need to add 6% tax.

The key to the low cost and high usability of this developmment system is in the use of the 6503 as the controller CPU. Because it can only address 4K of memory, all of it can easily be accessed by the host

VAK-1 MOTHERBOARD



We also carry:

SYM-1	$\$229^{00}$
AIM-65 w/1K	389^{00}
AIM-65 w/4K	439^{00}

We also do custom hardware and software for the 6502 microprocessor

PRICE: $\$139^{00}$

Call or write for shipping charges and our complete catalog.

The VAK-1 was specifically designed for use with the KIM-1, SYM-1 and the AIM 65 Microcomputer Systems. The VAK-1 uses the KIM-4* Bus Structure, because it is the only popular Multi-Sourced bus whose expansion boards were designed specifically for the 6502 Microprocessor.

SPECIFICATIONS:

- Complete with rigid CARD-CAGE
- Assembled (except for card-cage). Burned in and tested.
- All IC's are in sockets
- Fully buffered address and data bus
- Uses the KIM-4* Bus (both electrical Pin-out and card size) for expansion board slots
- Provides 8 slots for expansion boards on 1″ centers to allow for wire-wrap boards
- Designed for use with a Regulated Power Supply (such as our VAK-EPS) but has provisions for adding regulators for use with an unregulated power supply.
- Provides separate jacks for one audio-cassette, TTY and Power Supply.
- Board size: 14.5 in. Long x 11.5 in. Wide x 8 in. High
- Power requirements; 5V.DC @ 0.2 Amps.

**KIM-4 is a product of MOS Technology/C.B.M.*

RNB ENTERPRISES INCORPORATED

2951 W. Fairmount Avenue
Phoenix, AZ 85017
(602) 265-7564

master charge VISA

computer through the In Circuit Emulator. Another benefit is that since the 6503 only uses a 12 bit address bus, and the top 4 bits are ignored, programs which reside in *ANY* 4K block can be installed in a 2716 and run in the controller without needing to change any addresses at all!

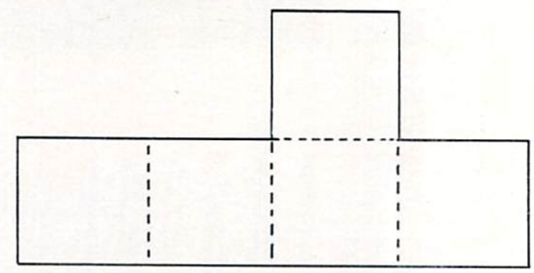
If you've ever tried to develop a small controller without tools like these, you'll know why I'm so enthusiastic about them. Get more information from R.J. Brachman Assoc. Inc., POB 1077, Havertown, PA 19083 (phone 215-622-5495).

Pentominoes—The Ultimate Demo

Every so often a non-computer type comes over to my house and our conversation may casually turn to the subject of computers. If I mention that I have a computer, he'll eventually want to see it in action. Unfortunately, watching an assembler or an editor doing its thing seldom excites the unknowing onlooker. As a result, I have been searching for the ultimate demonstration program. And...I think I just may have found it.

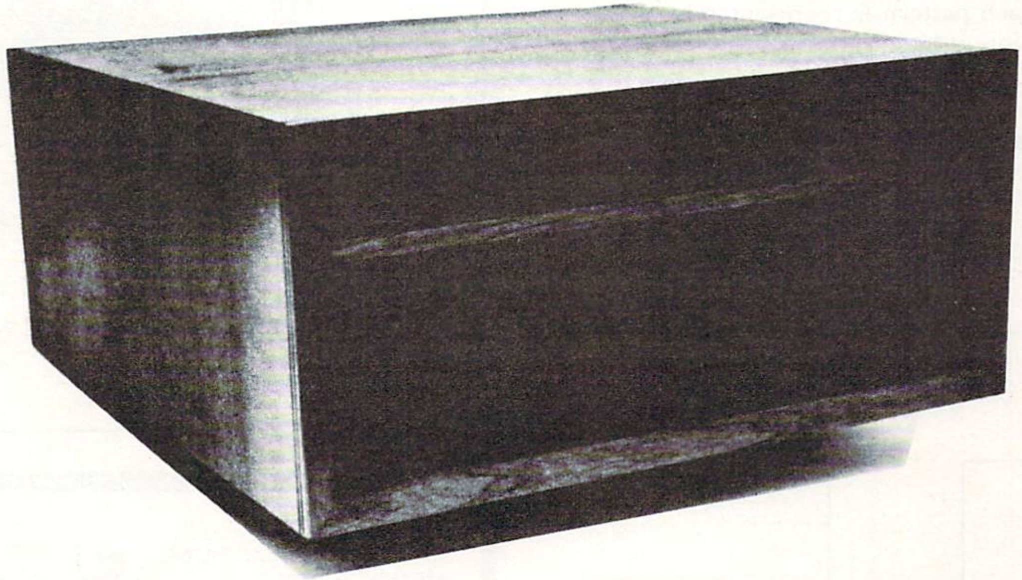
The "ultimate demo" should not require any input from the onlooker and needs to show how powerful the computer is in solving problems.

Pentominoes, a program from the 6502 Program Exchange meets these specifications very nicely. **Pentominoes** is the computeriztion of a jigsaw puzzle that has fascinated and frustrated math and logic types for years. The puzzle consists of 12 pieces that must be fitted together to form a rectangle. Each of the pieces represents one of the twelve possible patterns that can be assembled with five unit squares.



For example, here is one possible combination. There are eleven others.

Since each pattern consists of 5 units and there are 12 such possible patterns, the total area of the rectangle is 60 units. A 6 by 10 arrangement has been popular with puzzle solvers, but, other arrangements (5 by 12, 4 by 15, and 3 by 20) are also possible. The 3 by 20 seems to be the most difficult of the possible rectangle arrangements. According to the

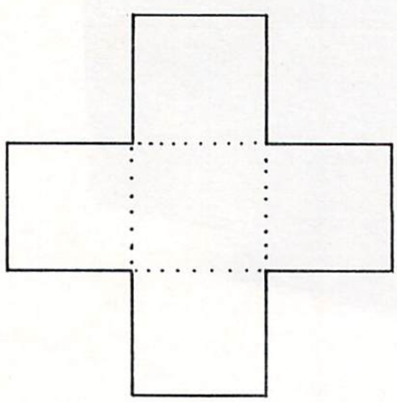


program documentation, out of 1,000,000,000,000,000 possible placement combinations of the 3 by 20 rectangle, only 2 solutions exist.

Pentominoes is a 6502 machine language program that solves the jigsaw puzzle by trying all possible combinations of the pieces for a given rectangle arrangement until it finds a combination that fits together. The desired rectangle size can be chosen by the user as well as whether all, some, or none of the attempts are displayed on the users Ascii output device. Modification information, as well as a complete source listing, is provided to enable you to configure Pentominoes to just about any 6502 system/terminal combination (including memory mapped video).

To let Pentominoes be displayed on any Ascii output device, each pattern is represented by a particular combination of letters arranged in the same pattern.

For example, the cross pattern



is represented by the letter 'A' in the same pattern.

```
 A
AAA
 A
```

On my Hazeltine 1500 terminal, Pentominoes clears the screen and homes up the cursor every time it prints an attempt. Since the terminal runs at 9600 baud with KIM, and the attempts are happening at about 2 or 3 times per second, it looks like an animation and is quite spectacular to view.

Pentominoes costs $15 and is available for KIM, SYM, AIM, AND TIM systems at memory locations $2380-$2801 or $0380-$08A1. Or send $1 for a complete catalog to:

6502 Program Exchange
2920 Moana Ln.
Reno, Nevada 89509

©

KIM Tidbits

Expanding Your System

Harvey B. Herman
Department of Chemistry
University of North Carolina
at Greensboro
Greensboro, North Carolina 27412

The January 1981 issue of **COMPUTE!** carried a brief review by me, of a disk operating system for an expanded KIM-1. I have been asked (can you be locked up for talking to yourself?) to supply details on how to expand KIM into a complete system including a disk drive. Even if you have reservations about this, as I am sure some readers do, read on and see how I did it. I am quite pleased with the finished project and feel it may make sense for other KIM owners to follow my lead.

First Steps

My work with microcomputers began in the summer of 1976 when I purchased a KIM-1, a small power supply and teletype (KSR-33). My justification was to have these serve as both an educational tool for me and my students and as an instrument controller. Our first project, a success, was to develop a machine language program, hand assembled by Jeffrey Schmoyer, to control a syringe pump. Hand assembly is a drag and the next natural step was addition of extra memory in order to run an assembler. An article in 6502 Users Notes (November 1976) encouraged me to order a 4K $100 memory board (S.D. Systems, Dallas, TX 75228) and attach it directly to the KIM bus.

At this point I was able to run a small assembler and tiny BASIC. I quickly saw the value of hybrid programs which mixed experimental control (in machine language) and data analysis (in BASIC). However, in general, the later required a floating point BASIC. Microsoft 8K BASIC was, in late 1977, just being offered by Johnson Computer (Medina, OH 44256). This program needed additional memory and I felt compelled to take the next step and add a mother board, KIMS I (Forethought Products, Eugene, OR 97402), which could support $100 8K static memory add-ons, e.g., Econoram II (Godbout Electronics, Oakland Airport, CA 94614).

Willi's Advice

Each step in the expansion of KIM this far was relatively painless with no problems and did not appear overly expensive. Now I was able to run a very nice enhanced BASIC (see previous KIM Tidbits) and a more comprehensive assembler.

The only frustration was loading from cassette tape (even at 6x normal speed using Butterfield hypertape). Obviously a conversion to floppy disk was the way to go but none of the commercial systems was in my budget range. I did have an 8'' disk drive and homemade power supply left over from an aborted project. Willi Kusche (Wilserv Industries, Haddonfield, NJ 08033) noted my dilemma and constraints on a visit here in the later part of 1979. He informed me that with the addition of a disk controller, cable, and software, which he sold, I could get a disk operating system (DOS) up on my KIM. This idea appealed to me and even though the going was not perfectly smooth I'm glad I followed his suggestion and began to put the system together.

Adding The Disk

The first step was to construct the disk controller board (Versafloppy I, S.D. Systems). Table I shows the jumpers we made on the board. The kit was not too difficult to make but one chip was bad and I clumsily blew out several more before I got the board working. After that auspicious start we (Leon Stokes is we, I is me) put together a cable from the controller to the drive. The disk drive was old enough to have a non-standard connector and we could not purchase a ready-made cable. Table II shows the interconnections that were necessary, between the controller and the drive. (Innovex 220M, now Innotronics, Lincoln, MA 01773). Sad to tell on power up the head loaded but nothing else happened. We subsequently found that the original lubricant (not used now) had congealed because the drive had not been used for four years. Our man in the machine shop (Henry Teague) freed things up for us and now everything worked. I was able to load the disk Willi had sent and made a back-up copy with his utility program.

Observations

My students and I have been using this system for slightly over a year. We originally loaded a short disk boot program from tape but are now using a 16K EPROM board (Digital Research Computer, Garland, TX 75040) with just one 2708 programmed by Willi for us. We make extensive use of disk data files which are then used as input to separate data analysis programs. A simple editor program (written by Becky Efird) prepares the data for storage on disk. A listing of the very similar PET version of the program is shown as illustration of the features of the DOS. The KIM version of this program uses the USR function for monitor calls as opposed to SYS for the PET. Otherwise the programs are almost identical.

Disk data files have the decided advantage of allowing recalculation by alternate models without re-keying of data. About 128K bytes of data can be stored on each 8'' disk. While some of the space is taken up by the disk directory and DOS the latter can be optionally omitted on initial formatting. Backup of data files is done with a copy utility and a second diskette or could even be done on tape. As further insurance I am considering a second system to protect against hardware failure.

Conclusions

Would I recommend this approach to others? Yes and no. The additional cost to me for disk operation was nominal as I already had the mother board, extra memory, disk drive and power supply. However, If I was starting from scratch, a complete system based on an Apple or PET would probably have been cheaper. Furthermore, there is a potential noise problem with an exposed single board computer which other commercial expansions might minimize. I have had no problems with this but it should be mentioned.

Working within a limited budget where only a few things could be purchased at a time, I was able to assemble a very nice system. I did have some difficulties but was able to overcome. Get a commercial system if you have the money. If not, consider occasional add-ons with the object of reaching a disk based system at some future point. It really is great bringing up BASIC in less than 4 seconds. You will never be satisfied with less.

Table I

Versafloppy Jumpers for 8'' Innovex 220M Drive	Technical Function
E1 - E2	Port E3/3 to $\overline{\text{DRVSEL 4}}$
E4 - E5	Port E3/2 to $\overline{\text{DRVSEL 3}}$
E7 - E8	$\overline{\text{HLD}}$ to $\overline{\text{HEAD LOAD}}$
E10 - E11	Port E3/3 to $\overline{\text{DRVSEL 4}}$
E15 - E17	HLT(1771) to $\overline{\text{Q}}$(U15)
E21 - E22	4MHz clock (8'' drive)
E24 - E25	pull up on pin 2 of U2
E32 - E33	Ports E3 to E7 (locations F0E3 to F0E7 on KIMSI)

Note -
Some jumpers are etched on boards (rev A) and may need to be cut. Most jumpers are same as given for Altair mainframe with Shugart SA800 disk drive.

Table II

Controller Versafloppy I	8'' Disc Drive Innovex 220 M	Function
36	L6	$\overline{\text{step}}$
34	L15	$\overline{\text{direction}}$
26	L13	$\overline{\text{device select}}$
20	L5	$\overline{\text{index}}$
42	L12	$\overline{\text{track zero}}$
6	L21	$\overline{\text{write current select}}$
18	L18	$\overline{\text{head load}}$
40	L7	$\overline{\text{write gate}}$
22	L8	$\overline{\text{ready}}$
46	L17	$\overline{\text{separated data}}$
38	L10	$\overline{\text{write data}}$
44	L16	$\overline{\text{write protect}}$

NOTE: Power - +5, -5, +24V, and AC made to drive.
Hard sector option on drive was disabled.
Unseparated data sent to controller on separated data line.

©

PRINTING A SYMBOL TABLE FOR THE AIM-65 ASSEMBLER

Richard F. Olivo
Biological Sciences, Smith College
Northampton, MA 01063

The assembler for Rockwell's AIM 65 makes assembly-language programming very convenient, particularly in conjunction with the excellent editor that is part of AIM 65's monitor. However, the assembler does not include an option to print the symbol table, although it does create such a table in memory. The following program is one way of decoding and printing the symbol table. In revising a program, a print-out of the symbol table can be very helpful.

On entering the AIM 65 assembler from the monitor, you are asked for the addresses that start and end the symbol table. The assembler places your answers in zero-page addresses 3A, 3B ("FROM") and 3E, 3F ("TO"). After assembly, the total number of symbols is available in addresses 0B, 0C (in high, low order). The symbol table itself consists of sequential eight-byte entries. The first six bytes of each entry are the symbol name, in ASCII characters (the assembler enters spaces if the symbol is less than six characters), and the last two bytes are the symbol's address, in hex notation.

The program to print the table reads through the table using indirect addressing indexed by Y. It establishes the variable ADDR (at locations 00 and 01), which provides the address of the first character of the current symbol. ADDR is initially set equal to the address in "FROM (3A, 3B); it is incremented by eight after each symbol is printed. For each symbol, the Y register is incremented from zero to seven to access the successive bytes of that symbol.

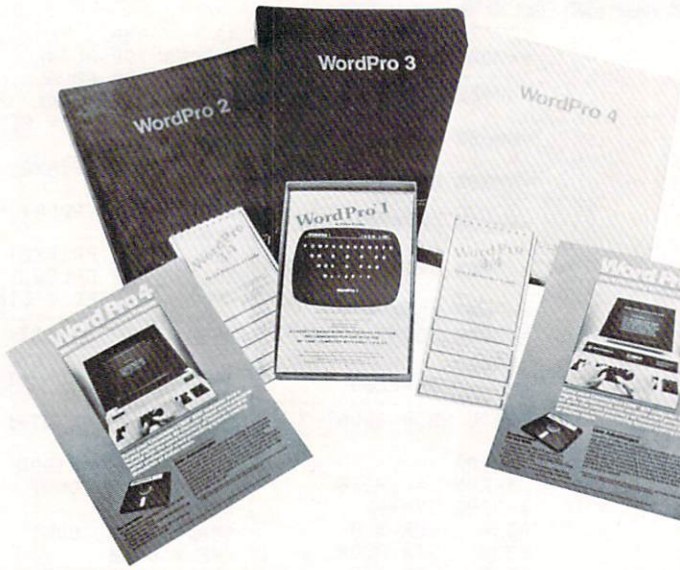
A second variable, COUNT (addresses 02 and 03), keeps track of the number of symbols that remained to be printed. COUNT is initially set equal to one less than the total number of symbols (from addresses 0B and 0C), and it is decremented by one after each symbol is printed. After COUNT reaches zero (the last symbol is numbered zero, which is why the initial count is one less than the total), the program exits and prints the total number of symbols in hex notation. The program uses AIM monitor subroutines to print the ASCII and hex characters. It also turns the AIM printer on and off at the start and end of the table, which I find very handy.

The listing given below places the program at locations 0200-027D, which are available on every AIM 65. The program could of course be placed in other memory locations, and it would be very convenient in a PROM. At the end of the listing, the program was run to list its own symbol table.

```
--------------------
==0000 BLANK=$E83E

==0000 CRLOW=$EA13

==0000 EQUAL=$E7D8

==0000 PRIASC=$E97A

==0000 PRIFLG=$A411

==0000 PRIHX2=$EA46

==0000 ADDR=0
==0000 COUNT=ADDR+2
==0000
       *=$0200
--------------------
INITL ADDR,COUNT,Y

==0200
;"FROM" = 3A,3B
==0200 SYMTBL
A53A   LDA $3A
8500   STA ADDR
A53B   LDA $3B
8501   STA ADDR+1
;ADDR ACCESSES TABLE

A50B   LDA $0B
8502   STA COUNT
A50C   LDA $0C
8503   STA COUNT+1
==0210
;COUNT=SYMBOLS TO GO
C603   DEC COUNT+1
;FIRST SYMB=0, NOT 1

A000   LDY #0
;INDX 8 BYTES/SYMBOL
A980   LDA #$80
8D11A4 STA PRIFLG
;TURN PRINTER ON
2013EA JSR CRLOW
2013EA JSR CRLOW
;SKIP 2 LINES AT TOP
--------------------
MAIN LOOP

==021F SYMLP
B100   LDA (ADDR),Y
C006   CPY #6
;BYTES 0-5 =ASCII
F007   BEQ SPACE
;PRINT 6 ASCII CHAR.
207AE9 JSR PRIASC
C8     INY
4C1F02 JMP SYMLP

;PRINT SPACE & EQUAL
==022C SPACE
48     PHA
203EE8 JSR BLANK
20D8E7 JSR EQUAL
203EE8 JSR BLANK
;NEXT 2 BYTES = HEX
68     PLA
2046EA JSR PRIHX2
C8     INY
B100   LDA (ADDR),Y
==023D
2046EA JSR PRIHX2
2013EA JSR CRLOW
;HAVE PRINTED 1 LINE
--------------------
DECR COUNT & TEST

C603   DEC COUNT+1
A9FF   LDA #$FF
C503   CMP COUNT+1
;FF = BORROW
D006   BNE NXTADR
C602   DEC COUNT
==024D
C502   CMP COUNT
;FF = DONE
F012   BEQ DONE
--------------------
UPDATE ADDRESS

==0251 NXTADR
18     CLC
A500   LDA ADDR
;LOW BYTE
6908   ADC #8
8500   STA ADDR
A501   LDA ADDR+1
;HIGH BYTE
6900   ADC #0
8501   STA ADDR+1
A000   LDY #0
4C1F02 JMP SYMLP
--------------------
PRINT TOTAL & EXIT

==0263 DONE
2013EA JSR CRLOW
A50B   LDA $0B
2046EA JSR PRIHX2
A50C   LDA $0C
2046EA JSR PRIHX2
;PRINT TOTAL,SKIP LN
2013EA JSR CRLOW
==0273
2013EA JSR CRLOW
A900   LDA #0
8D11A4 STA PRIFLG
;TURN PRINTER OFF
4C82E1 JMP $E182
;JUMP TO MONITOR

       END

BLANK  = E83E
CRLOW  = EA13
EQUAL  = E7D8
PRIASC = E97A
PRIFLG = A411
PRIHX2 = EA46
ADDR   = 0000
COUNT  = 0002
SYMTBL = 0200
SYMLP  = 021F
SPACE  = 022C
NXTADR = 0251
DONE   = 0263

000D
```

©

New Products



WordPro Plus Packages For Commodore Computers

Professional Software Inc. of Needham, Mass. has announced the availability of WordPro 3 Plus and WordPro 4 Plus word processing software for use on Commodore Business Machine computers.

In addition to all of the highly advanced features currently found in WordPro 3 and WordPro 4, many new capabilities are now easily accessible. New features include: Math Functions (adding and subtracting columns of numbers within text), Superscripts and Subscripts, Bold Overstrike, Exit to Basic, Variable Lines per Inch, Additional Pitch Settings, Audible Feedback, Pause Command, and Simultaneous Input/Output (WordPro 4 Plus Only).

For all its sophistication, however, a major benefit of WordPro Plus continues to be its "turn key" design and ease of use. Over 4,000 copies of WordPro are currently in use worldwide and are available to Dealers and International Distributors exclusively through Professional Software Inc.

Professional Software also announced that its WordPro Plus programs provide capability for multi-user word processing. Together with the Multi-Cluster, WordPro 3 Plus and 4 Plus allow the use of up to 8 CPU's with one disk drive, thereby creating a real-time "Multi-User" word processing system.

In addition to other forms of dealer support, Professional Software has introduced its exclusive Dealer Demo Pak. This Demo Pak contains demonstration programs of WordPro 3 Plus and WordPro 4 Plus, a Commercial Systems Sales Training Manual with emphasis on word processing sales, four-color WordPro Plus Posters, and an extremely complete WordPro Literature Package designed to give dealers concise and professional marketing support.

For more information contact:
Professional Software Inc.,
166 Crescent Road, Needham, MA 02194. (617) 444-5224 Telex 951579

June Workshops Announced

Virginia Polytechnic Institute And State University has announced a set of workshops. The programs will be directed by Dr. Paul Field, Dr. Chris Titus, Dr. Jon Titus, and Mr. David Larsen. These courses will be on the Virginia Tech campus in Blacksburg, Virginia:

1. Digital Electronics for Automaton and Instrumentation, June 22, 23, 1981.
2. Microcomputer Design Interfacing, Programming, and Application using the Z80/8085/8080. June 24, 25, 26, 27, 1981.

All workshops are hands on with the participant designing and testing concepts with the actual hardware. For more information, contact Dr. Linda Leffel, C.E.C., Virginia Tech, Blacksburg, VA 24061. (703-961-5241).

New Product releases are selected from submissions for reasons of timeliness, available space, and general interest to our readers. We regret that we are unable to select all new product submissions for publication. Readers should be aware that we present here some edited version of material submitted by vendors and are unable to vouch for its accuracy at time of publication.

SYBEX Announces The Pascal Handbook

SYBEX has announced the release of THE PASCAL HANDBOOK, By Jacques Tiberghien ($14.95). Described as a comprehensive dictionary of all features for most existing versions of Pascal, THE PASCAL HANDBOOK contains every symbol, reserved word, identifier and operator for UCSD, Jensen-Wirth, OMSI, Pascal Z, HP1000, ISO and CDC Pascals.

Arranged in alphabetical order, each of the over 180 entries contains the definition, syntax diagram, semantic description, implementation details, and program examples in a format structured for ease in accessibility and application.

This unique book has been designed to facilitate the use of Pascal for everyone: For the novice, it provides immediate access to definitions and examples. For the experienced programmer it is a single source of comprehensive information.

For more information, contact:
SYBEX, INC.
2344 Sixth Street
Berkeley, California 94710
415/848-8233

Clock/Calendar Feature Dropped From Apple III Personal Computers

CUPERTINO, CA — February 10, 1981 — Apple Computer Inc. said today it would no longer offer a special built-in clock/calendar circuit as part of the Apple III personal computer. As a result, the price of the Apple III has been reduced by $50.

The battery operated integrated circuit is not critical to the Apple III's operation. It is used to log time and date information automatically on files the computer has stored. Users, typically those keeping accounting records, can enter this information manually from the computer keyboard.

"We are removing the clock chip from the Apple III computer because we have not been able to obtain a supplier that can meet Apple's rigid quality and reliability standards," said Barry N. Yarkoni, Apple III product marketing manager. "We feel that elimination of this circuit will have a positive effect on Apple III manufacturing schedules," Yarkoni added.

Customers who currently have Apple IIIs will receive a $50 rebate from the company. Letters announcing the change and offering the rebate have been mailed to all Apple III owners who have returned a warranty card. Dealers have also been notified.

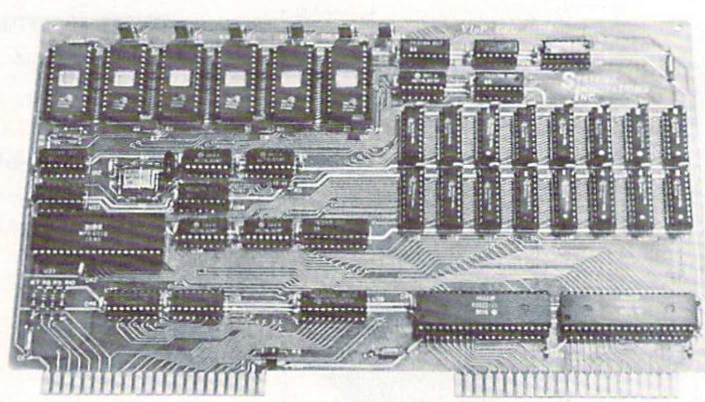
Customer deliveries of the Apple III computer began in November 1980. New U.S. prices start at $4,190.

New VP Marketing At Atari's Computer Division

Sunnyvale, CA — February 2 -Rigdon Currie has joined Atari, Inc. as vice president of marketing for the Computer Division, the announcement made by Raymond E. Kassar, Atari chief executive officer and chairman of the board. In the newly created position, Currie will be responsible for marketing and sales of the Atari 400™ and 800™ personal computers, including peripherals and software.

"We are very pleased to welcome Currie aboard," said Kassar. "He comes to Atari at a time when his marketing experience, both domestic and international, will be particularly valuable. The demand for Atari personal computer systems has

New 6502-Based Single Board Computer



The CPU 65/08 SBC for general purpose industrial and commercial applications is available from Systems Innovations, Inc., Lowell, Mass. Utilizing the popular 6502 Microprocessor, the board will accomodate up to 24K of ROM/EPROM and 8K of RAM.

Two on board VIA's provide 40 I/O lines including 4, 16 bit timer/counters, two serial lines and 14 levels of interrupt. The I/O buss supports DMA and is fully buffered with pinouts equivalent to the KIM 4 standard, thereby allowing the CPU to drive expansion boards directly.

In small quantities, the unit is priced at $275.00 each and is available off the shelf. For further information contact: Systems Innovations, Inc., N.R. Prevett, P.O. Box 2066, 505 Westford Street, Lowell, Mass., 01851

been rising dramatically. Currie's knowledge of worldwide computer marketing should facilitate our sales operation and allow the Computer Division to expand rapidly."

Currie will be reporting to Roger Badertscher, president of the Computer Division.

Previously, Currie spent more than 16 years with the Xerox Corporation in various managerial positions on the corporate staff and with Xerox Data Systems. For the past 3½ years he was vice president of marketing and planning for Diablo Systems, a Xerox subsidiary. While there, he led Diablo to increased sales of word-processing, small business and terminal products in the OEM and retail markets.

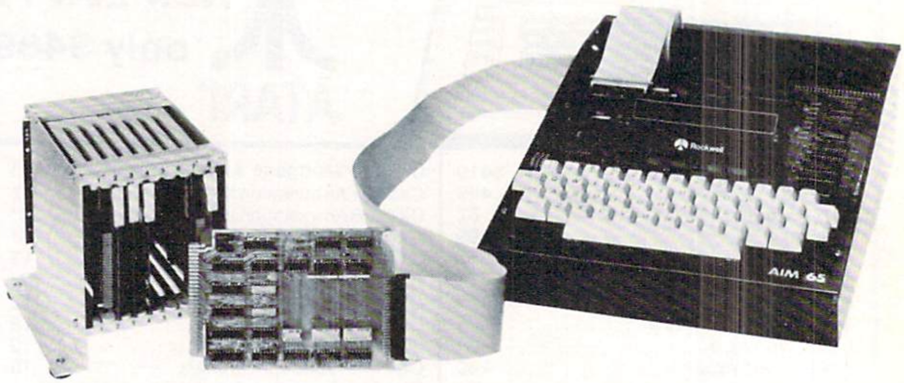
A native of Atlanta, Georgia, Currie received a bachelor's degree in industrial engineering from Georgia Institute of Technology in 1951, and received a master's degree in business administration from Harvard University in 1956.

Currie and his family reside in Palo Alto, California.

STD BUS Expansion for the AIM 65 Computer

As part of its AIM-Mate series of expansion products for the AIM 65, Forethought Products has released its STD-Mate* interface to the STD BUS. By directly substituting for the STD BUS processor card, STD-Mate allows full speed use of STD BUS cards by the AIM 65 both for expansion of existing AIM 65 systems and development/debugging of stand alone STD BUS systems. STD-Mate is available off-the-shelf for $140. Forethought Products, 87070 Dukhobar Rd., Eugene, OR 97402; (503) 485-8575

*TM Forethought Products



Atari Memory Boards

Microtek Peripherals Corporation has announced the release of two memory boards for the ATARI 400/800 computers. The AT-16, a 16k 200ns board, and the AT-32, a 32k 200ns board are designed to be completely compatible with all existing hardware and software, and are user-installable with no modifications. Both products carry MICROTEK's standard one year warranty. Retail pricing is $119.50 on the AT-16 and $199.50 on the AT-32.

Additional products currently available include cables and board extenders. MICROTEK PERIPHERALS plans to introduce a number of ATARI peripherals in the coming months. For further information contact: MICROTEK PERIPHERALS CORPORATION, 9514 Chesapeake Drive, San Diego, CA 92123. 800-854-1081 (in CA 714-278-0630)

Educational Software Announcement

Teacher's Pet offers a listing of over 20 original PET programs for intermediate grade students in math, language arts, and logic. These programs are written by Glenn Fisher, an experienced teacher and programmer who has published several articles on programming techniques. Most programs are drill and practice, with graphics and scoring. They have all been thoroughly tested in classrooms. Complexity ranges from "Times," a simple multiplication facts drill, to "Decimal X," a tutorial which takes a student step by step through decimal multiplication problems at the PET keyboard and allows the teacher to set the size of the problem and the number of decimal places.

Language programs vary from "Parts of Speech," a drill in recognizing word use, to "Comma," a program written so that inexperienced computer-users can change the data easily. Programs offered include a grading program and a program to handle California state enrollment data for elementary school.

All programs have input protected against careless users so they will run without problems in the classroom. Several programs allow 1 to 4 students to enter their names and be scored separately. To request the list of programs, write: Teacher's Pet, Dept. C, Glenn Fisher, 1517 Holly St., Berkeley, CA 94703

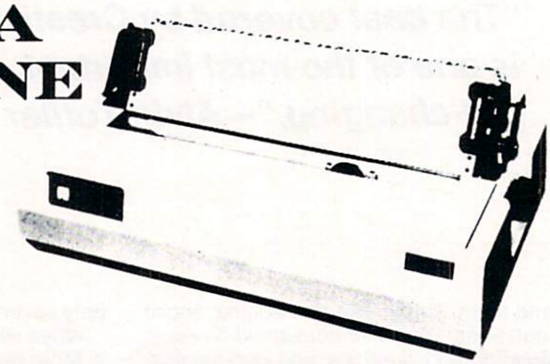
WEST COAST GRAND OPENING!!!

ΩMEGA SALES COMPANY

CALL TOLL FREE 1-800-235-3581

GRAND OPENING
SPECIAL OF THE MONTH!!

OKIDATA MICROLINE 83



$929

DIABLO 630

$1995.00
(TRACTOR $200)

NEC SPINWRITER 5510-5530
$2395.00
(TRACTOR $200)

APPLE II+
48K-$1150.00

C. ITOH STARWRITER 25
PARALLEL OR SERIAL
$1495.00

APPLE DISK DRIVE
DOS 3.3 W/CONTROLLER
$545.00

OKIDATA MICROLINE-80
$469.00

BASE II
$649.00

CBM
CALL FOR PRICE

HAZELTINE 1420
$799

---

1. TWO LOCATIONS TO SERVE YOU BETTER.
2. NO SURCHARGE FOR CREDIT CARDS.
3. WE ACCEPT C.O.D.'S.

ΩMEGA SALES CO - WEST COAST - 3353 OLD CONEJO RD #102
NEWBURY PARK CA 91320 TEL 1-805-499-3678

CAPUTE!

Corrections/Clarifications

In our three part series by Hal Chamberlin, **"Expanding KIM-Style 6502 Single Board Computers"**, we neglected to point out that Mr. Chamberlin is Vice President of Research And Development for Micro Technology Unlimited. You may write him at P.O. Box 12106, Raleigh, NC 27605.

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

Our February, 1981 issue carried an article by Bruce Land entitled, **"A Terminal For 'KAOS' (KIM, Aim, OSI, Sym)"**. In evaluating the Netronics terminal, Bruce said:

"The Netronics 20 ma current loop is not isolated like the Xitrex, and so may not work well with some devices. It does not work well with all the devices I have tried including KAOS systems." (p. 132, column 2)

At least that's what we printed. What Bruce really wrote was:

"It does work well with all the devices I have tried..."

* *

"The 25¢ Apple II Real Time Clock" (February, 1981; page 72, figure 3)

An error in illustration inadvertantly left a tie that "shouldn't bind". Here are both the incorrect and corrected versions:

"Ticker Tape Atari Messages", (February, 1981, page 75)

As you've noticed, there was a problem with the program. Here are the fixes:

1. The [ ]'s used in the listing are ( )'s. Nobody we talked to had a problem with this, but we thought we'd mention it.

2. In line 20, B is set equal to a blank; this should be B$.

3. Add a line 16 that sets Y$ = W$.

4. Finally, the contents of W$ and Y$ determine your "moving borders"; if you put 20 spaces in W$, you won't see anything. If you put in * alternating with spaces, then you'll see what we mean.

Play around with it, and compose your own borders.

* *

Program Listings for COMPUTE

Cursor control characters will appear in source listings as shown below:

```
h=HOME          , ĥ=CLEAR SCREEN
↓=DOWN CURSOR   , ↑=UP CURSOR
→=RIGHT CURSOR, ←=LEFT CURSOR
r=REVERSE       , r̂=REVERSE OFF
```

Graphics (i.e. shifted) characters will appear as the unshifted alphanumeric character with an underline. This does not apply to the cursor control characters. The Spinwriter thimble doesn't have a backarrow symbol, so a "~" is used instead.

The "¬" is used to indicate the beginning of a continuation line. It is also used to indicate the end of a line which ends with a space. This prevents any spaces from being hidden.

FIGURE 3

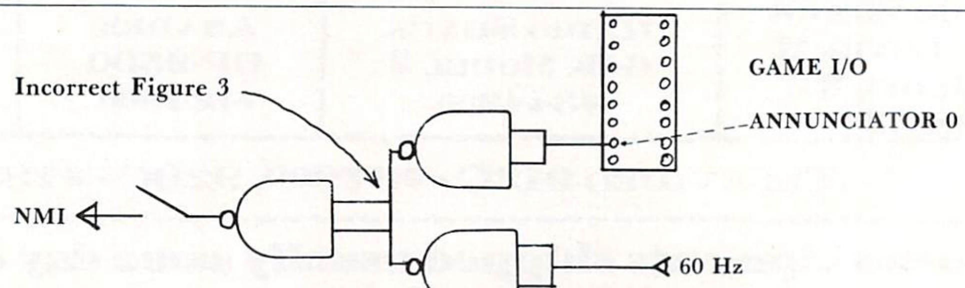
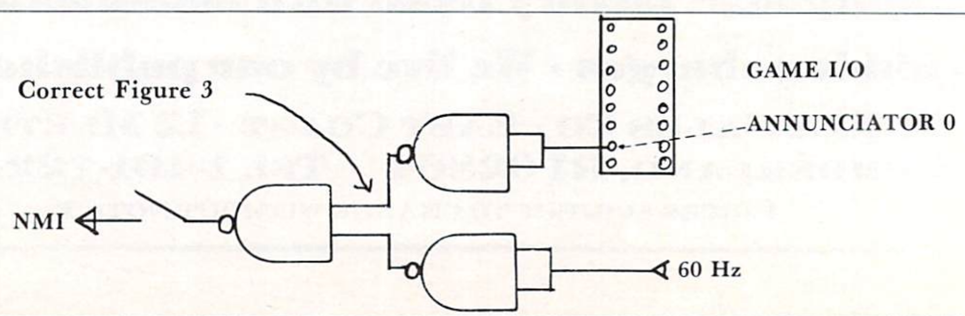
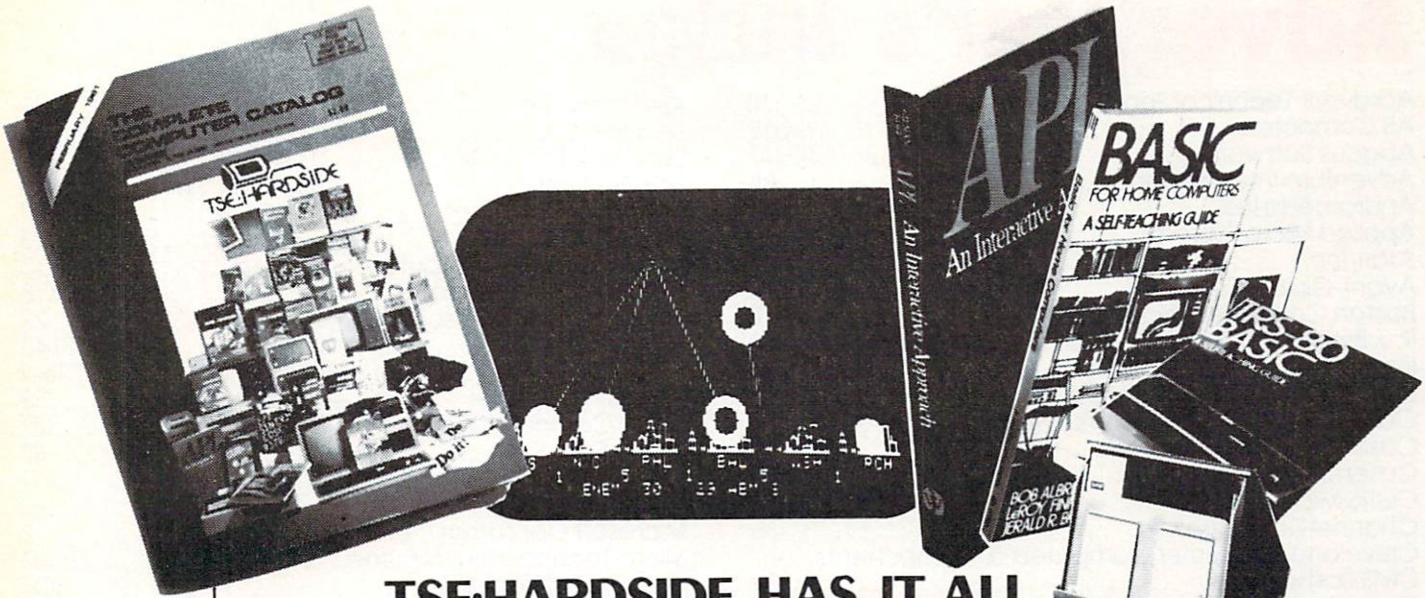


FIGURE 3




THE COMPLETE COMPUTER CATALOG
TSE-HARDSIDE
APL An Interactive Approach
BASIC FOR HOME COMPUTERS
A SELF-TEACHING GUIDE
TRS-80 BASIC

Advertiser's Index

The 6502 Resource Magazine
COMPUTE! PET•ATARI•APPLE
OSI•KIM•SYM•AIM

My computer is:

☐ **PET** ☐ **APPLE** ☐ **ATARI** ☐ **OSI**
☐ **KIM** ☐ **SYM** ☐ **AIM** ☐ **OTHER**____________
☐ **Don't yet have one...**

During the next year I expect to buy:

☐ **computer** ☐ **printer**
☐ **disk drive** ☐ **other peripherals**

☐ Please enter my 1 year (12 issue) subscription to COMPUTE!
☐ New subscription ☐ Renewal subscription
☐ $20.00 U.S. subscription
☐ $25.00 Canada and international surface mail: Payment in U.S. funds, drawn on U.S. bank or money order.
☐ Bill me (U.S. only). $1.00 billing fee.

Name, Address: ________________________

Charge my: ☐ Visa ☐ MC
Number____________________ Expires_____/_____/

4 5 6 7 8 9 10 11 12

The Editor's Feedback:

My computer is:

☐ **PET** ☐ **APPLE** ☐ **ATARI** ☐ **OSI**
☐ **KIM** ☐ **SYM** ☐ **AIM** ☐ **OTHER**____________
☐ **Don't yet have one...**

During the next year I expect to buy:

☐ **computer** ☐ **printer**
☐ **disk drive** ☐ **other peripherals**

Content:
Best Article This Issue (page #, title)

____

Other suggestions:

4 5 6 7 8 9 10 11 12

The 6502 Resource Magazine
COMPUTE! PET•ATARI•APPLE
OSI•KIM•SYM•AIM

My computer is:

☐ **PET** ☐ **APPLE** ☐ **ATARI** ☐ **OSI**
☐ **KIM** ☐ **SYM** ☐ **AIM** ☐ **OTHER**____________
☐ **Don't yet have one...**

During the next year I expect to buy:

☐ **computer** ☐ **printer**
☐ **disk drive** ☐ **other peripherals**

☐ Please enter my 1 year (12 issue) subscription to COMPUTE!
☐ New subscription ☐ Renewal subscription
☐ $20.00 U.S. subscription
☐ $25.00 Canada and international surface mail: Payment in U.S. funds, drawn on U.S. bank or money order.
☐ Bill me (U.S. only). $1.00 billing fee.

Name, Address: ________________________

Charge my: ☐ Visa ☐ MC
Number____________________ Expires_____/_____/

4 5 6 7 8 9 10 11 12

Place
Stamp
Here

COMPUTE.
Post Office Box 5406
Greensboro, NC 27403

Place
Stamp
Here

COMPUTE!
Post Office Box 5406
Greensboro, NC 27403

Place
Stamp
Here

COMPUTE!
Post Office Box 5406
Greensboro, NC 27403

The Great American Solution Machine.

Meet Commodore. The business computer that's providing solutions for more than 100,000 people all over the world. Built by one of the pioneers in office machines. With a reputation for quality that can only come from vertical integration of the total manufacturing process. Commodore builds, not assembles.

Compare Commodore's word and data processing capabilities with computers costing twice or even three times as much. You'll see why so many small businesses are turning to Commodore for solutions to problems as varied as these:

☐ A car leasing company's customers were terminating too early for account profitability. Solution: A 16K Commodore. It analyzes cash flow on over 1200 accounts, identifies those for early penalties, and even writes up lease contracts. Commodore paid for itself within weeks.

☐ A law firm needed a high quality, easy-to-use, affordable word processing system. Solution: Commodore plus its WordPro software package. At a $6,000 savings.

☐ A gasoline retailer needed to inventory, order and set prices; determine Federal and state income taxes; and comply with Federal pricing and allocation regulations. All done daily, weekly, monthly and yearly. Solution: Commodore. It keeps his business on track—and Uncle Sam off his back.

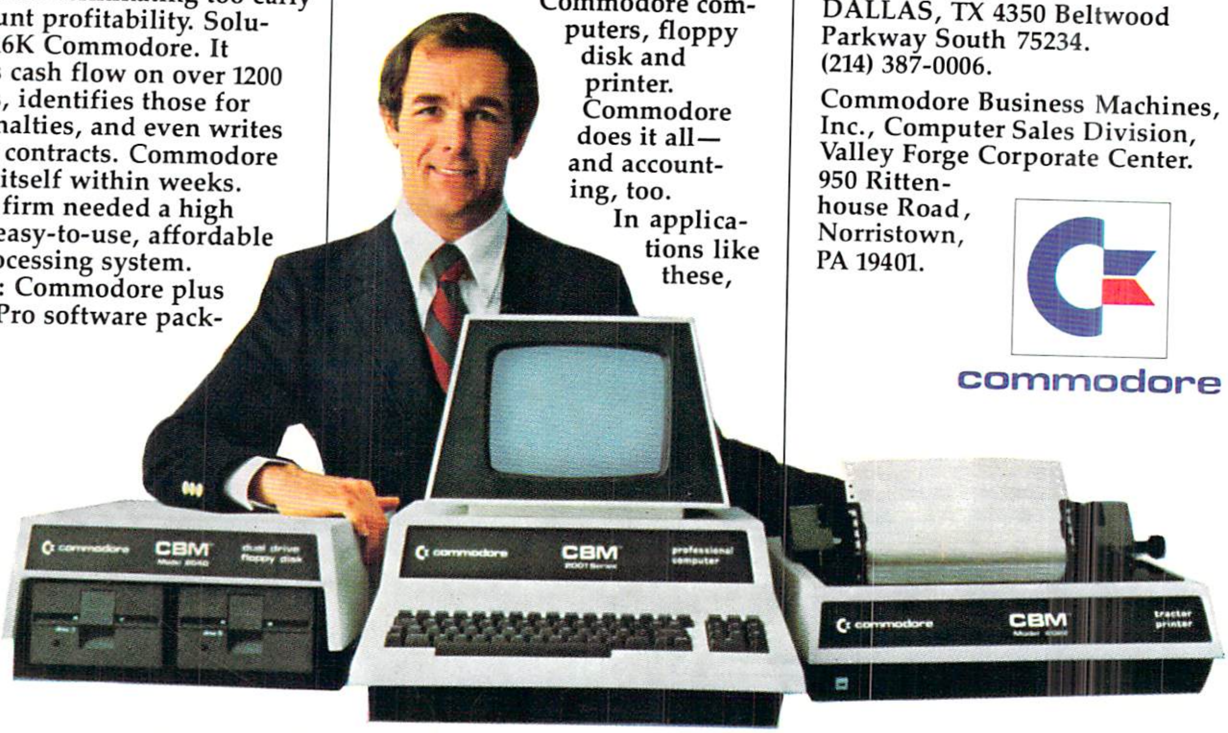
☐ A paint and wallpaper store had to inventory over 600 expensive wallpaper lines for profitability, monitor distributor sales, set and track salesmen's goals, and help the customer select the right size, pattern and quantity. Solution: Two 32K Commodore computers, floppy disk and printer. Commodore does it all—and accounting, too.

In applications like these, and many more, Commodore solves the problems that stand in the way of increased profitability. Commodore can provide the solution in your Great American business, too. Find out more by calling or writing any of Commodore's District Sales Offices.

COSTA MESA, CA 2955 N. Airway Avenue 92626. (714) 979-6307.
SANTA CLARA, CA 3330 Scott Boulevard 95051. (408) 727-1130.
DECATUR, GA 5360 Snapfinger Woods Drive 30035. (404) 987-3311.
BENSENVILLE, IL 790 Maple Lane 60106. (312) 595-5990.
NORRISTOWN, PA 950 Rittenhouse Road 19401. (215) 666-7950.
DALLAS, TX 4350 Beltwood Parkway South 75234. (214) 387-0006.

Commodore Business Machines, Inc., Computer Sales Division, Valley Forge Corporate Center. 950 Rittenhouse Road, Norristown, PA 19401.

commodore



COMMODORE